INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2017;**30**:e3091.
Published online 2 December 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.3091

A new learning automata-based sampling algorithm for social networks

Alireza Rezvanian and Mohammad Reza Meybodi*,†

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

SUMMARY

Recently, studying social networks plays a significant role in many applications of social network analysis, from the studying the characterization of network to that of financial applications. Due to the large data and privacy issues of social network services, there is only a limited local access to the whole network data in a reasonable amount of time. Therefore, network sampling arises to studying the characterization of real networks such as communication, technological, information, and social networks. In this paper, a sampling algorithm for complex social networks that is based on a new version of distributed learning automata (DLA) reported recently called extended DLA (*e*DLA) is proposed. For evaluation purpose, the *e*DLA-based sampling algorithm has been tested on several test networks and the obtained experimental results are compared with the results obtained for a number of well-known sampling algorithms in terms of relative error and Kolmogorov–Smirnov test. It is shown that *e*DLA-based sampling algorithm outperforms the existing sampling algorithms. Experimental results further show that the *e*DLA-based sampling algorithm in comparison with the DLA-based sampling algorithm has a 26.93% improvement for the average of Kolmogorov–Smirnov value for degree distribution taken over all test networks.

Received 1 January 2015; Revised 10 October 2015; Accepted 17 October 2015

KEY WORDS: complex social networks; social network analysis; network sampling; learning automata; extended distributed learning automata

1. INTRODUCTION

Todays, online social networks (OSN) provide a suitable platform for sharing and broadcasting a variety of information and multimedia by users. Due to the various applications of OSN among researchers from the study the characterization of the network to that of financial applications, studying and analyzing of the social networks arise new challenging research topics in the literature [1–4]. OSNs have attracted widespread of attention in the existing network researches. Some researchers tried to analyze the real network such as social network by observing structures and properties of network and then generate a model to reveal their properties. For example, many of real networks reflect universal properties such as small-world property, which indicates that most of real networks have a short average distance between any pair of nodes in real-world networks [5]. Another famous phenomenon for real-world networks is scale-free property with a power law degree distribution, which implies that several nodes on these networks as known hubs have a very large number of edges in comparison with other nodes abundantly [6]. Furthermore, most of real-world network such as social networks is large, dynamic, and complex, and also due to privacy settings of OSNs, some part of OSNs is not fully available [7]. Thus, in the most of the time, it is impossible to have a direct access on these networks to study and analyze their characterization and properties.

*Correspondence to: Mohammad Reza Meybodi, Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran.
†E-mail: mmeybodi@aut.ac.ir

Therefore, researchers try to study and analyze real networks such as communication, technological, information and OSNs via some metrics (i.e., centrality measures) or some techniques (i.e., network sampling methods) to estimate the characterization of these networks. Besides, some hidden important properties of networks (e.g., user age distribution, user activities, user connectivity, and many more in OSN) are discovered by studying on small sampled data. It is noted that sometimes sampling methods applied on the networks subject to preserve vital properties of the networks to reach a suitable visualization.

A social network can be represented as a graph with a set of nodes such as users and a certain type of relationship between users as edges of graph. Let $G = \langle V, E \rangle$ be a given social network graph, where V is the set of nodes and E is the set of edges. Sampled network $G_s = \langle V_s, E_s \rangle$ is the induced subgraph of G by sampling rate ϕ, where $V_s \subseteq V$, $E_s \subseteq E$ and $0 < \phi < 1$. It is noted that the main goal of sampling algorithms is to achieve a smaller subgraph similar to the original graph G. Sampling algorithms [8–10] play an important role in preprocessing, characterizing, studying, and analyzing the real networks. Network sampling can be applied for characterization of a small portion of a network subject to preserve main properties of the original network. For example, due to privacy restrictions of some OSN services, accessibility to the whole network information is not practical, therefore, the network is studied and characterized by estimated small samples from an important part of the original network. Taking into account large size, complexity, and dynamic properties of such real networks, most classical methods are not applicable. Moreover, some OSNs have limited accessibility to the entire networks simultaneously due to privacy or computational issues [9–12]. In addition to the aforementioned notes about analysis of these networks, by sampled network with lower size can improve the performance of information processing in a limited amount of time with lower computational time and it will be reduce cost (i.e., process, storage, bandwidth, and money), instead of the original network with larger size. Hence, there are some particular challenges and complexities for real network sampling, which can be still argued as new research domains [13].

In this paper, a sampling algorithm for social networks which is based on a new version of distributed learning automata (DLA) [14] called extended DLA (*e*DLA) [15] is proposed. In the proposed sampling algorithm, a set of learning automata that forms an *e*DLA isomorphic to the input network cooperate with each other in order to take appropriate samples from the network. The algorithm iteratively tries to identify promising nodes or edges by traversing promising parts of the network and then forms the sampled network. In order to investigate the performance of the proposed sampling algorithm, the proposed sampling algorithm is investigated on several real and synthetic networks, and the results are compared with the results obtained for a number of well-known sampling algorithms such as random node sampling (RNS) [13], random edge sampling (RES) [13], random walk sampling (RWS) [16], Metropolis–Hastings random walk sampling (MHRW) [17], and DLA-based sampling [7]. Experimental results show that the *e*DLA-based sampling algorithm outperforms existing algorithms with respect to relative error and Kolmogorov–Smirnov test.

The rest of this paper is organized as follows. Section 2 as background and preliminaries introduces related work on sampling techniques, learning automata (LA), variable action set learning automata, DLA, *e*DLA, and distributed learning automata-based sampling algorithm (DLAS). In Section 3, the proposed *e*DLA-based algorithm for sampling OSNs is described. The performance of the *e*DLA-based sampling algorithm is studied through several computer simulations on the well-known real and synthetic test networks in Section 4, and finally, Section 5 concludes this paper.

2. BACKGROUND AND PRELIMINARIES

2.1. Related work

Because, a few number of recent works on studying, characterizing, and estimating the properties of complex real networks such as OSN via sampling are presented in literature, accuracy and reliability of the studying, characterizing, and estimating the properties of OSN depends on the goodness of the network sampling methods. In literature for network sampling, the simple studies of existing research for network sampling focused on the random selection sampling and its variants. The

DOI: 10.1002/dac

random selection sampling methods construct sampled network by random selecting based on either nodes (also called random node sampling or random node selection) or edges (also called random edge sampling or random edge selection) without considering its topological structure of original network [13].

Another group of network sampling algorithms tried to take advantage of the topological structure of the original network by traversing in the original network. Depth-first search [18] and breadth-first search (BFS) [19] are the basic network sampling algorithms that they have been widely using for sampling in several kinds of complex networks. The BFS begins at a root node and traverses all the adjacent nodes. Then for each of those adjacent nodes in turn, it explores their neighboring nodes which were not selected, and so on. Finally, BFS traverses all vertices within some distance from the root vertex. In [20], it is shown that in random graphs, BFS is equivalent to other graph traversal-based sampling such as depth-first search, forest fire sampling [20], and snowball sampling [21]. Because the BFS traverses the whole graph whereas there are several methods which traverse only a portion of the original graph such as RWS method and its variations, RWS [16] method has been easily used for collecting samples from real-world networks with taking advantages of natural topological structure of original network. RWS [22] as a graph traversal-based sampling method extensively has also been applied in various network domains. RWS begins at a seed node and visits iteratively an adjacent node uniformly at random. Then for each of next adjacent nodes in turn, it traverses their next neighboring nodes that were not selected, and so on. Due to the simplicity of RWS, variants of RWS methods such as weighted random walk [23], MHRW [17], respondent-driven sampling (RDS) [24], and stratified weighted random walk [23] are developed and used in literature. It is noted that, in RWS, a node with more edges will have higher probability of being sampled. Therefore, the sample mean of sampled network collected by RWS tends to overestimate the average degree.

