Appl Intell
DOI 10.1007/s10489-016-0784-0

Biased sampling from facebook multilayer activity network using learning automata

Ehsan Khadangi[1] · Alireza Bagheri[1] · Amin Shahmohammadi[2]

© Springer Science+Business Media New York 2016

Abstract Although much research has been devoted to unbiased sampling of various networks, bias is not always disadvantageous, but sometimes useful. Especially for many real-world applications such as detecting influential nodes, spam users, and the most trustful people, it is preferred to sample users with special properties. Since sampling from friendship network alone cannot collect these important nodes appropriately, one may use interactions occurred among users. This paper deals with biased sampling of multilayer activity network. The proposed method initially learns the transition probabilities according to the considered application using learning automata. Then we sample the graph by running an application-based random walk following the learnt probabilities, in order to be guided to suitable nodes and collect their information. At last, the performance of the proposed method in terms of different applications such as fame, spam, and trust is evaluated and compared with those of common sampling algorithms. According to the experiments done, biased sampling method based on learning automata outperforms all other sampling approaches including simple random walk, Metropolis-Hastings random walk, BFS, forest fire, degree, and uniform sampling in terms of all the evaluation measures. To the best of our knowledge, our method is the first and only biased sampling method which can be used in a multilayer activity network.

Keywords Network sampling · Social network analysis · Activity network · Facebook · Learning automata · Social media marketing

1 Introduction

Complex networks are almost present everywhere, and are studied by researchers for different purposes. The wide range of networks such as those of social, collaboration, trust, mobile sensor, web, biological, and different interactions all belong to complex networks. As the popularity of web 2.0 has grown, online social networks have developed swiftly and now websites such as Facebook, Twitter, LinkedIn, and Google+ are among the most common methods of people's communication. These websites have changed the pattern of people's interactions and have caused many people to spend much time on surfing and doing activities therein. For example, Facebook had 12 million users in 2006. This number reached 600 million at the end of 2010, and now it is around 1 billion and 300 million, which shows the increasing appeal of social networking services. Since the number of social network users, especially Facebook, is very high, researchers develop their methods on a small sample of the network and generalize it to the whole

✉ Alireza Bagheri
ar_bagheri@aut.ac.ir

[1] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

[2] Computer Engineering Department, Pooyesh Institute of Higher Education, Qom, Iran

Published online: 14 May 2016

network. Therefore, many methods have been proposed for social network sampling. Since in most cases, the whole network is not available, researchers often use graph traversal techniques. So far, many works have been done in the field of unbiased sampling of complex networks, but works focusing on biased sampling are few in number. It should be noted that biased sampling means that sampling is biased toward choosing the nodes which are important according to the target application.

The structure of a social network can contain useful information for different applications. Based on the structural properties of the network, the sampler may be guided in a way that more influential nodes be collected. In this paper, influence is defined by special applications using application-based centrality measures. In addition, the studied network in most papers is the friendship network. This network shows users' friendship relationship but their closeness and interactions are left aside. For solving this problem, some works have studied the activity network [1–3]. In such a network, nodes represent users and links model activities and interactions between them [4]. It is known that an activity network represents relationships and interactions among users better than a friendship network [1, 5]. Therefore, the structure of this network can give us valuable information for biased sampling of the network. Although Ahmed et al. presented several papers on sampling activity networks, their works usually dealt with sampling from social network activity streams rather than traversing the activity network and gathering samples from it [6–8]. In this paper we present a learning random walk which learns to move toward influential nodes by walking on the multilayer activity network. Random walk probabilities are learned by means of learning automata.

The paper is organized as follows. Related works are presented in Section 2. In Section 3, the methodology and problem definition are elaborated. The proposed method based on the learning automata is presented in Section 4. Section 5 includes the experimental results and finally the paper is concluded in Section 6.

2 Related works

Many works have dealt with sampling different complex networks [9, 10]. Among these, sampling social networks as significant examples of complex networks is widely considered due to their swift development in recent years. Since the topology of social networks is hidden, most of social network sampling methods are based on graph traversal. The most important examples include breadth first search (BFS) [11], depth first search (DFS) [11], forest fire sampling (FFS) [10], snowball sampling [12], simple random walk (RW) [13], Metropolis Hastings random walk (MHRW) [14], reweighted random walk (RWRW) [15], and respondent driven sampling (RDS) [16].

Simple random walk BFS and snowball sampling are biased toward high degree nodes. The bias of random walk in undirected networks has been characterized, thus several works seek to analyze and remove the sampling bias.

Some methods adjust transition probabilities of random walks in order to remove the bias. A well-known example is the Metropolis-Hastings random walk [9]. In contrast to common methods, Rezvanian et al. [17], used distributed learning automata for unbiased sampling of complex networks. They tried to correct the sampling bias using distributed learning automata by learning transition probabilities. Torkestani and Akbari also used learning automata to guide web crawlers towards the most relevant URLs [18]. Some other methods like reweighted random walk and RDS correct the bias after sampling in the approximation step. Gile and Handcock correct the bias of RDS for sampling hidden networks [19]. Subsequently Salehi et al. [20] used this method for sampling from Twitter social network. They showed that RDS is less biased than MHRW. Gjoka et al. [9, 21] tried to perform unbiased sampling of Facebook friendship network. They showed that reweighted random walk and Metropolis-Hastings random walk perform better when compared to simple random walk and BFS. Although many works have been done on unbiased sampling of undirected networks, few works have focused on unbiased sampling of directed networks. However, recently some works have dealt with analyzing and correcting the bias of random walk in directed networks [22, 23].

Contrary to random walk based methods, most of social network analysis works applied their research on networks sampled by BFS-based crawling [1, 24–26], regardless of the bias of these methods. Although Kurant et al. analyzed the bias of BFS and presented a method which corrects its bias to some extent [27], BFS bias is still not characterized. In addition, some works have focused on representative sampling of different complex networks. In these works, researchers are seeking a sub-network which represents some properties of the original network, especially the community structure [28, 29]. There are also some works that study characteristics of the sampled network [30, 31].

In addition to crawling-based methods described above, some works have involved random nodes/links sampling. When the sampling objective is the extraction of a subgraph, these methods do not lead to a suitable result but are very accurate for measuring the properties of nodes and have been used as ground truth for evaluating other

methods [9, 32, 33]. In some social networks such as Twitter and LiveJournal, the list of users' recent activities together with their names is saved somewhere called timeline. Accordingly, some papers have used timeline for sampling from these networks [34, 35].

Although many works have focused on unbiased sampling, bias is not always disadvantageous but sometimes useful. Maiya et al. [36] studied the bias of different network sampling methods. They concluded that bias is sometimes useful since biased sampling collects nodes which may be suitable for the considered application. After presentation of "Benefits of Bias" by Maiya et al., few researchers have paid attention to biased sampling. Notably, Zhang et al. [37] tried to gain the core network and accordingly the influential people in the social network Sina Weibo using sampling. After studying different methods, they concluded that snowball sampling was the best for their purpose. In another work, Wang and Lu presented star sampling and used it for collection of top bloggers, i.e. users with many followers. They showed that star sampling is more efficient than random walk [38].

A suitable method for imposing the appropriate bias on random walks is the employment of a learning method so that the walker learns how to walk in the network. In this way, the random walk agent gradually learns to visit important nodes. For example, Backstrom and Leskovec presented supervised random walk. In their method, the random walk agent starts from a node and learns to move toward nodes which are likely to have future links with the first node [39]. They used this sampling method for prediction and recommendation of new links. This paper aims at biased sampling of Facebook activity network by learning the probabilities of transition using learning automata. The reward and penalty of the automata occurs based on the information of users' different interactions including *like*, *comment*, *post* and *share* and the sampling is done in simple and multilayer activity networks.

In the following, we briefly describe the sampling algorithms used in our experimental studies.

2.1 Simple random walk

Simple random walk is very common in social network sampling. This algorithm starts with an initial node. At each step, one of the neighbors of the current node is selected uniformly at random and the walker visits it and the same action is repeated iteratively. It can be shown that in a connected undirected graph, the probability of being at the particular node v converges to (1):

$$\pi_v = k_v \big/ 2\,|E| \tag{1}$$

Where π_v is the probability of visiting node v, k_v is the degree of v, and $|E|$ equals the number of links [9, 32]. Therefore, simple random walk is biased toward high-degree nodes. Since the bias of random walk is characterized, some methods have been presented for correcting the bias of random walk, out of which reweighted random walk may be mentioned. Algorithm 1 shows the pseudo-code of simple random walk.

Algorithm 1 Random walk sampling

```
v ← initial node
while stopping criterion not met do
Select node w uniformly at random from neighbors of v.
v ← w.
end while
```

In reweighted random walk, the degree bias is corrected by reweighting the measured values. This is done using Horvitz-Thomson estimator [40].

2.2 Metropolis-hastings random walk

In MHRW, instead of correcting the bias after sampling, the transition probabilities are changed in a way that the walk converges to the uniform distribution. The probability of transition from node v to its neighbors w follows (2) [9, 32].

$$p_{v,w}^{MH} = \begin{cases} \frac{1}{k_v} \min\left(1, \frac{k_v}{k_w}\right) & w \in N(v) \\ 1 - \sum_{z \neq v} p_{v,z}^{MH} & w = v \\ 0 & otherwise \end{cases} \tag{2}$$

Where $N(v)$ is a set of v's neighbors, and k_v and k_w are degrees of nodes v and w.

2.3 BFS (Breadth First Search)

BFS is among the oldest and most common methods of graph traversal. BFS starts from an initial node v and visits all its neighbors. At each step, the oldest node which has not been chosen so far is chosen and all its neighbors are visited. This method visits all the nodes with a particular distance from the initial node [11].