A worthy research for sampling from complex networks with exhaustive experiments was presented by Leskovec *et al.* [13]. They introduced several basic sampling methods for large graph subject to back in time and down scale. In this research, it is shown that RNS and RES do not provide appropriate sampled network. In [18] Lee *et al.,* it is demonstrated that RNS performs better than RES in terms of estimating the clustering coefficient of networks. Kim *et al.* provide a wide view about sampling in scale-free networks [25]. They ranked randomly sampled nodes based on degree, betweenness, and closeness centrality measures in order to obtain high central nodes. In [26], a practical framework for uniform sampling from *Facebook* users has been presented. In this study, the advantages of unbiased estimation of MHRW and re-weighted random walk over random sampling and BFS have been addressed with comparing several network sampling methods. Sampling algorithm based on trace routing for estimating cumulative distribution of degrees is presented by Ribeiro and Towsley [27]. Ribeiro *et al.* presented *m* dependent random walkers called frontier sampling in order to mitigate the high estimation errors caused by disconnected or loosely connected components in order to obtain better sampling. The authors showed that the frontier sampling is more robust than single and multiple independent random walkers to estimate in-degree distributions. Sampling for modular structure of networks has been studied by Maiya *et al.* via identifying the communities of the original networks [28]. Rejaie *et al.* tried to estimate the number of users of *MySpace* and *Twitter* by generating sequential user id [29], but this technique failed for those OSN sites where the user id is randomly generated and there is no prior knowledge about the process of generating user id by OSN sites. In [30], RDS analyzed in order to eliminate the biases related to chain referral sampling of hidden populations. In [31], authors used RDS sampling from *Twitter* to study several characterizations of Twitter. They have also shown through experimentations on *Twitter* that RDS has lower error in comparison with MHRW.

An analytical comparison between RWS and BFS sampling has been addressed by Kurant *et al.* [11] to sample from network. Their study revealed that the graph degree is overestimated by the BFS, while it is underestimated by the RWS. Therefore, they suggested analytical solutions to eliminate the biasness of estimation. Based on the different types of relationship between users in OSN, RWS on multi-graph has been introduced in [9] and their simulation results indicated superiority of the *Gjoka's* method. Random jump strategy into MHRW is introduced in order to avoid being trapped in local structures by Jin *et al.* for unbiased estimation [32]. It has also been considered the structure of bipartite graphs by Wang *et al.* [33] for some types of OSN. They utilized MHRW for sampling from bipartite graphs and improved the stability of MHRW. Analysis of RWS method has been addressed by Cooper *et al.* [34]; in this study, the authors tried to take a sample from the high-degree vertices and similar networks regarding the power law distribution. Lu *et al.* studied network sampling on *Twitter* data, and they showed that RWS is much better than RNS [35]. Sampling by crawling the edges of the original graph has been discussed in [36] by the idea of PageRank algorithm to find strong community structures to include the set of nodes with highest proximity to the initial node. They have been shown that this algorithm performs better than RDS in terms of Kolmogorov–Smirnov (KS) distance on some well-known real networks. In [37] for network sampling, authors focused on the node attributes of original network such as estimation of user attributes, user profiles, user tags, user interests, and user preferences of OSN and then they investigated several network sampling techniques.

In [38], the uniform random sampling of connected induced subgraphs of a prescribed size from a network studied by Lu *et al.* In this work, nodes that are initially selected by random walk method are replaced with random nodes in order to reduce the bias toward high-degree nodes. They investigated the practicality and effectiveness of several techniques such as acceptance–rejection sampling, random vertex expansion, Metropolis–Hastings sampling, and random walk on a Markov chain connected induced subgraphs. Sampling from directed graphs has been presented in [39] using random walk with jump. In [10], in the duration of the network sampling, they avoid visiting all nodes in the vicinity of a user in order to improve the performance of sampling. Sampling from directed heterogeneous graph for semantic sampling [40] is proposed by Yang *et al.* They demonstrated that their sampling algorithm preserves relational profile property. A network sampling method based on DLAS is reported by Rezvanian *et al.* [7] in order to collect highly important nodes (e.g., central nodes) of graph. It has been shown that this algorithm performs better than RDS and RWS in terms of relative error and KS distance on some well-known real networks. Based on the basic snowball sampling, a random multiple snowball with Cohen process sampling is developed by Gao *et al.* [41] to explore global information and local structure of the networks at the same time. Their simulations on computer generated networks indicated that this method is able to preserve local and global structure of network. In [42], an improved version of forest fire sampling algorithm was proposed to decrease the scale of network data and preserve the network community structure of original network as well. Summary of some of the existing algorithms for sampling social network with their network statistics are given in Table I.

2.2. Learning automata

The LA [43] refers to an abstract model that randomly chooses an action through its finite set of actions and performs it on an unknown random environment. The action selected at random depends on the probability distribution kept over the set of actions of learning automaton, and at each time, the selected action is considered as the input to the unknown random environment. Environment then evaluates the chosen action and responds with a reinforcement signal to the automaton. According to the chosen action and received reinforcement signal, the learning automaton updates its internal state and chooses its next action. In general, the aim of a learning automaton is to approach the optimal action from the set of actions so that the average penalty received from the environment is minimized. Formally, the environment of learning automata can be defined by a 3-tuple $E=\{\alpha, \beta, c\}$, where $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of inputs (set of r actions), $\beta=\{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of output (values that can be taken by the reinforcement signal), and $c=\{c_1, c_2, \ldots, c_m\}$ is the set of penalty probabilities (the probability computed by the reinforcement signal of the environment) that each c_i is depend on the given action α_i.

The environment is classified into stationary or non-stationary by which the penalty probabilities are constant or variable, respectively. Besides, according to the type of the reinforcement signal β, the environment can be categorized into *P-model*, *Q-model*, and *S-model*. P-model refers to an environment wherein the reinforcement signal is able to take two binary values 0 and 1. The environment in which the reinforcement signal can take a finite number of the values in the interval [0, 1] is

Table I. Summary of some of the existing algorithms for sampling social networks.