2.4 Snowball sampling

Snowball Sampling traverses the graph similarly to BFS. At each stage of this algorithm, named wave, a limited number of the current node's neighbors are visited [41]. In this method, V_0^* shows the initial nodes. Snowball sampling extends these nodes to $V_1^* \subseteq N(V_0^*) \cap \bar{V}_0^*$, where $N(V_0^*)$ is

a set of V_0^*'s neighbors. $\bar{V}_0^*$ also equals $V - V_0^*$ where V is the set of nodes. The second wave of the sampling process is also $V_2^* = N(V_0^*) \cap \bar{V}_0^* \cap \bar{V}_1^*$. The process continues until we reach the final sample as follows:

$$V^* = V_0^* \cup V_1^* \cup V_2^* \cup \ldots \cup V_k^*$$

[12] describes snowball sampling more accurately.

2.5 Forest fire sampling

Leskovec and Faloutsos presented Forest Fire Sampling for using the benefits of the two methods Breadth First Search and random walk. Initially, node v is chosen uniformly at random. At each iteration of the algorithm, a number of neighbors of the current nodes are chosen randomly. The number of chosen nodes at each stage is defined by a parameter. Such involvement with sampling the network has originated from how the forest starts to light. At each stage, node v and its connected links are burned. If a link is burnt, the node at the other side finds the chance of burning. The implementation is such that a random number x is initially produced with geometric distribution and the average $\frac{p_f}{1-p_f}$ and the equal number of neighbors of node v will be chosen. Leskovec and Faloutsos recommended using $p_f = 0.7$. In this method, nodes are not visited more than once. If the fire is put off, it lights again by another random node. The pseudo code of the Forest Fire Sampling is presented in Algorithm 3 [10, 42].

Algorithm 2 Forest fire sampling

```
Input: graph g, sample size maxIter, initial node vinit,
  forward burning probability Pf
Select initial node v
While stopping criterion does not met do
        Generate a random number x geometrically
        distributed with mean Pf/(1-Pf)
Select x nodes uniformly at random from out-neighbors
of v
Add the nodes to queue Q
Select a node w from queue Q
V ← w
End
```

2.6 Degree sampling method

Degree sampling method consists of greedily choosing the node $v \in N(S)$ with the biggest degree. For this, the number of neighbors for each $v \in N(S)$ should be known This means that at each step, we should know the neighbors of the current node's neighbors [43]. Algorithm 3 shows this method.

Algorithm 3 Degree sampling (DS)

```
Input: graph g, sample size maxIter, initial node vinit
Output: Sequence of sample nodes S
v ← vinit
Add v to S
neighbor=all the neighbors of node v in graph g and v
itself
mark v as selected
i ← 2
while i < maIter do
  add new nodes from neighbors of v to the neighbor
  compute degree of new nodes node in graph g
  w ← an unselected node in neighbor with maximum
  degree
  Mark node w as selected
  Add w to S
  i ← i +1
  v ← w
end while
```

2.7 Sample edge count method

In this method, after visiting each new node, the existent links between the new node and the old chosen nodes are also added to the sample. This algorithm is like degree sampling, with the difference that at each stage the nodes with the highest degree in the sampled graph are chosen [36].

3 Methodology

In this section, we present the definition of biased network sampling problem. Then high level characteristics of the used dataset are presented. After that some of the important definitions such as special users, and weights of activities are introduced. We finally propose two application-based centrality measures for evaluating the sampled network according to the amount of bias.

3.1 Problem definition

The input of the biased sampling algorithm is a directed network but it can also be used to sample undirected networks. In this problem, the directed and connected graph G(V, E) is given, where V and E are the set of nodes and links respectively. The integer number n as the size of the sample and the node $v^0 \in V$ as the seed node are given. The objective is to learn transition probabilities of each link so as to sample the graph G using a random walk based method and form this subgraph consisting of suitable nodes according to application-based measure. For this problem, different

Table 1 High level characteristics of activity and friendship networks

	Clustering coefficient	Avg. path length	# of components	Size of giant component
Mixed	0.16	5.4	15	0.99
Like	0.15	5.4	23	0.98
Comment	0.09	7	107	0.93
Post	0.039	6.2	198	0.75
Share	0.04	5	181	0.6

applications can be considered, out of which trust, closeness, fame, spam, and politeness are focused on this paper.

3.2 Dataset

The collected data include profile information, and the friendship network of 36204 Facebook users. In addition, for friend users, the information about the number of activities including *like*, *comment*, *post, and share* as well as the number of exchanged words in their comments are available every one month over a period of 3 years from January 1, 2011 to January 1, 2014.

3.2.1 Sampling process

In the activity network, the links usually exist only between the nodes which are friends. Therefore, we collected friendship network before collecting users' profile and interaction data. Due to the users' privacy settings, the timeline and friend list data was not accessible for some users. Accordingly, a two-stage method was implemented for data collection. The stages are as follows:

1. Sampling the Friendship Graph: In this stage, we sampled nodes whose friend list was available and ignored the rest
2. Collecting the timeline and profile information of users whose friend list was available

According to these steps, the friendship graph was initially obtained by using several threads on one computer. Then we gathered the activities and profile information of users with several threads on different computers. Finally, six activity networks were composed to form the final activity network. It is important to note that the collected data was thoroughly anonymized to protect the identities of sampled users.

3.2.2 High level analysis of the collected data

Table 1 presents the high-level characteristics of the friendship network besides activity networks of *like*, *comment*, *post*, *share* and *mixed*, which is a mixture of the previous four.

Figures 1 and 2 also show the indegree distribution of different activity networks. It is seen that the indegree distributions of all activity networks in log-log scale are approximately linear. Therefore, indegree distributions of different activity networks follow the power-law. It should be noted that the outdegree distribution is similar to that of the indegree.

Figure 3 also shows path length distribution in different activity networks.

As can be seen in Fig. 3, most path lengths in all networks except *post* and *share* are lower than 6 and the average path length in these networks is low. Therefore, these networks are small-world. In *share* and *post* networks, this property is seen with a lower degree.

Table 2 also presents reciprocity of various interaction networks. As seen therein, contrary to what was thought so far, the activity network is not highly reciprocated. Therefore, it is better to model the activity network as a directed graph.

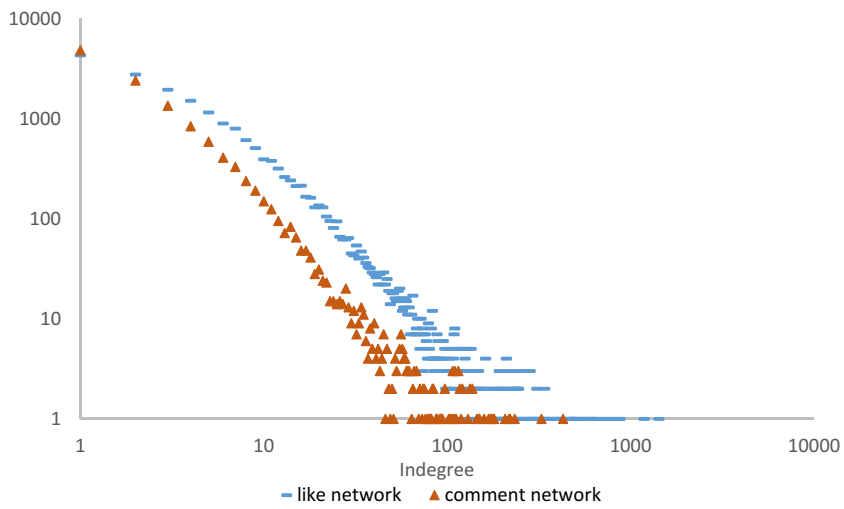


Fig. 1 In-degree distributions of like and comment activity networks

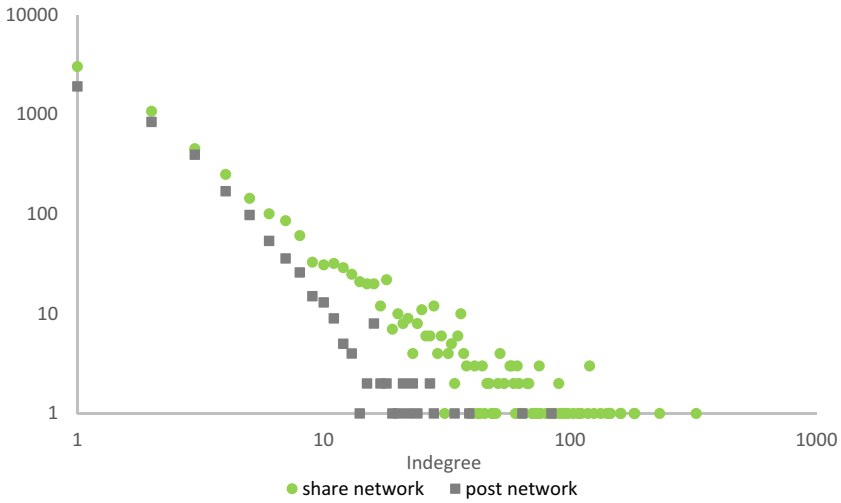


Fig. 2 In-degree distributions of *post* and *share* activity networks

Based on the analysis in [5] as well as the diagrams and tables presented above, we conclude that the friendship and activity networks are intrinsically different. In addition, the activity network is a better representation of the real world. Therefore, it is better to learn transition probabilities based on the information of users' activities, and the sampling should be carried out over the activity network rather than the friendship network.

3.3 Learning automata

Learning automata is one of the most common reinforcement learning schemes. It has been used for solving different problems in many fields such as network sampling [17], social network recommender systems [44], network routing [45], and etc.

In this method, the automata interacts with a random environment. The learning automata improves its performance while interacting with the environment by learning how to choose the optimum action from a finite set of actions. At each state, the action is chosen at random based on the learnt probability distribution. After performing an action, the environment gives a response to the automata, which transfers the automata to another state. The automata updates its action probability vector according to the environment's response. Accordingly, if the new state is better than the previous one, the taken action is rewarded and if worse, it may be penalized or nothing may happen. If it is rewarded, the probability of choosing the action which has newly been chosen increases for the next steps. At the end of the learning process, the probability vector of the learning automata is updated so that the average penalty

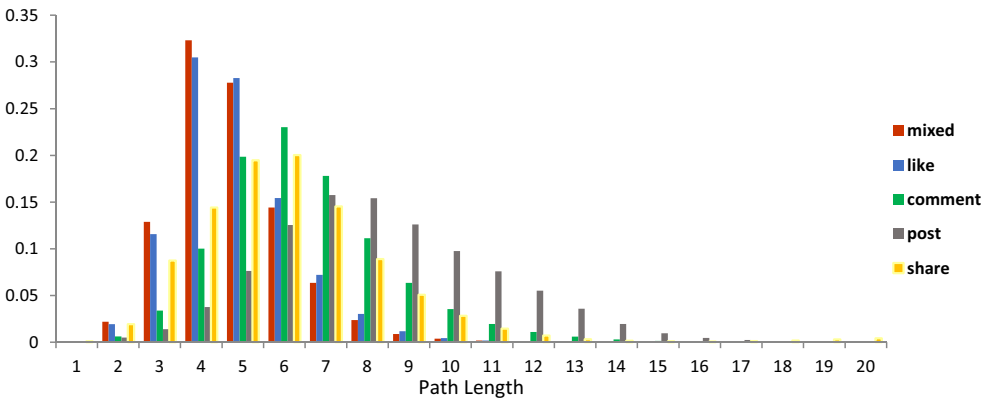


Fig. 3 Path length distribution in different activity networks

Table 2 Reciprocity of Facebook interactions

Network	Reciprocal links in unweighted network
Like	33 %
Comment	26 %
Post	6 %
Share	11 %
Mixed	36 %

is minimized. Figure 4 shows how the learning automata communicates with the environment.

Suppose that $P_j(n)$ is the probability of the j_{th} action to be chosen at stage n. If the automata is rewarded, the probabilities of actions are updated by (3).

$$P_j(n+1) = \begin{cases} P_j(n) + a\left[1 - P_j(n)\right] & j = i \\ (1-a)\,P_j(n) & j \neq i \end{cases} \quad (3)$$

Where a is the reward rate, and i is the rewarded action. Conversely, when the automata is penalized for the action i, the probabilities are updated using (4).

$$P_j(n+1) = \begin{cases} (1-b)\,P_j(n) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)\,P_j(n) & j \neq i \end{cases} \quad (4)$$

Where b is the penalty rate, and r is the number of actions that can be chosen by the automaton. If b equals 0, the automata will be of the type L_{RI}, meaning linear reward-inaction. If a and b are equal, then the automata will be of type L_{RP} which is linear reward-penalty. If a is much bigger than b, then the automata will be of type $L_{R\varepsilon P}$ which means linear reward ε-Penalty [17].

3.4 Some definitions

Some concepts used in the rest of the paper are defined in the following.

3.4.1 Importance of activities

To guide the random walk towards visiting the influential nodes, we use the information of all activities as much as possible by measuring the importance of each activity. Each activity has its own importance according to the considered application.

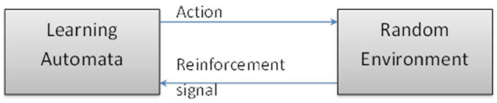


Fig. 4 Diagram of a learning automata and its relationship with environment

To calculate the importance of activities, we initially implemented a Facebook application which asked questions from different Facebook users about their friends. Then according to the replies, we calculated the trust, and closeness between 506 pairs of users. In addition, we collected the number of *like*, *comment*, *post*, and *share* activities between pairs. After data collection, we applied linear regression on the collected data and the coefficients of different activities were obtained for calculating trust and closeness. Since the number of activities (as input variables) and trust and closeness (as output variables) were all positive, the calculated coefficients were all positive. Consequently, we can use the calculated coefficients as the importance of different activities. Obviously, these values aren't accurate enough because the similarity of users based on different demographic attributes can influence the amount of trust and closeness between pairs. Therefore, we used information gain and PCA to improve the result of linear regression.

For this end, we discretized the amount of trust and closeness. Then we used PCA and information gain as two popular attribute weighting methods to calculate the weight of different attributes. In this stage, the input attributes were numbers of like, comment, post, and share in addition to similarity of users' pairs based on age, gender, mutual friends, groups, events, and pages. Then we normalized the calculated weights and averaged them with equal weights. The weights of different activities thereof were calculated as the average of values from attribute weighting and linear regression coefficients and presented as follows.

1. Trust: $W_{like} = 9$, $W_{comment} = 7$, $W_{share} = 23$, $W_{post} = 11$
2. Closeness: $W_{like} = 8$, $W_{comment} = 16$, $W_{share} = 10$, $W_{post} = 16$

As can be seen, the importance of different activities is different for different applications.

In addition, in order to calculate the importance of nodes in terms of the time spent, by analyzing the collected data, the average number of words in each comment is calculated as 10.7. In addition, upon some users' posts which lacked any pictures, the average number of words in each post is gained 22.5, according to which and the fact that some posts included pictures it can be estimated that each post takes approximately 2.5 times of each comment. Based on the calculations done and the knowledge of different activities gained after interviewing 10 active Facebook users, the concluded weights are as follows:

1. Time: $W_{like} = 1$, $W_{comment} = 4$, $W_{share} = 4$, $W_{post} = 10$

As can be seen, the weight of the like activity is very low since it occurs by only a mouse click and actually takes the user very little time.

3.4.2 Special users

Further to trust, closeness, and time applications already mentioned, we want to sample from special nodes such as famous, spam, or polite nodes We explain these special nodes as follows.

Famous users Some Facebook users have gained fame outside online social networks and use social networking services to communicate with people. Among these are actors and actresses, politicians, sportsmen, sportswomen, singers, and famous scientists. Finding these people and collecting their information may be very useful for some applications such as collecting data from famous people, finding important users for social media marketing, and automatic verification of celebrities. It should be noted that the high value of different centrality measures does not equal the user's fame, as popular centrality measures show the user's importance only based on the position of the user inside the social network, not outside it.

Since rewards and penalties in our proposed methods in Section 4 should be calculated according to different applications in every training step, we should define the amount of fame (or being spam) as simple as possible. Therefore, we should present a rationale and simple formula for calculating the amount of fame and being spam for Facebook users only based on the number of interactions occurred between them without considering textual content and profile information. Since there was no implicit spam/fame attribute, we reviewed the dataset manually and extracted some of the celebrities in our dataset. We also extracted some spam-like users based on their username, number of activities, etc. manually. By analyzing these users, we realized that:

1. One of the important difference between celebrities and others is the high amount of incoming activities versus outgoing activities. In other words, in famous users $N_{in}\,(activity) - N_{out}(activity)$ is generally very higher than the average.
2. In spam users, the amount of $N_{out}\,(activity) - N_{in}(activity)$ is much higher than the average.

However, some ordinary users also have high amount of outgoing versus incoming activities, and vice versa. Therefore, we used the concept of purity in our fame and spam equations as follows.

$$Fame\,(i) = \frac{\sum_{a\in\{like,comment,post,share\}} \left[\left(\max(inweight_a\,(i) - outweight_a\,(i)\,,0)\right) * W_a * purity_a\,(i)\right]}{\sum_{a\in\{like,comment,post,share\}} W_a} \tag{5}$$

where

$$inweight_a\,(i) = \sum_{j\in in-neighbors\{i\}} w^a_{ji}$$

and

$$outweight_a\,(i) = \sum_{j\in out-neighbors\{i\}} w^a_{ij}$$

W_a also shows the importance of the activities $a \in \{like, comment, post, share\}$. In this paper, we used $W_a = 1$ for all activities. However, we can calculate the importance of activities for fame and spam applications using the method presented in Section 3.4.1

For calculating purity, we used two popular uncertainty measures from information theory: Entropy and Gini Index. According to [46] Entropy and Gini index are two popular methods for calculating impurity of a data partition. As we want to calculate the purity of incoming/outgoing activities sets, we used 1-*Entropy* and $1 - Gini$ for this end as follows

$$\begin{aligned} purity_a\,(i) &= 1 - gini_a\,(i) = \left(\frac{inweight_a\,(i)}{weight_a\,(i)}\right)^2 \\ &\quad + \left(\frac{outweight_a\,(i)}{weight_a\,(i)}\right)^2 \end{aligned} \tag{6}$$

$$\begin{aligned} purity_a\,(i) &= 1 - entropy_a\,(i) \\ &= 1 + \frac{inweight_a\,(i)}{weight_a\,(i)} log_2^{\frac{inweight_a(i)}{weight_a(i)}} \\ &\quad + \frac{outweight_a\,(i)}{weight_a\,(i)} log_2^{\frac{outweight_a(i)}{weight_a(i)}} \end{aligned} \tag{7}$$

where

$$weight_a\,(i) = \sum_{j\in in\text{-}neighbors^a\{i\}} w^a_{ji} + \sum_{j\in out-neighbors^a\{i\}} w^a_{ij}$$

In these equations, $inweight_a\,(i)$ is the sum of the weight of input links to the node i at the activity layer a. $weight_a\,(i)$ also equals the number of input and output $a$of the node i.