Refs.	Basic strategy	Main idea	Network statistics
[16]	Graph traversal (RW)	Using statistical properties of the network and a random walk algorithm in order to obtain high-degree nodes.	Clustering coefficient and degree distribution
[17]	Graph traversal (RW, MHRW)	Avoiding visiting the already visited nodes in order to increase the search efficiency and also obtaining unbiased graph sampling.	Degree distribution
[7]	Graph traversal	Traversing network using distributed learning automata in order to collect highly important nodes (e.g., central nodes) of graph.	Clustering coefficient and degree distribution
[20]	Graph traversal (BFS)	Applying the degree bias correction to improve the sample obtained using breadth-first search sampling algorithm.	Degree distribution
[25]	Node selection	Ranking randomly sampled nodes based on degree, betweenness, and closeness centrality measures in order to obtain high central nodes.	Degree, betweenness, and closeness
[27]	Graph traversal (RW)	Using multiple dependent random walkers in order to mitigate the high estimation errors caused by disconnected or loosely connected components in order to obtain better sampling.	In-degree distribution, out-degree distribution, and clustering coefficient
[31]	Graph traversal (RW)	Using random walk-based sampling algorithm called respondent-driven sampling for sampling Twitter.	Out-degree distribution, In-degree distribution
[32]	Graph traversal (MHRW)	Introducing a random jump strategy into Metropolis–Hasting random walk in order to avoid being trapped in local structures.	In-degree distribution, Out-degree distribution
[33]	Graph traversal (MHRW)	Extending Metropolis–Hastings random walk algorithm for sampling bipartite graphs and improving the stability of Metropolis–Hastings random walk.	Out-degree distribution
[34]	Graph traversal (RW)	Using biased random walk algorithm to find high-degree nodes.	Degree distribution
[36]	Graph traversal	Using the PageRank algorithm to find strong community structures to include the set of nodes with highest proximity to the initial node.	Degree distribution
[38]	Graph traversal (RW)	Replacing nodes that are initially selected by random walk algorithm with random nodes in order to eliminate the bias toward high-degree vertices.	In-degree distribution and clustering coefficient
[39]	Graph traversal (RW)	Using random walk algorithm with jump for directed graphs.	Degree distribution
[41]	Graph traversal (SNB)	Combining random node and snowball sampling to explore global information and local structure of the networks at the same time.	Degree distribution, connectivity rate, and average shortest path

RW, random walk; MHRW, Metropolis–Hastings random walk sampling; BFS, breadth-first search.

Q-model. In an S-model of the environment, the reinforcement signal is a continuous random variable in the interval [0, 1]. Figure 1 depicts a simple relationship between the LA and its random environment.

Let $\alpha_i(k) \in \alpha$ be the action that is selected by a learning automaton from its action-set α and $p(k)$ is the action probability vector at instant k. At each instant k, in linear learning algorithm, the action probability vector $p(k)$ is updated by the following equation

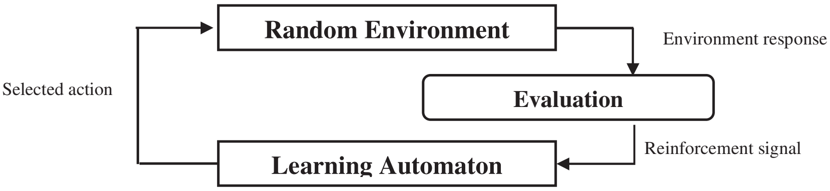


Figure 1. Relationship between the learning automaton and its random environment.

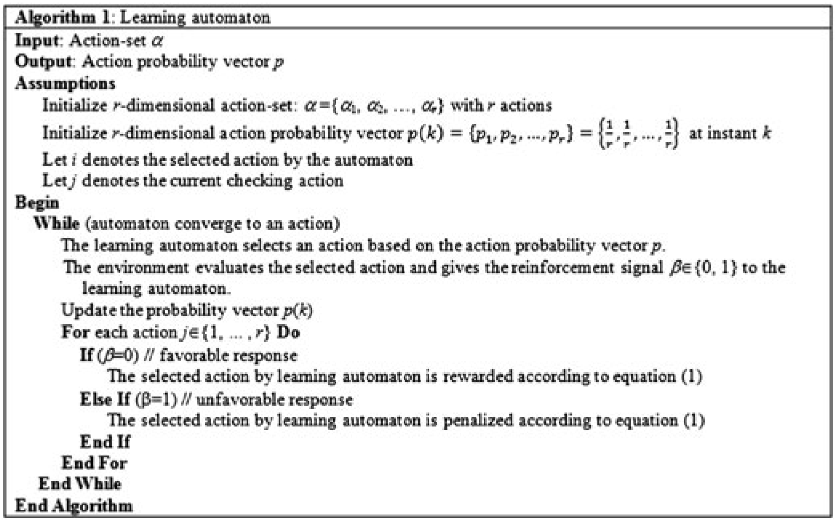
$$p_j(k+1) = \begin{cases} p_j(k) + a\left[1 - p_j(k)\right] & \text{if } i = j \text{ and selected action } \alpha_i(k) \text{ is rewarded} \\ (1-a)p_j(k) & \text{if } i \neq j \text{ and selected action } \alpha_i(k) \text{ is rewarded} \\ 1 - bp_j(k) & \text{if } i = j \text{ and selected action } \alpha_i(k) \text{ is penalized} \\ \dfrac{b}{r-1} + (1-b)p_j(k) & \text{if } i \neq j \text{ and selected action } \alpha_i(k) \text{ is penalized} \end{cases} \quad (1)$$

where a and b are reward parameter and penalty parameter, respectively. And also, r denotes the number of actions that can be taken by LA. The parameters a and b indicate the amount of increase and decrease of the action probability values, respectively. If $a = b$, the aforementioned recurrence equation is called *linear reward-penalty* ($L_{R\text{-}P}$) algorithm; if $a >> b$, the presented equation is called *linear reward-εpenalty* ($L_{R\text{-}\varepsilon P}$) algorithm; and finally, if $b = 0$, it is called *linear reward-Inaction* ($L_{R\text{-}I}$) algorithm. A simple pseudo code for updating probabilities of a learning automaton in a stationary environment with $\beta \in \{0,1\}$ and r actions is presented in Figure 2.

The LA as a simple and effective technique has found successfully applications in complex and dynamical environments where a large amount of uncertainty or lack of information about the environment exists. In the recent years, LA have arisen to different network applications such as wireless sensor networks [44], WiMAX networks [45], network security [46], wireless mesh networks [47], mobile video surveillance [48], vehicular environment [49, 50], Peer-to-Peer networks [51], wireless data broadcasting systems [52–54], smart grid systems [55], grid computing [56], and cloud computing [57], to mention a few.

2.3. Learning automata with variable number of actions

In variable action-set learning automata (also called a learning automaton with variable number of actions), the number of available actions that can be taken by the learning automaton at every



```
Algorithm 1: Learning automaton
Input: Action-set $\alpha$
Output: Action probability vector $p$
Assumptions
    Initialize $r$-dimensional action-set: $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ with $r$ actions
    Initialize $r$-dimensional action probability vector $p(k) = \{p_1, p_2, \ldots, p_r\} = \left\{\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r}\right\}$ at instant $k$
    Let $i$ denotes the selected action by the automaton
    Let $j$ denotes the current checking action
Begin
  While (automaton converge to an action)
      The learning automaton selects an action based on the action probability vector $p$.
      The environment evaluates the selected action and gives the reinforcement signal $\beta\in\{0, 1\}$ to the
         learning automaton.
      Update the probability vector $p(k)$
      For each action $j\in\{1, \ldots, r\}$ Do
         If ($\beta$=0) // favorable response
             The selected action by learning automaton is rewarded according to equation (1)
         Else If ($\beta$=1) // unfavorable response
             The selected action by learning automaton is penalized according to equation (1)
         End If
      End For
  End While
End Algorithm
```

Figure 2. Pseudocode of action probability vector update of a learning automaton.