Spam users Spammer detection in social networks is an important and recently studied task. Recently many works have dealt with spam detection in social networks, especially Twitter [47, 48]. Detecting and collecting spam users without using the information of users' activities is not possible. This is because the difference between spammers and others is by how they act. Most researches in discovering spam in social networks used comment or tweet content for their work [49, 50]. As stated before, in this paper we use the number of different Facebook activities without considering the user-generated contents to detect and collect spam users. As stated in famous users section, the spam user or the user whose behavioral pattern is spam-like is the person with many outputs in different activity networks but few inputs. In other words, most spam users have many output activities in the social network but do not have many input activities. Accordingly, (8) shows how the level of being spam can be calculated based on the four activities. The more the calculated value of (8), the higher the spamming probability of the node will be.

$$Spam\,(i) = \frac{\sum_{a\in\{like,comment,post,share\}} \left[\max(outweight_a\,(i) - inweight_a\,(i)\,, 0)\right] * W_a * purity_a\,(i)}{\sum_{a\in\{like,comment,post,share\}} W_a} \tag{8}$$

$inweight_a\,(i)$, $outweight_a\,(i)$ and $purity_a\,(i)$ can be calculated as mentioned in section famous users It should be noticed that the weight of different activities are important for calculating spam or famous node. For example, spam users are interested in putting many *posts* on other users' profile. Therefore, the importance of *post* is very significant to the definition of the spam node. However, in this paper we used $W_a = 1$ for all activities.

Polite users Social ethics suggest that requests are usually to be answered. This concept can be generalized to users' interactions. In this regard, a user is considered polite if he or she reacts to their input activities suitably. The (9) elaborates how to measure politeness of a node.

$$Politeness\,(i) = \frac{\sum_{a\in\{like,comment,post,share\}} \sum_{j\in neigh_a(i)} W_a * (1 - purity_a\,(i,\,j))}{\sum_{a\in\{like,comment,post,share\}} W_a} \tag{9}$$

Where $purity_a\,(i,\,j)$ shows the purity of $i - j$ edge in a activity network. In many cases, the response to the activity is based on the same mechanism of the activity. For instance, the response to a *comment* is often done by a *comment*, although this is not always true. In many cases, response to *posts* is done by *comments*, *likes*, or occasionally *shares*; or *comments* are responded to by *likes*. Therefore, the considered equation is not reliable. To solve this problem, the mixed activity network may be employed as follows:

$$Politeness\,(i) = \sum_{j\in neigh_{mixed}(i)} (1 - purity_{mixed}\,(i,\,j)) \tag{10}$$

It should be noted that we have normalized the calculated values before presenting them.

3.5 Evaluation measures

There are different centrality measures to calculate users' importance in complex networks. Some of them are degree, closeness, betweenness, eigenvector, PageRank, hub and authority [51]. According to Table 2, since the activity graph is intrinsically asymmetric, directed centrality measures are better to be used for measuring the importance of sampled nodes and the comparison of the presented algorithm with others. Our recommendation is PageRank. Because it is directed and very common. In addition, the node's PageRank is measured according to in-degrees rather than out-degrees. Therefore, especially for activity networks, the result of applying PageRank is more real. On the other hand, it is better to consider the information of different activities simultaneously for measuring importance of nodes. Since specifications of different activity networks are different, it is recommended that this network be modeled as multilayer. In addition, the evaluation method must be application-based because our method is about an application-based biased sampling. Therefore, we also need a PageRank-based method applicable over the multilayer graph, presenting importance of nodes based on the target application. We presented the pseudo-code of our method in algorithm 4.With a simple and effective approach, this algorithm can measure users' importance for different applications. It only requires that we measure the weight of different layers of the network for the considered applications and give them to the algorithm as input.

A-BMLPR (Application-based Multilayer PageRank) takes the four activity graphs of *like*, *comment*, *post*, and

Table 3 Average of different evaluation measures in activity network

Method	Mean
FameGini	0.402
FameEntropy	0.352
SpamGini	0.567
SpamEntropy	0.617
PoliteGini	0.012
PoliteEntropy	0.013

share as well as their importance. Vector p is filled according to the importance of different activities. The number of input graphs which in fact are different layers of the multilayer network may easily change. In addition, "initial" shows the initial node and "DF" shows the damping factor which is between 0 and 1. Lines 13 and 14 select activities like, comment, post, and share according to their importance, which is stored in vector p. Then the new node is selected based on the weight of the links outgoing from the current node. Finally, the influence of nodes is measured between 0 and 1 according to the number of nodes' visits based on the two-stage application-based random walk.

For spam and fame applications, we can use (8) and (5) for edge weighting policy in multilayer PageRank. Therefore, we can gain nodes' importance based on spam and fame applications using multilayer PageRank. On the other hand, to evaluate the presented method for applications fame, spam, and politeness, we can use the definitions stated in Section 3.4.2. Therefore, we will have two evaluation measures for each application based on Entropy and Gini Coefficient. Equations (5), (8), and (10) are respectively used for evaluation of fame, spam, and politeness. Table 3 presents the average of these measures for different applications. We use these values for examining the amount of the bias of sampling methods. We also normalized these numbers according to the min and max values between nodes.

4 Biased sampling from multilayer activity network

The presented method aims at sampling the activity graph in a way to collect suitable nodes based on the considered application. It acts in two stages: learning transition probability, and biased sampling of the activity network. Using the learning automata, we initially learn the probability of transitioning from each node to its neighbors. These probabilities are learnt so that the sampling agent would move on the suitable path. Secondly, we sample the graph based on the learnt probabilities of the previous step.

Algorithm 4 Application-based multilayer PageRank (A-BMLPR)

Input: gLike,gComment,gPost,gShare,initial, maxIter,DF,vnum, activity weights
1. p=[likeWeight, commentWeight, postWeight, shareWeight]
2. for i=1 to vnum do
3. Visits[i] ← 1
4. end
5. v ← initial node
6. i ← 1
7. while(not converge and i<maxIter) do
8. rand ← a number uniformly at random between 0 and 1
9. if(rand>DF) then
10. W ← select a uniformly random node
11. else
12. begin
13. while(degree^{Act}(v)=0) do
14. Act ← select activity i with probability p[i]
15. N ← neighbors^{Act}(v)
16. foreach(vertex $n \in N$)
17. prob[v]← $w_{vn}/(\sum_{n\in N} w_{vn})$
18. W ← select a vertex from N with probabilities of prob
19. end
20. V ← W
21. $Visits$[v] ← $Visits$[v]+1
22. i ← i+1
23. end
24. for all $v \in V$ do
25. PageRank[v] ← Visits[v]/i
26. end

Normal PageRank:
LikeWeight=1, CommentWeight=1, PostWeight=1, ShareWeight=1
Trust application:
LikeWeight=9, CommentWeight=7, PostWeight=23, ShareWeight=11
Time application:
LikeWeight=1, CommentWeight=4, PostWeight=4, ShareWeight=10

4.1 Learning transition probabilities phase

Learning transition probabilities is the core of biased sampling from the activity graph. At this stage, the probabilities

are updated so that the sampling agent would be guided to influential nodes. To this end, we used learning automata. On the monolayer mode, we placed a learning automata on each node of the considered activity network. Then we defined the probability of transition from the node v to its neighbors as $\frac{1}{k_v}$ where k_v is the degree of v. In case the network is weighted, however, the probability of transition from v to z equals $\frac{W_{vz}}{\sum_{k \in neighbors(v)} W_{vk}}$. Then by performing a random walk over the network, we update transition probabilities. In this regard, each automata learns its transition probabilities over time. To learn the automata, we start from an initial node V_0 which is selected uniformly at random. Regardless of the current node at each stage, we move from one node to another of its neighbors using a traversal strategy and correct the transition probabilities of the current learning automata. This strategy can be based on a simple random walk or the probabilities learnt by learning automatons. The automata on the current node can take actions as many as the number of its output links. Each action is corresponding to the selection of one of the output links of the current node. Having chosen the output links, we reach the node at the end of the edge, i.e. a new learning automata. If the new node is better than the previous node in terms of the application-based measure (Random Environment), we reward the link or chosen action, otherwise, we penalize it or do not change the probabilities (Reinforcement Signal). Reward and penalty are calculated based on (3) and (4).

The presented learning scheme is applicable to both monolayer and multilayer graphs. Algorithm 5 shows how to use learning automata to learn the probabilities of transition in the monolayer network. It should be noted that we used simple PageRank for evaluating biased sampling of monolayer networks.

At each step of the algorithm, v is the current node and we intend to go to one of its out-neighbors. Vinit is the initial node; maxIter is the maximum number of iterations; and goodness is a function which shows the node's importance in terms of the considered application including fame, spam, and politeness according to equations of Section 3.4.2. The winSize is the size of the window which we examine falling into the trap in it. Line 9 is related to cases when v does not have an out-neighbor. In such a case, the random walk has reached a dead end and has to jump to the best node visited (vjump). Herein, the good node is the one with many out-neighbors since there are more choices for moving from this node to its neighbors. Line 11 evaluates the case when after vjump, not many nodes are available. In such a case, returning to vjump is probably repeated many times and this node is visited many times. If the mean of the distance vector from the node vjump is less than the number of times selecting the node vjump, we should replace the current vjump with another node. In line 12, this node has been chosen absolutely by random out of all the nodes having out-neighbors.