instant can be changed with time [58]. In variable action set learning automata, at each instant *k,* a learning automaton selects an action based on the scaled action probability vector $\widehat{p}(k)$ defined as

$$\widehat{p}_i(k) = \frac{p_i(k)}{K(k)} \tag{2}$$

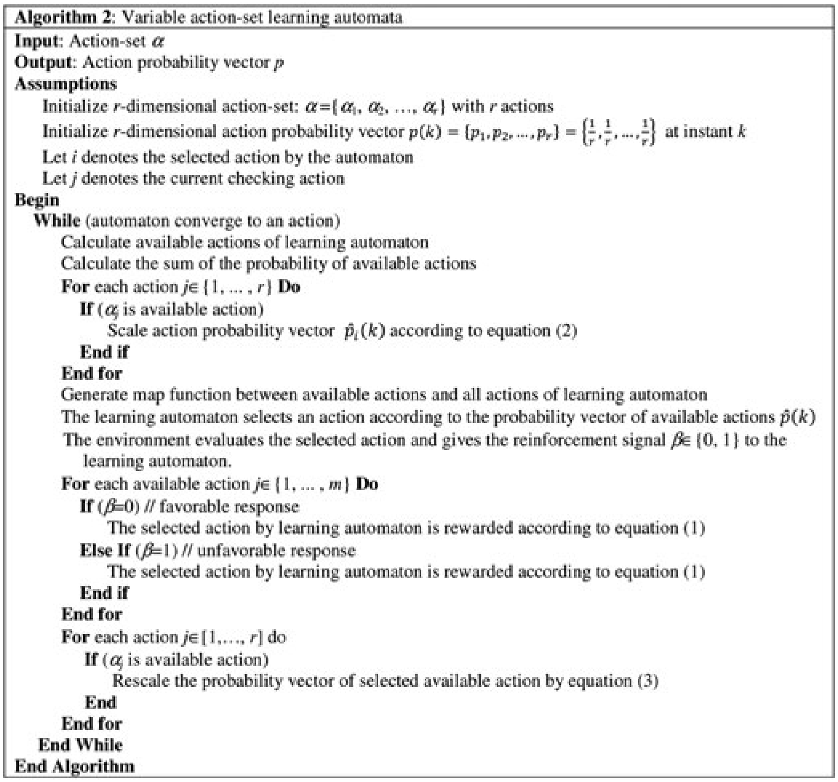
where $K(k)$ is the sum of the probabilities of the available actions at instant k, and $p_i(k) = prob[\alpha(k) = \alpha_i]$. And then the selected action performed on the environment. After receiving the reinforcement signal, the LA will similarly update probability vector of available actions via learning algorithm given in Equation (1). Finally, the probability vector of available actions $p(k+1)$ is rescaled as follows:

$$p_i(k+1) = \widehat{p}_i(k+1).K(k) \tag{3}$$

The pseudo code of probability update of a variable action set learning automaton is shown in Figure 3.

2.4. Distributed learning automata

A DLA [14] is a network of LA that collectively cooperates to solve a particular problem. A DLA can be modeled by a directed graph in which the vertex-set of graph constructs the set of LA, and usually, the set of neighboring vertices (or also set of connected edges) for each vertex constructs the action-set for corresponding learning automaton. In other words, the number of actions for a learning automaton in DLA equals to the number of LA that are connected to this learning automaton. In DLA, when an action is selected by an LA in the network of LA, an LA corresponding to the selected neighboring vertex (or also other LA at the end of the chosen edge) is activated. Formally, a DLA can be defined by a 4-tuple $\langle A, E, T, A_0 \rangle$, where $A = \{A_1, A_2, \ldots, A_n\}$ is the set of



```
Algorithm 2: Variable action-set learning automata
Input: Action-set $\alpha$
Output: Action probability vector $p$
Assumptions
    Initialize $r$-dimensional action-set: $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ with $r$ actions
    Initialize $r$-dimensional action probability vector $p(k) = \{p_1, p_2, \ldots, p_r\} = \left\{\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r}\right\}$ at instant $k$
    Let $i$ denotes the selected action by the automaton
    Let $j$ denotes the current checking action
Begin
  While (automaton converge to an action)
      Calculate available actions of learning automaton
      Calculate the sum of the probability of available actions
      For each action $j \in \{1, \ldots, r\}$ Do
          If ($\alpha_j$ is available action)
              Scale action probability vector $\hat{p}_i(k)$ according to equation (2)
          End if
      End for
      Generate map function between available actions and all actions of learning automaton
      The learning automaton selects an action according to the probability vector of available actions $\hat{p}(k)$
      The environment evaluates the selected action and gives the reinforcement signal $\beta \in \{0, 1\}$ to the
         learning automaton.
      For each available action $j \in \{1, \ldots, m\}$ Do
          If ($\beta$=0) // favorable response
              The selected action by learning automaton is rewarded according to equation (1)
          Else If ($\beta$=1) // unfavorable response
              The selected action by learning automaton is rewarded according to equation (1)
          End if
      End for
      For each action $j \in [1, \ldots, r]$ do
          If ($\alpha_j$ is available action)
              Rescale the probability vector of selected available action by equation (3)
          End
      End for
  End While
End Algorithm
```

Figure 3. Pseudocode of action probability vector update of a variable action-set learning automaton.

LA, $E \subseteq A \times A$ is the edge-set in which edge (v_i, v_j) corresponds to action α_i^j of learning automaton A_i, T is the learning scheme with which the LA update their action probability vectors, and A_0 is the starting LA (or also root LA) of DLA from which the starting LA activates.

The procedure of performing a DLA can be described as follows. Let A_0 be the starting learning automaton in DLA and action set of each LA be the set of outgoing edges of corresponding vertex. The starting learning automaton A_0 randomly selects an action according to its action probabilities and then activates the LA at the other end of the chosen edge. The activated LA selects an action that results in the activation of another LA. The process of selecting an action by a learning automaton and activating the other learning automaton is continued until some stopping criteria depending on the problem for which DLA is designed are met. DLA may stop if either the action of the activated LA is empty or all the vertices of the graph have been visited. The selected actions along the traversed path or in the part of a solution of a particular problem by the activated LA are performed on the random environment. The environment evaluates the performed actions and generates a reinforcement signal to the DLA. The action probability vectors of the activated LA along the traversed path or LA of the vertices that are part of a solution to the problem are then updated on the basis of the reinforcement signal. Activation of DLA by starting from its root LA is repeated a specified number of times until the solution of the problem for which DLA is designed is obtained.

Since the introduction of DLA by Beigy and Meybodi [14], DLA has been applied for various applications such as vehicle routing problem [59], stochastic graph problems [15, 60–63], wireless sensor networks [64], web mining [65], complex networks [7], social network analysis [66], and grid computing [67].

2.5. *Extended distributed learning automata*

An *e*DLA [15] is a new extension of DLA supervised by a set of rules governing the operation of the LAs. Mollakhalili Meybodi *et al.* using *e*DLA presented a framework based on LA for solving stochastic graph optimization problems such as stochastic minimum spanning tree problem and stochastic shortest path problem. Here, we provide a brief introduction to the *e*DLA and the interested reader may refer to [15] for theoretical properties of the *e*DLA. In general in *e*DLA, the ability of a DLA is improved by adding communication rules and changing the activity level of each LA depends on a problem to be solved by an *e*DLA. An *e*DLA similar to a DA can be modeled by a directed graph in which the vertex-set of graph constructs the set of LA and the number of actions for a LA in *e*DLA equals to the number of LAs that are connected to this LA. In *e*DLA, at any time, not only each LA can be in one mode of activity level but also each LA with a high activity level can be employed an action according to its probabilities on the random environment. Formally, an *e*DLA can be described by a 7-tuple $\langle A, E, S, P, S^0, F, C \rangle$, where A is the set of LA, $E \subseteq A \times A$ is the edge-set of communication graph $G(V, E)$ and $S = \{s_1, s_2, \ldots, s_n\}$ is a set of activity levels corresponding to each LA in *e*DLA; specially s_i indicates the activity level for learning automaton A_i in which $s_i \in \{Pa, Ac, Fi, Of\}$ consists of one of the following activity levels: *Passive* (initial level of each LA and can be changed to *Active*), *Active* (activity level for set of available LAs and its level can be upgraded to *Fire*), *Fire* (the highest level of activity, LA can be performed and its level can be changed to *Off*), and *Off* (the lowest level of activity, LA is disabled and its level stay unchanged), represented briefly by *Pa*, *Ac*, *Fi*, and *Of*, respectively. As mentioned, at any time only one LA in *e*DLA can be in the *Fi* level of activity and can be determined by *Fire function C*, which randomly selects an LA from a set of LAs with activity level of *Ac. Governing rule P* is the finite set of rules that governs the activity levels of each LA. *P* according to the current activity level of each LA, its adjacent LA or depending on the particular problem which *e*DLA is designed is defined. $S^0 = (s_1{}^0{}_, s_2{}^0, \ldots, s_n{}^0)$ and $F = \{S^F \mid S^F = (s_1{}^F, s_2{}^F, \ldots, s_n{}^F)\}$ are the *initial state* and *final conditions* of *e*DLA.