Algorithm 5 Learning transition probabilities in monolayer graph

```
1   Input: graph, goodness, maxIter, vinit, rewardRate,
    penaltyRate,winSize,winSmall
2   Output: graph
3
4   v ←vinit
5   vjump ←v
6   i ←2
7   while(i<maxIter)
8   {
9   ifout-degree of node v is 0
10   {
11   if vjump is repeated more times
12     vjump ← select a uniformly random node with out-
       neighbors>0
13   v ←vjump
14   }
15   if number of unique nodes in winSize last selected
     nodes is less than winSmall
16     v ←new randomly selected node
17   N ←out-neighbors of node v
18   prob ←probability of each out-edge from node v
19   w ←select a new node from n using probabilities prob
20   i ←i+1
21   if goodness[w] >goodness[v]
22    prob ←reward(prob, index of w in N ,a=rewardRate)
23   else
24     prob ←penalty(prob, index of w in N,b = penal-
   tyRate)}
25   Update prob in graph
26   if (out-neighbors of node w>out-neighbors of vjump)
27   {
28     vjump ←w
29   }
30   v ←w
31  }
```

Line 15 evaluates the conditions of falling into trap. Since the network is directed, there may be highly clustered sub-graphs or small sub-graphs whose nodes have out-links only to each other, but not to rest. In such a case, entering those sub-graphs, the random walk gets blocked. Therefore, the winSize number of recently visited nodes is kept. If out of these nodes, the winSmall number of nodes are not unique,

it means we probably have fallen into a trap and should jump to a random node. Line 17 puts out-neighbors of v into subset N. Among these neighbors, every neighbor has a visiting probability located in prob vector. These probabilities are initially calculated as $\frac{weight_{vw}}{\sum_{k \in neighbors(v)} weight_{vk}}$ and are updated over time. Then based on these probabilities, the node w is chosen. If w is better than v in terms of the considered application, the chosen link should be rewarded to increase its visiting probability. Otherwise, that link should be penalized.

Algorithm 6 shows the learning automata algorithm for the multilayer network. Herein, one layer is defined for each activity. Since the number of collected activities is 4 (*like*, *comment*, *post* and *share*) the activity graph will be 4-layered. The vector ActWeight in the algorithm shows importance of different activities. For calculating the transition probabilities in the multilayer network, we initially put four learning automatons over each node each of which learns the probability of transition at each layer of the activity network. In lines 10 and 11 of the algorithm, the probability of choosing the activity is measured by the weight of activities. Lines 12 to 22 try to choose the activity. Therein, it is tried to initially choose the activity according to its importance. The current node at that layer have some out-neighbors. If the current node has no out-neighbors, the walker is in dead-end. The strategy of traversal and resolving the problems of walk in the directed network such as dead end, and trap are the same as in the previous algorithm.

As we noted previously, we use multilayer activity network because it models the real world better. Moreover, if the walker reached a dead end it can also use other layers to continue its way.

In this algorithm, the walk occurs at the layer the activity of which has been chosen, but the learning process of automata will happen at all layers of the multilayer network. In line 31, the node w is chosen based on the learnt probabilities at the chosen layer. It is noticeable that in a multilayer graph, each link at any of the graph layers, has a particular weight. Each node may also have several input links. The sum of the input links to w at the layer *act* equals InWeight of w in graph^{Act}. Lines 35 to 38 perform the transition probability update. If the Inweight of w in graph^{Act} is more than that of v, the link E^{Act}(v,w), if existent, is rewarded; otherwise it will be penalized. In addition, the automaton's learning is possible in all three modes L_{RI}, L_{RP}, and $L_{R\varepsilon P}$ and would produce different outcomes

Algorithm 6 trains different automatons for different activities since one node's value may be different in terms of different activities. This algorithm can be used for biased sampling according to different applications such as trust, closeness, and time. However, we require information of all layers of the multilayer network to sample according to fame, spam, and politeness applications. Therefore, one node's goodness in these terms is the same at all layers. So placing one automaton on each node is adequate for these applications. Learning transition probabilities for these three applications is presented in Algorithm 7. In this algorithm, if the new node is better than the old one in terms of the considered application, presented by formulas (5), (8) and (10), the transition to the new node is rewarded and the probability of transition to this node is increased. Otherwise, this probability is decreased or not changed. It is noticeable that another mode of learning is also possible: that only one of the layers be rewarded or penalized. This mode is single layer learning. But the one considered in Algorithm 7 is multilayer learning.

Algorithm 5 uses a simple monolayer network, which is simple in implementation. It can be used to bias toward nodes with high amount of PageRank. Although Algorithm 5 is usable in fame, spam, and polite applications, algorithm 7 uses more information and is more accurate. In addition, the application of Algorithm 6 is not similar to Algorithms 5 and 7. This algorithm is useful for applications trust, closeness, and time.

4.2 Sampling phase

After learning the transition probabilities using learning automata, we applied a random walk process on the activity network using the learnt probabilities. The random walk at each stage chooses one of the out-links of the current node according to the probabilities. The new chosen node is put in the list of sampled nodes and replaces the current node of the walk. It is noticeable that these probabilities are set such that the walk be guided toward to the considered target.

It should be noticed that all the algorithms presented in this section can easily be modified to a one-stage format. This means that sampling should be done while learning transition probabilities. The result of the one-stage biased sampling is almost better than most of the previous sampling methods. However, since the two-stage sampling produced the best results, we elaborated the two-stage biased sampling and reported its results.

5 Experimental results

Since most of our applications have been defined on the multilayer network, the objective of sampling the simple

Algorithm 6 Learning transition probabilities in multilayer graph

```
Input:graph, ActWeight, maxIter, vinit, rewardRate, penaltyRate, winSize, winSmall
Output: graph

v ← vinit
vjump ← v
i ← 2
while(i<maxIter)
{
 p ← ActWeight
 normalize p with sum of activity weights
 Selected ← False
 for j from 1 to number of activities
 {
  Act ← randomly select an activity with probability p.
  b ← degree(graph[[act]],v)
  if output_Degree^Act(v)==0
    p[act] ← 0
    normalize p with some of remaining activities weights
  else
    Selected ← True
  }
  if Selected=False
    if vjump is repeated more times
      vjump ← new randomly selected node, with Output degree >0 in each layer
    v ← vjump
  if number of unique nodes in winSize last selected nodes is less than winSmall
    v← new randomly selected node, with Output degree >0 in each Activity
  N← output-neighbors of node v in graph^Act
  prob← probability of each output-edge^Actfrom node v
  w← select a new node from N with probabilities prob
  for each Act
    if w is Output_neighbor of v in graph^Act
      prob ← probability of each output-edge^Act from node v
      if InWeight of w in graph^Act>InWeight of v in graph^Act
        prob ←reward(prob, Index of w in prob, a =rewardRate)
      else
        prob ←penalty(prob, Index of w in prob, b=penaltyRate)
  if (InWeightSum of w in all Layers >InWeightsSum of vjump)
      vjump ←w
  v ←w
  i ←i+1
  }
```

activity network is to sample nodes with high PageRank. The considered network of the experiments on the monolayer network is the *like* network. In this section, we initially compare the bias of the presented method with others including simple random walk, forest fire, degree sampling, Metropolis-Hastings random walk, and BFS. For this, using the sampling algorithm, we collected 1000 nodes of the network and compared the results. Random-walk-based

Algorithm 7 Learning transition probabilities in multilayer graph for fame, being spam and politeness applications

```
Input: graph, goodness, ActW, maxIter, vinit, rewardRate, penaltyRate, winSize, winSmall
Output: graph

v←vinit
i ← 2
while (i<maxIter)
{
 p ← ActWweight
 normalize p with sum of activity weights
 Selected ← False
 for j from 1 to number of activities)
 {
  Act ← randomly select an activity with probability p.
  b ← degree(graph[[act]],v,mod=mDirect)==0
  if output_Degree Act (node v)==0
   p[act] ← 0
   normalize p with some of remaining activities weights
  else
   Selected ← True
 }
 if Selected=False
  if vjump is repeated more times
   vjump ← new randomly selected node, with Output degree >0 in each layer
  v ← vjump
 if number of unique nodes in winSize last selected nodes is less than winSmall
  v ← new randomly selected node, with Output degree >0 in each Activity
 N ← output-neighbors of node v in graph^Act
 prob ← probability of each output-edge^Act from node v
 w ← select a new node from N with probabilities prob.
 if goodness[w]>goodness[v]
  for each Act
   if w is Output_neighbor of v in graph^Act)
    prob ← probability of each output-edge^Act from node v
    prob ← reward(prob, Index of w in prob, a = rewardRate)
 else
 for each Act
  if w is Output_neighbor of v in graph^Act)
   prob ← probability of each output-edge Act from node v
   prob ← penalty(prob, Index of w in prob, b = penaltyRate)

 if InWeightSum of w in all Layers >InWeightsSum of vjump
  vjump ← w
 v ← w
 i ← i+1
}
```

algorithms have been evaluated with and without replacement. In addition, we compare biased sampling in terms of different applications trust, closeness, and time. Finally, the samples gained by the presented sampling method are compared with common sampling methods in terms of bias toward famous, spam, and polite nodes.