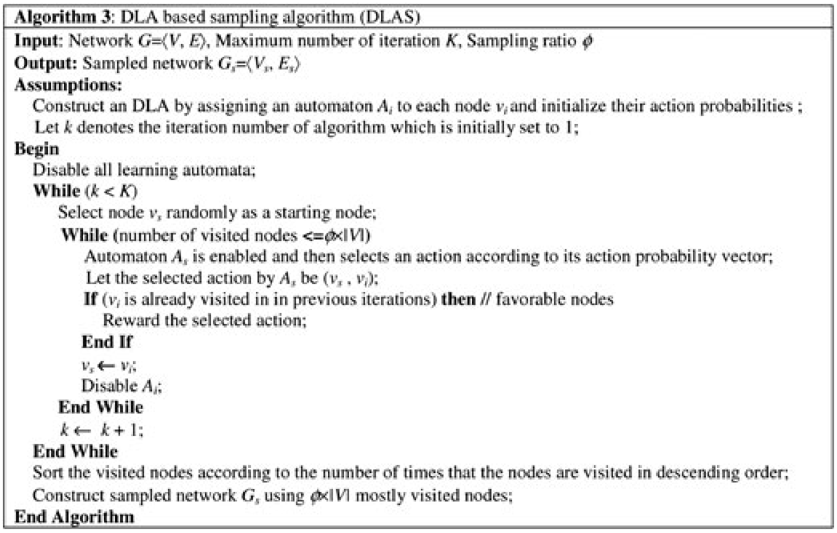
The operation of *e*DLA can be described as follows. In *e*DLA, at first at initial state S^0, a starting LA is randomly selected by firing function F to fires, selects one of outgoing edges (actions) according to its action probabilities and performs it on the random environment and at the same time the activity level of fired LA and neighboring LA are changed to *Of* and *Ac*, respectively. Changing activity levels of LAs result in the state of *e*DLA transferred from state S^k to state S^{k+1} at instant k by governing rule P. Then the firing function C fires one LA from the set of LA with activity level of Ac to select an action and then changes its activity level and neighboring LAs. The process of firing one LA by firing function, performing an action by fired LA, and changing the activity level of fired LA and its neighbors by governing rule P is continued until the final condition of *e*DLA F is reached. F can be defined based on a set of criteria in terms of activity levels of LAs such that, if one of them is satisfied, the final condition of *e*DLA is realized. The environment evaluates the performed actions by fired LA and generates a reinforcement signal to *e*DLA. The action probabilities of fired LA along the visited vertices or LA of the vertices that are part of a solution to the problem of graph are then updated on the basis of the reinforcement signal according to the learning algorithm. Firing LAs of *e*DLA by starting from randomly LA is repeated a predefined number of times until the solution of the problem for which *e*DLA is designed is obtained.

2.6. Distributed learning automata-based sampling algorithm

Because the proposed sampling algorithm in this paper (*e*DLAS) is an extension of DLAS algorithm [7], in this section, we briefly explain DLAS. For this algorithm, it is assumed that each node of a social network is available by a unique ID of each user as nodes of input network and also their allowable connected users as corresponding edges between them. DLAS uses a set of LA in order to guide the process of visiting nodes by visiting the important parts of the input network, those parts of the network which contain important nodes. A DLA is constructed by assigning a learning automaton A_i to node v_i of the input network. Action-set of each LA corresponds to selecting the edges of the node to which the LA is assigned. Each LA in DLA initially selects its actions with equal probabilities. In an iteration of the algorithm, DLA starts from a randomly chosen node and then follows a path of nodes according to the probability vectors of set of LA constituting DLA. If a LA selects an action which corresponds to a node which has been already visited in the previous paths (iterations) then the probability of selecting that action will be increased according to the learning algorithm. After several iterations, the visited nodes are sorted according to their probabilities of selecting. The sampled network is then constructed using a particular given number (sampling ratio multiply the number of nodes in the network or also a given percentage) of mostly visited nodes. Pseudocode of DLAS is given in Figure 4.

3. PROPOSED *E*DLA-BASED SAMPLING ALGORITHM

In the proposed sampling algorithm, it is assumed that each node of a social network is available by a unique ID of each user as nodes of input network and also their allowable connected users as



```
Algorithm 3: DLA based sampling algorithm (DLAS)
Input: Network $G=\langle V, E\rangle$, Maximum number of iteration $K$, Sampling ratio $\phi$
Output: Sampled network $G_s=\langle V_s, E_s\rangle$
Assumptions:
  Construct an DLA by assigning an automaton $A_i$ to each node $v_i$and initialize their action probabilities ;
  Let $k$ denotes the iteration number of algorithm which is initially set to 1;
Begin
  Disable all learning automata;
  While ($k < K$)
    Select node $v_s$ randomly as a starting node;
    While (number of visited nodes $<=\phi\times|V|$)
      Automaton $A_s$ is enabled and then selects an action according to its action probability vector;
      Let the selected action by $A_s$ be ($v_s$ , $v_i$);
      If ($v_i$ is already visited in in previous iterations) then // favorable nodes
        Reward the selected action;
      End If
      $v_s \leftarrow v_i$;
      Disable $A_i$;
    End While
    $k \leftarrow k + 1$;
  End While
  Sort the visited nodes according to the number of times that the nodes are visited in descending order;
  Construct sampled network $G_s$ using $\phi\times|V|$ mostly visited nodes;
End Algorithm
```

Figure 4. Pseudocode of distributed learning automata-based sampling algorithm (DLAS).

corresponding edges between them. The proposed sampling algorithm uses an *e*DLA to guide the process of visiting nodes by visiting the important parts of the input network, those parts of the network which contain more hub and central nodes. At first, an *e*DLA isomorphic to the input network is constructed by assigning a learning automaton A_i to node v_i of the input network. Action-set of each LA corresponds to the selecting an edge of the node to which the LA is assigned with equal probabilities. In *e*DLA, at any time, a LA can be in one of four levels: *Passive*, *Active*, *Fire*, and *Off.* Thus, in the proposed sampling algorithm, all LA are initially set to *Passive* level. At the first of algorithm, one of the nodes in *e*DLA is chosen by firing function to be the starting node and its activity level is set to *active* by governing rule. In what follows, an iteration of the proposed sampling algorithm is explained.

At the beginning of an iteration, one of the *active* nodes chosen randomly by firing function and then fired (its activity level changes to *fire* and the activity level of its *passive* neighboring nodes change to *active*) and chooses one of its actions that correspond to one of the edges of the *fired* node, and then the level of the *fired* LA changes to *off* by governing rule. Then one of LA in *e*DLA with activity level *active* is chosen and fired (its activity level changes to *fire* and the activity level of its *passive* neighboring nodes change to *active*). The fired LA chooses one of its actions that correspond to an edge of the fired node, and then the activity level of the fired LA changes to *off* by governing rule. The process of selecting a learning automaton from the set of LA with activity level *active* in *e*DLA by firing function, firing it, choosing an action by fired LA, changing activity levels of *passive* adjacent LA to *active* LA, and changing activity level of fired LA from *fire* to *Off* by governing rule is repeated until either the number of LA with activity level *Off* reached a given number (sampling ratio multiply the number of nodes in the network) or the set of LA with activity level *active* is empty. In the proposed sampling algorithm, the action probabilities of all fired LA are updated on the basis of the responses received from the environment. The probabilities are updated as follows: the probabilities of choosing actions of all fired LA that correspond to the edges traversed in the previous iterations are updated according to the learning algorithm. At the end of each iteration, the activity level of all learning automata in *e*DLA is set to *passive.* Thus, the proposed sampling algorithm iteratively visits a set of paths and also updates its action probability vectors until it iterates the maximum number of iterations. After that, the visited nodes are sorted according to the number of times that they have been visited in descending order. The sampled network is then constructed using a given number (sampling ratio multiply the number of nodes in the network or also a particular percentage) of mostly visited nodes. Figure 5 is the pseudocode of *e*DLA-based sampling algorithm.

In the proposed sampling algorithm, visiting a node causes the probability of visiting this node in later iterations to be increased. This means that mostly visited nodes have incoming edges with high probabilities. That means that the proposed sampling algorithm constructs a sample of the network, which includes central and hub nodes leading to a better sampling of the network.

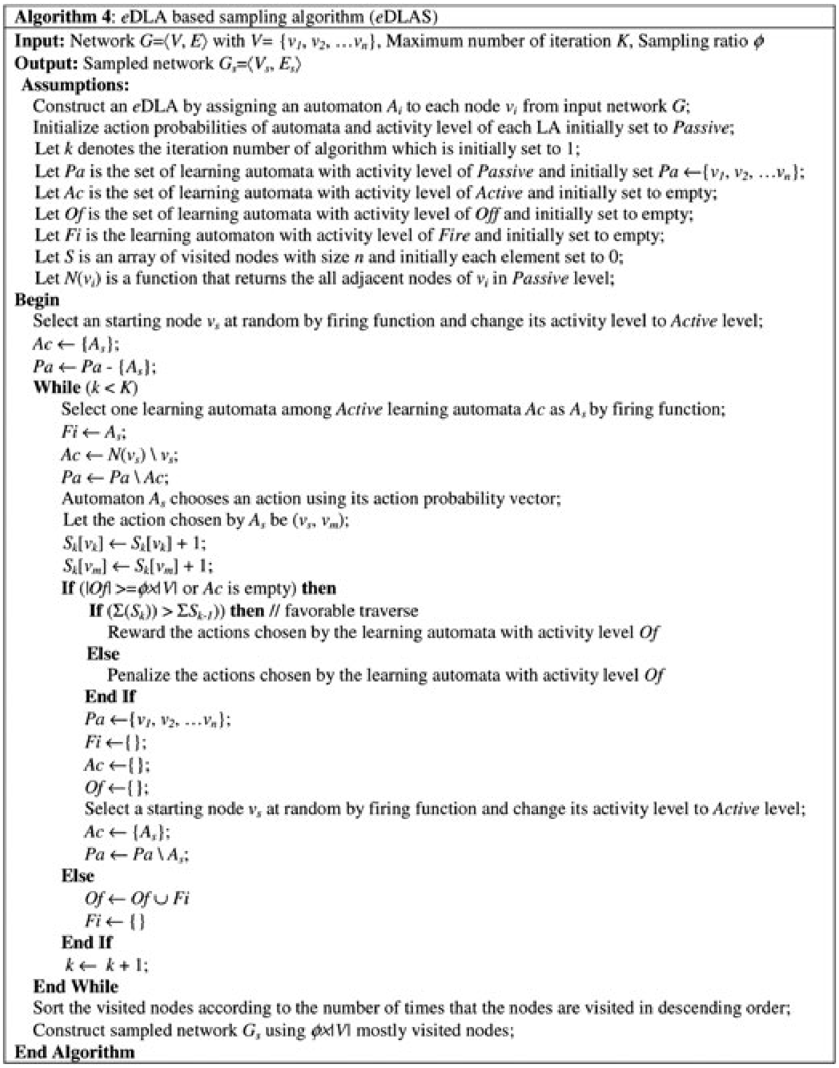
4. SIMULATION RESULTS

In this section, performance of the *e*DLA-based sampling algorithm is investigated on several well-known real and synthetic test networks, which are commonly used benchmarks for simulations. Table II describes the test networks used for the experimentations and their characteristics. We use well-known computer-generated network model of Barabási–Albert model (BA model) for scale-free networks with heavy-tailed degree distribution [6], which is utilized widely in literature. We set network parameters of BA model as $N = \{1000, 2000, 5000, 10000\}$ and $m_0 = m = 5$. We also test the proposed sampling algorithm on several real networks such as *ca-HepTh* (Collaboration network of ArXiv High Energy Physics), *ca-GrQc* (Collaboration network of ArXiv General Relativity), and *ca-CondMat* (Collaboration network of ArXiv Condense Matter Physics).

4.1. Evaluation metrics

In this paper, we use KS test and relative error (RE) as distance measures for performance studies. The evaluation metrics are described in the rest of this subsection.

DOI: 10.1002/dac



```
Algorithm 4: eDLA based sampling algorithm (eDLAS)
Input: Network $G=\langle V, E\rangle$ with $V=\{v_1, v_2, \ldots v_n\}$, Maximum number of iteration $K$, Sampling ratio $\phi$
Output: Sampled network $G_s=\langle V_s, E_s\rangle$
 Assumptions:
   Construct an eDLA by assigning an automaton $A_i$ to each node $v_i$ from input network $G$;
   Initialize action probabilities of automata and activity level of each LA initially set to Passive;
   Let $k$ denotes the iteration number of algorithm which is initially set to 1;
   Let $Pa$ is the set of learning automata with activity level of Passive and initially set $Pa \leftarrow \{v_1, v_2, \ldots v_n\}$;
   Let $Ac$ is the set of learning automata with activity level of Active and initially set to empty;
   Let $Of$ is the set of learning automata with activity level of Off and initially set to empty;
   Let $Fi$ is the learning automaton with activity level of Fire and initially set to empty;
   Let $S$ is an array of visited nodes with size $n$ and initially each element set to 0;
   Let $N(v_i)$ is a function that returns the all adjacent nodes of $v_i$ in Passive level;
Begin
   Select an starting node $v_s$ at random by firing function and change its activity level to Active level;
   $Ac \leftarrow \{A_s\}$;
   $Pa \leftarrow Pa - \{A_s\}$;
   While ($k < K$)
       Select one learning automata among Active learning automata $Ac$ as $A_s$ by firing function;
       $Fi \leftarrow A_s$;
       $Ac \leftarrow N(v_s) \setminus v_s$;
       $Pa \leftarrow Pa \setminus Ac$;
       Automaton $A_s$ chooses an action using its action probability vector;
       Let the action chosen by $A_s$ be $(v_s, v_m)$;
       $S_k[v_k] \leftarrow S_k[v_k] + 1$;
       $S_k[v_m] \leftarrow S_k[v_m] + 1$;
       If ($|Of| >= \phi \times |V|$ or $Ac$ is empty) then
           If ($(\Sigma(S_k)) > \Sigma S_{k-1})$) then // favorable traverse
               Reward the actions chosen by the learning automata with activity level $Of$
           Else
               Penalize the actions chosen by the learning automata with activity level $Of$
           End If
           $Pa \leftarrow \{v_1, v_2, \ldots v_n\}$;
           $Fi \leftarrow \{\}$;
           $Ac \leftarrow \{\}$;
           $Of \leftarrow \{\}$;
           Select a starting node $v_s$ at random by firing function and change its activity level to Active level;
           $Ac \leftarrow \{A_s\}$;
           $Pa \leftarrow Pa \setminus A_s$;
       Else
           $Of \leftarrow Of \cup Fi$
           $Fi \leftarrow \{\}$
       End If
       $k \leftarrow k + 1$;
   End While
   Sort the visited nodes according to the number of times that the nodes are visited in descending order;
   Construct sampled network $G_s$ using $\phi \times |V|$ mostly visited nodes;
End Algorithm
```