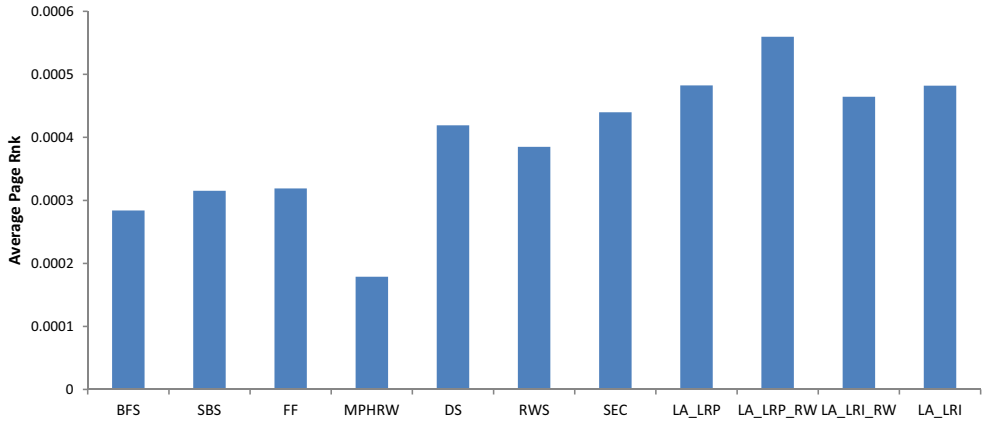


Fig. 5 Average PageRank of 1000 sampled nodes with different sampling algorithms from Single Layer *like* network without replacement. BFS: Breadth First Search, SBS: Snowball Sampling, FF: Forest Fire, MPHRW: Metropolis-Hastings Random Walk, DS: Degree Sampling, RWS: Random Walk Sampling, SEC: Select Edge Count, LA_LRP: Learning Automata (Linear Reward Penalty), LA_LRI: Learning Automata (Linear Reward Inaction)

Fig. 6 Average PageRank of 1000 sampled nodes using different sampling algorithms from Single Layer *like* network with replacement

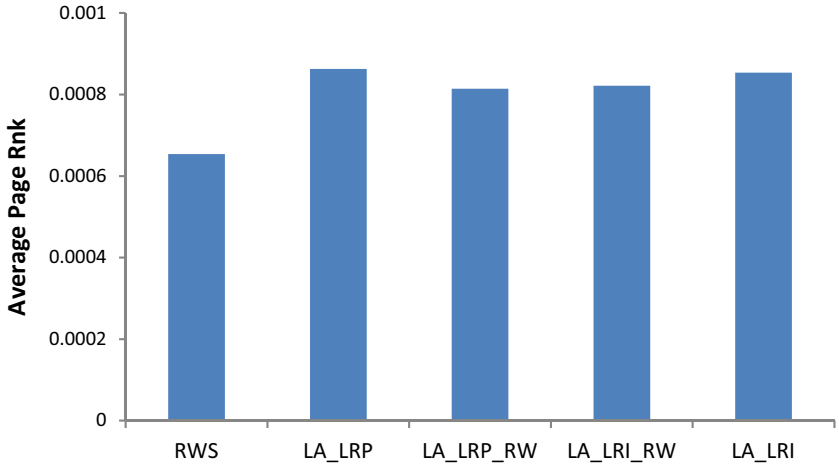
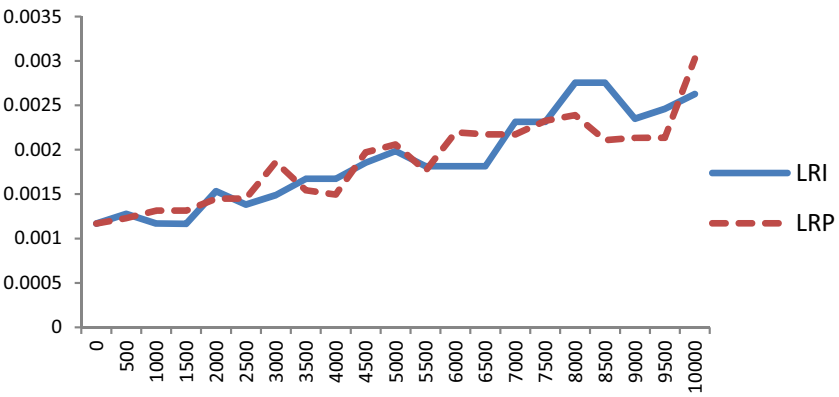


Fig. 7 Change of average A-BMLPR for different iterations of automata-based methods in trust application. In all of the subsequent figures vertical axes are average application-based PageRank



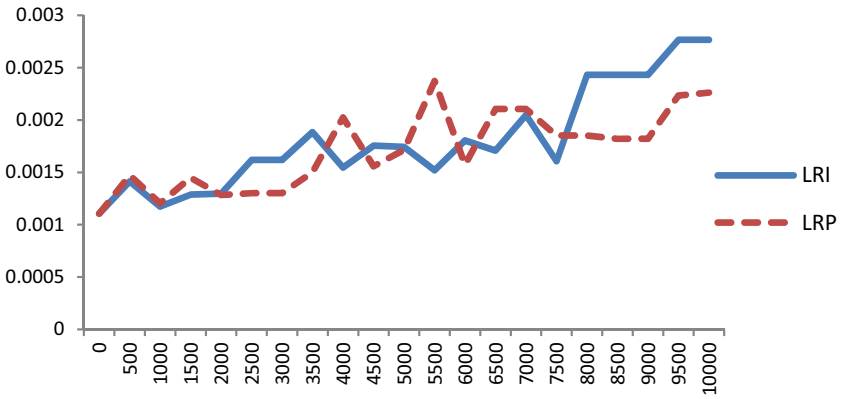


Fig. 8 Change of average A-BMLPR for different iterations of automata-based methods in closeness application

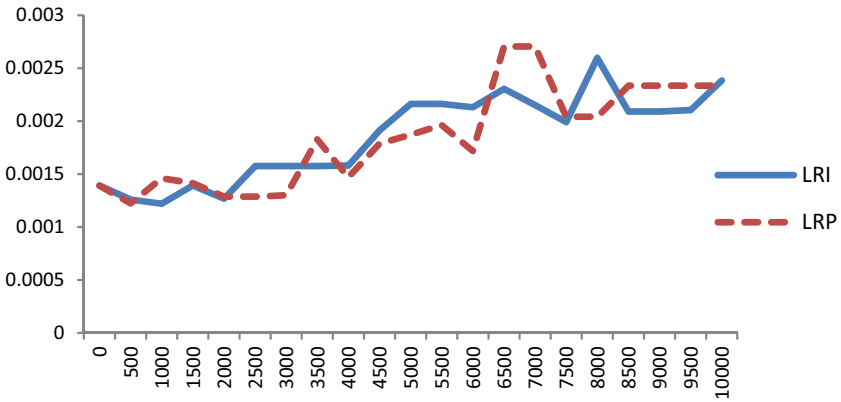


Fig. 9 Change of average A-BMLPR for different iterations of automata-based methods in time application

Table 4 Comparison of presented methods with the existent methods in terms of bias toward famous, spam, and polite nodes

	LRI-MLL	LRI-MLL-RW	LRP-MLL	LRP-MLL-RW	LRI-SLL	LRI-SLL-RW	LRP-SLL	LRP-SLL-RW	ML-RW	Uniform
FameGini	0.473	0.465	0.471	0.464	0.458	0.465	0.459	0.463	0.424	0.403
FameEntropy	0.423	0.41	0.421	0.406	0.402	0.4	0.397	0.402	0.364	0.372
SpamGini	0.587	0.59	0.59	0.589	0.589	0.589	0.59	0.588	0.536	0.566
SpamEntropy	0.642	0.642	0.641	0.64	0.639	0.641	0.641	0.639	0.594	0.618
PoliteGini	0.083	0.098	0.09	0.086	0.095	0.095	0.087	0.096	0.066	0.012
PoliteEntropy	0.084	0.094	0.092	0.101	0.103	0.099	0.102	0.096	0.065	0.012

Table 5 Comparison of presented methods with the existent methods in terms of bias toward famous, spam, and polite nodes by measuring application-based PageRank multiplied by 10000

	LRI-MLL	LRI-MLL-RW	LRP-MLL	LRP-MLL-RW	LRI-SLL	LRI-SLL-RW	LRP-SLL	LRP-SLL-RW	ML-RW	Uniform
FameGini	18.81	15.37	17.47	14.62	14.73	14.23	13.38	14.35	11.07	1.292
FameEntropy	20.08	14.80	17.89	14.68	13.91	14.22	13.48	13.8	10.38	1.263
SpamGini	11.18	11.34	10.25	10.82	10.75	10.34	11.65	12.2	5.78	1.219
SpamEntropy	11.34	10.99	10.95	12.41	11.20	9.891	11.26	10.87	5.42	1.244
PoliteGini	16.468	13.668	16.21	13.233	12.86	12.805	12.517	12.703	11.66	1.204
PoliteEntropy	16.057	14.089	17.49	13.937	13.28	12.669	13.326	13.292	10.89	1.247

5.1 Monolayer network

Figure 5 shows the mean PageRank of the collected nodes from the *like* network using different sampling methods. In this section, all experiments are without replacement. LA_LRP is the learning automata where the rate of reward and penalty is 0.01. In LA_LRP_RW method, rather than the learnt probabilities, the probabilities of weighted random walk are used for moving to the new node at the learning stage of the automata. The method LA_LRI is the learning automata where the penalty rate is zero and while walking, it uses the learnt probabilities in the learning phase of the automata. The method LA_LRI_RW is also similar to LA_LRI, but while moving at the learning stage, it does not use the learnt probabilities. In this experiment, the learning phase is carried out for 2000 iterations. Then the resulted graph is sampled 20 times and 1000 nodes are chosen. At the end, the mean PageRank of these nodes are measured. It is seen that automata-based methods have more bias toward high-PageRank nodes versus all other methods, so they have performed better.