Figure 5. The pseudocode of the extended distributed learning automata-based sampling algorithm (*e*DLAS).

Table II. Description of test networks.

Network	Node	Edge	Description
BA-1000	1000	9952	Synthetic scale-free network
BA-2000	2000	19952	Synthetic scale-free network
BA-5000	5000	49944	Synthetic scale-free network
BA-10000	10000	99945	Synthetic scale-free network
ca-GrQc	5242	28980	Collaboration network of ArXiv General Relativity
ca-HepTh	9877	25998	Collaboration network of ArXiv High Energy Physics
ca-CondMat	23133	93497	Collaboration network of ArXiv Condense Matter Physics

4.1.1. Kolmogorov–Smirnov test. Kolmogorov–Smirnov D-Statistic is one of the statistical test methods used for calculating the distance between two cumulative distribution functions. *KS distance* is a commonly used measure for acceptability between distribution of the original network and distribution of the sampled network. It is calculated as the maximum absolute distance between the two distributions. The result of this test is a value between 0 and 1. As closer as it is to zero, both distributions will have a greater similarity, and as closer as it is to the unit, the two distributions will show a greater discrepancy [68]. This metric has been calculated as the following equation

DOI: 10.1002/dac

$$KS(F, F_s) = \max|F(x) - F_s(x)| \tag{4}$$

where F and F_s are cumulative distribution functions of a given distribution of the original network and a given distribution of the sampled network, respectively, and x is the range of the random variable. So it is calculated as the maximum vertical distance between the two distributions. In this paper, we apply KS test on the degree distribution of the original network and the sampled network.

4.1.2. Relative error. Relative error can be applied to assess the accuracy of the results, which is defined by the following equation:

$$RE = \frac{|\theta - \theta_s|}{\theta} \tag{5}$$

where, θ and θ_s denote the values of a network parameter (i.e., clustering coefficient) in the original network and the sampled network, respectively [10]. In this paper, we calculate RE for clustering coefficient in the original network and the sampled network. The local clustering coefficient C_i for node v_i is calculated as follows

$$C_i = \frac{2T_i}{d_i(d_i - 1)} \tag{6}$$

where T_i is the number of edges in its adjacent node of v_i and d_i is the degree of node v_i.

4.2. Experimental results

In order to investigate the performance of the *e*DLA-based sampling algorithm (DLAS), several experiments are conducted on the real and synthetic test networks as given in Table II. All simulations are conducted 30 times, and the average results are reported. Linear learning scheme $L_{R\text{-}I}$ with learning parameter 0.05 is used in all experiments except experiment I. For evaluation purpose, RE and KS test are used.

4.2.1. Experiment I. This experiment is carried out to study the effect of the learning parameter a on the performance of the proposed sampling algorithm. The learning parameter a is set as $a \in \{0.005, 0.007, 0.009, 0.01, 0.03, 0.05, 0.070.09, 0.1, 0.2\}$. For this experiment, the sampling rate is set to 20%. The result of RE for clustering coefficient is given in Table III for synthetic random graphs: ER-1000, ER-2000 and ER-5000; and also synthetic scale-free networks: BA-1000, BA-2000, and BA-5000. From the results, we observe that the performance of the proposed sampling algorithm is very sensitive to the choice of learning parameter a. Large value for learning parameter a results in lower accuracy while small value for learning parameter results in higher accuracy. Therefore, in the rest of experiments conducted in this paper, the learning parameter a is set to 0.05.

Table III. Relative error for clustering coefficient for different values of learning parameter a.

Learning parameter (a)	ER-1000	ER-2000	ER-5000	BA-1000	BA-2000	BA-5000
0.005	0.2304	0.0981	0.0232	0.3570	0.4148	0.4089
0.007	0.2289	0.0968	0.0233	0.3598	0.4139	0.4074
0.009	0.2269	**0.0931**	0.0236	0.3573	0.4198	0.4067
0.010	**0.2267**	0.0989	**0.0231**	0.3595	0.4197	0.4038
0.030	0.2371	0.1006	0.0245	0.3583	0.4131	**0.4024**
0.050	0.2383	0.1008	0.0253	**0.3545**	**0.4111**	0.4032
0.070	0.2992	0.1262	0.0394	0.3672	0.4696	0.4275
0.090	0.3598	0.1350	0.0433	0.4354	0.4846	0.4495
0.100	0.3682	0.1391	0.0415	0.4585	0.4955	0.4834
0.200	0.3598	0.1408	0.0453	0.4591	0.5039	0.4857

The best result is highlighted in boldface.

4.2.2. Experiment II. This experiment is performed to compare the performance of the *e*DLAS with RNS [13], RES [13], RWS [16], MHRW [17], and the only learning automata-based sampling algorithm (DLAS) [7] for sampling rates 10%, 20%, and 30%. The linear learning scheme $L_{R\text{-}I}$ with learning parameter $a = 0.05$ is used for both DLAS and *e*DLAS. The comparison is made in terms of KS test for degree distribution and RE for clustering coefficient (CC), and RE for degree distribution parameter, and then the results are summarized in Tables IV–XII. Tables IV–VI are for KS test for degree distribution, Tables VII–IX for RE for clustering coefficient, and Tables X–XII for RE for degree distribution. According to the results, we may conclude that for all the test networks, higher (lower) value for sampling rate results in lower (higher) value of RE and KS distance. From the results, we also see that the proposed sampling algorithm (*e*DLAS) outperforms other sampling

Table IV. Kolmogorov–Smirnov test for degree distribution for sampling rate = 10%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.89	0.86	0.56	0.63	0.59	0.45
BA-2000	0.89	0.83	0.68	0.55	0.57	0.43
BA-5000	0.88	0.73	0.64	0.54	0.52	0.38
BA-10000	0.53	0.52	0.49	0.51	0.49	0.31
ca-GrQc	0.70	0.27	0.16	0.28	0.24	0.09
ca-HepTh	0.67	0.22	0.15	0.25	0.15	0.12
ca-CondMat	0.65	0.55	0.56	0.51	0.49	0.43