Figure 6 shows the mean PageRank of the chosen nodes corresponding to different walking methods with replacement. Learning automata based methods have also performed better than the simple random walk method with replacement.

5.2 Multilayer network [trust, closeness, and time applications]

For learning probabilities and guiding the random walk toward special nodes for applications trust, closeness, and time, Algorithm 6 has been used. We used the presented application-based multilayer PageRank algorithm for evaluating the bias of sampling methods (Algorithm 4).

In this experiment, we initially calculated importance of nodes in terms of trust, closeness, and time. After that the learning phase was carried out with different iterations. Then based on the learnt probabilities, we sampled 1000 nodes and calculated the mean importance of those nodes in terms of trust. This sampling was done 20 times. Figure 7 shows that by increasing the number of learning iterations in the multilayer automata method, the bias toward highly trustful nodes has increased. The vertical axis represents the mean importance of collected users. It is seen that by increasing the number of learning stages of the automata,

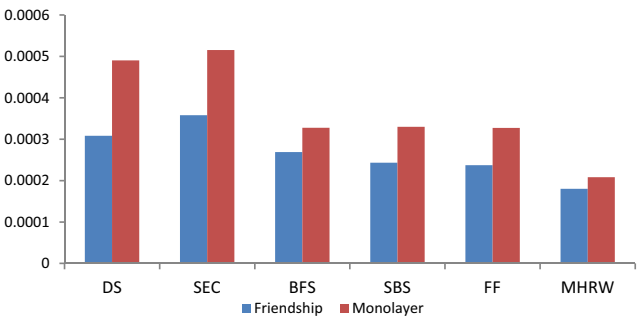


Fig. 10 Comparison of the bias of sampled nodes of friendship and activity network in closeness application

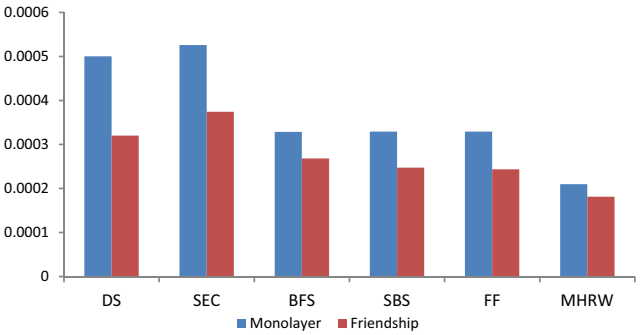


Fig. 11 Comparison of the bias of sampled nodes from friendship and monolayer activity network in trust application

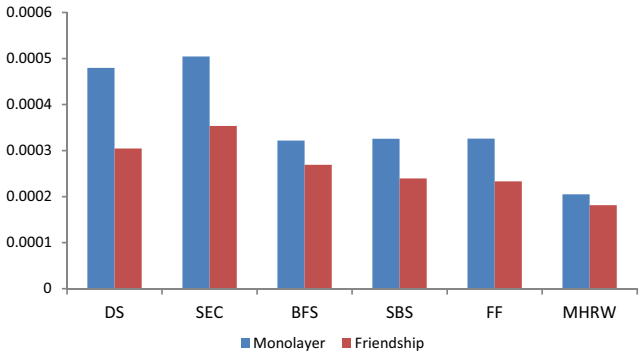


Fig. 12 Comparison of the bias of sampled nodes from friendship and monolayer activity network in time application

the probabilities are set in a way that the walking algorithm is more biased toward to the considered nodes and collects them. Figures 8 and 9 respectively represents the same experiment on applications closeness, and time.

5.3 Multilayer network [Fame, spam, and politeness applications]

In fame, spam, and politeness applications, we have used two measures for evaluating the bias of algorithms: evaluation measures which presented in (5), (8) and (10) equations and PageRank for the considered application. Table 4 shows the bias of the presented algorithms together with the existent algorithms which are usable in the multilayer network. As already mentioned, the reward and penalty can be either multi-aspect or mono-aspect, that is, whether to reward all activities or only the current activity. These two parameters have been named MLL (multilayer learning) and SLL (single-layer learning). In addition, the learning phase has been done in the two ways: with and without random walk. The results from algorithm ML-RW that uses random walks on the multilayer graph are also presented.

Higher difference between the presented values in Table 4 and the average values presented in Table 3 means higher bias toward the considered applications. As can be seen, random walk is a little biased toward famous and polite nodes, but its bias toward spams is negative. In addition, the presented methods are more biased than random walk in terms of fame, spam and politeness. On the fame mode, methods of MLL are more biased than SLL, whereas on the polite mode, the reverse is true. When the considered application is spam, the presented method is biased toward influential nodes, contrary to random walk, and collects more important nodes. It also can be seen that uniform sampling is unbiased in all cases.

Table 5 shows the same experiments of Table 4 by measuring application-based PageRank. It is seen that the presented methods are more biased toward nodes with higher PageRank in particular applications than other methods.

Table 5 shows the higher performance of the presented method than the results of multilayer random walk and uniform sampling.

5.4 Comparison of activity and friendship networks

Our experiments indicated that the proposed methods perform better than other existing algorithms. In this section we compare different networks, in terms of bias of sampling methods according to discussed applications. First, using each sampling method on different networks (if applicable), we sample 1000 nodes. Then, we average scores of sampled nodes in different discussed applications. Simple random walk algorithm displayed in Algorithm 1 is applicable on Monolayer directed and undirected networks. We also have discussed its Multilayer version. But sampling methods detailed in Section 2 were presented just for Monolayer networks. So Figs. 10, 11 and 12 compares this methods for Monolayer activity network and friendship network of the same nodes (users) in terms of Closeness, Trust and Time applications.

These three figures show using sampling methods on the activity network, samples more biased nodes under these three applications. Therefore, it is better to use activity network than friendship network for biased sampling application.

Table 6 Comparison of friendship and activity networks in terms of bias of different sampling methods in famous, spam, and polite applications multiplied by 10000

	FameEntropy		FameGini		PoliteEntropy		PoliteGini		SpamEntropy		SpamGini	
	activity	Friend	activity	Friend	activity	Friend	activity	Friend	activity	Friend	activity	Friend
DS	5.4749	3.7743	5.4912	3.794	5.4736	3.7486	5.4444	3.7158	5.2559	3.7139	5.2905	3.7071
SEC	5.6503	4.1355	5.6735	4.1485	5.6656	4.0899	5.6317	4.0974	5.243	3.9432	5.2824	3.9589
BFS	3.5927	2.8704	3.5801	2.8178	3.6096	2.8724	3.5505	2.8609	3.3734	2.7254	3.3927	2.7025
SBS	3.9129	3.3039	3.9167	3.3150	3.9355	3.3046	3.8915	3.3025	3.7594	3.2398	3.7624	3.2486
FF	4.0263	3.4409	4.0152	3.4442	4.0234	3.4363	3.9911	3.4338	3.8490	3.4216	3.8590	3.4315
MHRW	2.1891	1.7711	2.1802	1.7815	2.1953	1.7595	2.1802	1.7621	2.1464	1.7805	2.1659	1.7903

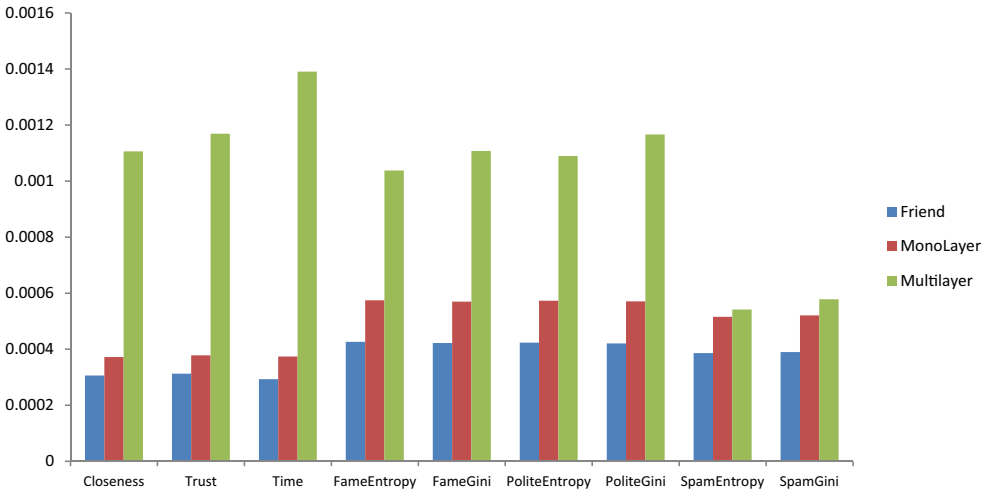


Fig. 13 Comparison of the different biases of Random Walk on 3 networks

Table 6 shows the same experiment on other applications fame, politeness and spam for these 2 networks. It also shows that the activity network is more suitable than friendship network for biased sampling in these aspects.

Random walk algorithm is applicable to all of our networks. So Algorithm 1 is applicable to Friendship and monolayer activity network. We use special multilayer random walk for each of Closeness, Trust and Time and set the probability of choosing each activity according to their importance in the considered applications. Figure 13 displays the bias of random walk Algorithm on 3 different networks. This shows the multilayer activity network is better than others in biased sampling.

6 Conclusion

In this paper, a biased sampling method was presented. The presented sampler is guided toward nodes which are important in some considered applications by using the information of different activities of Facebook. For this, a two-stage algorithm was proposed. At the first stage the transition probabilities from each node to another are learnt using learning automata and at the second stage, it samples the nodes of the considered network by using the learnt probabilities. This algorithm was examined on the Facebook activity network in two modes: monolayer and multilayer. For measuring the bias of the presented method, the A-BMLPR method which examines importance of nodes for different applications was used. The applications used included fame, spam, and politeness in addition to trust, closeness, and time. The results show that the proposed method works better than popular sampling methods in terms of all the defined applications. Therefore, this method is more efficient in biased sampling. For related future works, evaluation of the presented method by other applications or social networks can be focused on. In addition, using other learning methods for learning transition probabilities can be considered a developmental study to the present work. Another future work is using methods such as MLP and SVM to learn the transition probabilities from both activity and profile information to improve the biased sampling.

References

1. Viswanath B, Mislove A, Cha M, Gummadi KP (2009) On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM workshop on online social networks, pp 37–42
2. Wilson C, Boe B, Sala A, Puttaswamy KP, Zhao BY (2009) User interactions in social networks and their implications. In: Proceedings of the 4th ACM european conference on computer systems, pp 205–218
3. Wilson C, Sala A, Puttaswamy KP, Zhao BY (2012) Beyond social graphs: User interactions in online social networks and their implications. ACM Trans Web (TWEB) 6:17
4. Chun H, Kwak H, Eom Y-H, Ahn Y-Y, Moon S, Jeong H (2008) Comparison of online social relations in volume vs interaction: a case study of cyworld. In: Proceedings of the 8th ACM SIGCOMM conference on internet measurement, pp 57–70
5. Khadangi E, Bagheri A (2015) Analyzing structural and topological properties of various facebook activity networks journal of informetrics
6. Ahmed NK, Neville J, Kompella R (2012) Space-efficient sampling from social activity streams. In: Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: algorithms, Systems, Programming Models and Applications, pp 53–60
7. Ahmed NK, Berchmans F, Neville J, Kompella R (2010) Time-based sampling of social network activity graphs. In: Proceedings of the eighth workshop on mining and learning with graphs, pp 1–9
8. Ahmed NK, Neville J, Kompella R (2014) Network sampling: From static to streaming graphs. ACM Trans Knowl Disc Data (TKDD) 8:7
9. Gjoka M, Kurant M, Butts CT, Markopoulou A (2010) Walking in facebook: a case study of unbiased sampling of osns. In: INFOCOM, 2010 Proceedings IEEE, pp 1–9

10. Leskovec J, Faloutsos C (2006) Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 631–636
11. Even S (2011) Graph algorithms. Cambridge University Press, Cambridge
12. Kolaczyk E (2009) Statistical Analysis of Network Data, volume 69 of Springer Series in Statistics ed: Springer New York
13. Yoon S, Lee S, Yook S-H, Kim Y (2007) Statistical properties of sampled networks by random walks. Phys Rev E 75:046114
14. Lee C.-H., Xu X, Eun DY (2012) Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. In: ACM SIGMETRICS Performance evaluation review, pp 319–330
15. Kurant M, Gjoka M, Butts CT, Markopoulou A (2011) Walking on a graph with a magnifying glass. In: Proceedings of ACM SIGMETRICS
16. Salganik MJ, Heckathorn DD (2004) Sampling and estimation in hidden populations using respondent-driven sampling. Sociol Methodol 34:193–240
17. Rezvanian A, Rahmati M, Meybodi MR (2014) Sampling from complex networks using distributed learning automata. Physica A: Stat Mech Appl 396:224–234
18. Torkestani JA (2012) An adaptive focused web crawling algorithm based on learning automata. Appl Intell 37:586–601
19. Gile KJ, Handcock MS (2010) Respondent-driven sampling: An assessment of current methodology. Sociol Methodol 40:285–327
20. Salehi M, Rabiee HR, Nabavi N, Pooya S (2011) Characterizing twitter with respondent-driven sampling. In: Dependable, autonomic and secure computing (DASC), 2011 IEEE ninth international conference on, pp 1211–1217
21. Gjoka M, Kurant M, Butts CT, Markopoulou A (2009) Unbiased sampling of facebook. preprint arXiv: 0906.0060
22. Ribeiro B, Wang P, Murai F, Towsley D (2012) Sampling directed graphs with random walks. In: INFOCOM, 2012 Proceedings IEEE, pp 1692–1700
23. Salehi M, Rabiee HR (2013) A measurement framework for directed networks. IEEE J Sel Areas Commun 31:1007–1016
24. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media? In: Proceedings of the 19th international conference on world wide web, pp 591–600
25. Ahn Y.-Y., Han S, Kwak H, Moon S, Jeong H (2007) Analysis of topological characteristics of huge online social networking services. In: Proceedings of the 16th international conference on world wide web, pp 835-844
26. Mislove A, Koppula HS, Gummadi KP, Druschel P, Bhattacharjee B (2008) Growth of the flickr social network. In: Proceedings of the first workshop on online social networks, pp 25–30
27. Kurant M, Markopoulou A, Thiran P (2011) Towards unbiased BFS sampling. IEEE J Sel Areas Commun 29:1799–1809
28. Maiya AS, Berger-Wolf TY (2010) Sampling community structure. In: Proceedings of the 19th international conference on world wide web, pp 701–710
29. Salehi M, Rabiee HR, Rajabi A (2012) Sampling from complex networks with high community structures. Chaos: An Interdisciplinary J Nonlinear Sci 22:023126
30. Lee SH, Kim P.-J., Jeong H (2006) Statistical properties of sampled networks. Phys Rev E 73:016102
31. Blagus N, Šubelj L, Weiss G, Bajec M (2015) Sampling promotes community structure in social and information networks. Physica A: Stat Mech Appl 432:206–215
32. Gjoka M, Kurant M, Butts CT, Markopoulou A (2011) Practical recommendations on crawling online social networks. IEEE J Sel Areas Commun 29:1872–1892
33. Gjoka M, Butts CT, Kurant M, Markopoulou A (2011) Multigraph sampling of online social networks. IEEE J Sel Areas Commun 29:1893–1905
34. Corlette D, Shipman IIIF. (2009) Capturing on-line social network link dynamics using event-driven sampling. In: Computational science and engineering, 2009. CSE'09. International conference on, pp 284–291
35. Hughes AL, Palen L (2009) Twitter adoption and use in mass convergence and emergency events. Int J Emerg Manag 6:248–260
36. Maiya AS, Berger-Wolf TY (2011) Benefits of bias: Towards better characterization of network sampling. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, pp 105–113
37. Zhang C, Xie J, Xie J, Wu M, Huang Y, Huang X (2013) Detecting the core network of microblog using snowball sampling. In: Wireless personal multimedia communications (WPMC), 2013 16th international symposium on, pp 1–5
38. Wang H, Lu J (2013) Detect inflated follower numbers in OSN using star sampling. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, pp 127–133
39. Backstrom L, Leskovec J (2011) Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the fourth ACM international conference on web search and data mining, pp 635–644
40. Horvitz DG, Thompson DJ (1952) A generalization of sampling without replacement from a finite universe. J Am Stat Assoc 47:663–685
41. Kurant M, Markopoulou A, Thiran P (2010) On the bias of bfs (breadth first search). In: Teletraffic congress (ITC), 2010 22nd international, pp 1–8
42. Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: Densification laws, shrinking diameters and possible explanations. In: Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining, pp 177–187
43. Adamic LA, Lukose RM, Puniyani AR, Huberman BA (2001) Search in power-law networks, vol 64
44. Mehr SM, Taran M, Hashemi AB, Meybodi M (2011) A new recommendation algorithm using distributed learning automata and graph partitioning. In: Hybrid intelligent systems (HIS), 2011 11th international conference on, pp 351–357
45. Jahanshahi M, Dehghan M, Meybodi MR (2013) LAMR: Learning automata based multicast routing protocol for multi-channel multi-radio wireless mesh networks. Appl Intell 38:58–77
46. Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques: Elsevier
47. Hu X, Tang J, Zhang Y, Liu H (2013) Social spammer detection in microblogging. In: Proceedings of the twenty-third international joint conference on artificial intelligence, pp 2633–2639
48. Miller Z, Dickinson B, Deitrick W, Hu W, Wang AH (2014) Twitter spammer detection using data stream clustering. Inf Sci 260:64–73
49. Sureka A (2011). preprint arXiv: 1103.5044
50. Zhu Y, Wang X, Zhong E, Liu NN, Li H, Yang Q (2012) Discovering spammers in social networks. In: AAAI
51. Newman M (2010) Networks: An introduction: OUP Oxford

Ehsan Khadangi (khadangi@gmail.com, www.khadangi.ir) received his MSc in Information Technology from Amirkabir University of Technology in 2009. He is currently a 5th year PhD student in Computer Engineering at Amirkabir University of Technology and simultaneously a lecturer at Qom University. His current research interests mainly focus on Social Media Marketing, Social Network Analysis, Link Prediction, Data Mining on Social Networks, Viral Marketing, Recommender Systems, and etc.

Alireza Bagheri received his B.S. and M.S. degrees in computer engineering from Sharif University of Technology (SUT) at Tehran. He received his Ph.D. degree in computer science from Amirkabir University of Technology (AUT) at Tehran. Currently he is an assistant professor in the computer engineering and IT department at Amirkabir University of Technology (AUT) at Tehran. His research interests include computational geometry, graph drawing and graph algorithms.

Amin Shahmohammdi received his BSc degree in Computer Engineering (software branch) from the Islamic Azad University, Kaleibar Branch, Iran in 2013. He also received his MSc degree in Computer Engineering from the Pooyesh Institute of Higher Education, Qom, Iran in 2016. His current research interests include Data Mining, Machine Learning, Social Network Analysis, Distributed Systems, Optimization, and Parallel algorithms. His programming experiences are in R, and C++ language.