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table V. Kolmogorov–Smirnov test for degree distribution for sampling rate = 20%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.83	0.63	0.50	0.19	0.41	0.13
BA-2000	0.84	0.44	0.51	0.36	0.30	0.31
BA-5000	0.83	0.55	0.52	0.41	0.41	0.37
BA-10000	0.49	0.48	0.52	0.30	0.46	0.28
ca-GrQc	0.51	0.17	0.19	0.31	0.12	0.07
ca-HepTh	0.55	0.18	0.14	0.24	0.14	0.11
ca-CondMat	0.59	0.50	0.51	0.48	0.43	0.41

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table VI. Kolmogorov–Smirnov test for degree distribution for sampling rate = 30%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.83	0.50	0.44	0.22	0.23	0.12
BA-2000	0.82	0.36	0.40	0.23	0.37	0.26
BA-5000	0.86	0.36	0.42	0.20	0.15	0.21
BA-10000	0.46	0.41	0.39	0.24	0.21	0.23
ca-GrQc	0.45	0.16	0.18	0.41	0.19	0.06
ca-HepTh	0.46	0.17	0.22	0.39	0.26	0.07
ca-CondMat	0.49	0.46	0.47	0.43	0.39	0.36

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table VII. Relative error for clustering coefficient for sampling rate = 10%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.80	0.52	0.52	0.48	0.48	0.42
BA-2000	0.71	0.57	0.66	0.61	0.56	0.56
BA-5000	0.56	0.53	0.52	0.51	0.49	0.46
BA-10000	0.52	0.51	0.53	0.49	0.51	0.48
ca-GrQc	0.88	0.58	0.37	0.36	0.32	0.30
ca-HepTh	0.52	0.48	0.44	0.41	0.40	0.38
ca-CondMat	0.53	0.49	0.45	0.40	0.41	0.39

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table VIII. Relative error for clustering coefficient for sampling rate = 20%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.68	0.50	0.41	0.42	0.47	0.34
BA-2000	0.67	0.52	0.57	0.53	0.45	0.43
BA-5000	0.50	0.49	0.46	0.49	0.42	0.41
BA-10000	0.49	0.46	0.45	0.45	0.44	0.43
ca-GrQc	0.74	0.36	0.28	0.38	0.29	0.23
ca-HepTh	0.50	0.42	0.43	0.36	0.34	0.35
ca-CondMat	0.49	0.45	0.41	0.39	0.38	0.36

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table IX. Relative error for clustering coefficient for sampling rate = 30%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.80	0.47	0.39	0.39	0.38	0.32
BA-2000	0.57	0.51	0.55	0.48	0.43	0.41
BA-5000	0.48	0.46	0.42	0.47	0.40	0.37
BA-10000	0.47	0.43	0.43	0.39	0.38	0.36
ca-GrQc	0.65	0.32	0.25	0.25	0.26	0.20
ca-HepTh	0.45	0.34	0.42	0.34	0.32	0.30
ca-CondMat	0.46	0.43	0.38	0.36	0.35	0.34

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table X. Relative error for degree distribution parameter for sampling rate = 10%.

Methods						
Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.21	0.07	0.06	0.11	0.09	0.05
BA-2000	0.15	0.13	0.12	0.19	0.05	0.04
BA-5000	0.19	0.11	0.13	0.17	0.08	0.06
BA-10000	0.25	0.19	0.15	0.17	0.10	0.11
ca-GrQc	0.39	0.31	0.22	0.19	0.14	0.16
ca-HepTh	0.10	0.10	0.20	0.02	0.08	0.01
ca-CondMat	0.27	0.21	0.16	0.15	0.12	0.12

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table XI. Relative error for degree distribution parameter for sampling rate = 20%.

Methods Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.13	0.17	0.06	0.09	0.05	0.04
BA-2000	0.13	0.12	0.07	0.15	0.04	0.03
BA-5000	0.17	0.10	0.09	0.12	0.04	0.04
BA-10000	0.19	0.17	0.13	0.16	0.09	0.09
ca-GrQc	0.32	0.26	0.19	0.18	0.12	0.11
ca-HepTh	0.09	0.08	0.19	0.02	0.07	0.01
ca-CondMat	0.20	0.17	0.13	0.12	0.09	0.08

RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

Table XII. RE for degree distribution parameter for sampling rate =30%.

Methods Networks	RNS	RES	RWS	MHRW	DLAS	*e*DLAS
BA-1000	0.07	0.04	0.04	0.05	0.03	0.03
BA-2000	0.11	0.10	0.05	0.06	0.03	0.02
BA-5000	0.14	0.08	0.04	0.11	0.03	0.03
BA-10000	0.15	0.14	0.09	0.14	0.08	0.07
ca-GrQc	0.24	0.22	0.16	0.12	0.10	0.09
ca-HepTh	0.07	0.06	0.10	0.02	0.01	0.01
ca-CondMat	0.15	0.13	0.11	0.08	0.04	0.03

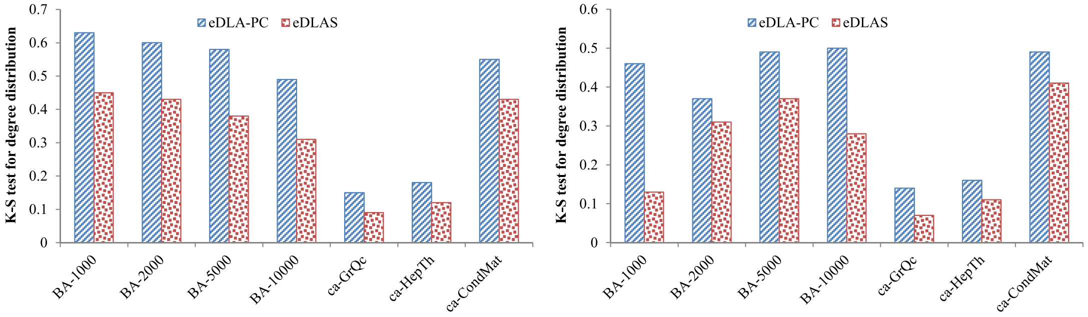
RNS, random node sampling; RES, random edge sampling; RWS, random walk sampling; MHRW, Metropolis–Hastings random walk sampling; DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling.

methods for many of the test networks and different sampling rates. The results also show that the *e*DLAS in comparison with the DLAS has a 26.93% improvement for the average KS value for degree distribution taken over all tested real and synthetic networks. These improvements for the average of RE values for degree distribution and clustering coefficient taken over all tested real and synthetic networks are 18.27% and 7.89%, respectively. Besides, the results indicate that in terms of KS test, the proposed sampling algorithm (*e*DLAS), DLAS, MHRW, RWS, RES, and RNS, has rank 1, 2, 3, 4, 5, and 6, respectively, for real and synthetic networks.

4.2.3. Experiment III. This experiment is conducted to compare *e*DLA-based sampling algorithm (*e*DLAS) with *e*DLA algorithm in which the leaning automaton residing in each node is replaced by a pure chance automaton (*e*DLA-PC). In a pure chance automaton, the actions are always chosen with the same probabilities [43]. These two algorithms are compared in terms of KS test distance for degree distribution and RE for clustering coefficient and degree distribution. Results of this experiment are given in Figures 6–8. According to the results, we may conclude that *e*DLAS perform better than *e*DLA-PC. That is indication of the fact that traversal of the network with the help of LA leads to a better sampling of the network. Similar results from comparing DLAS and DLAS in which the leaning automaton residing in each node is replaced by a pure chance automaton (DLAS-PC) are also obtained.

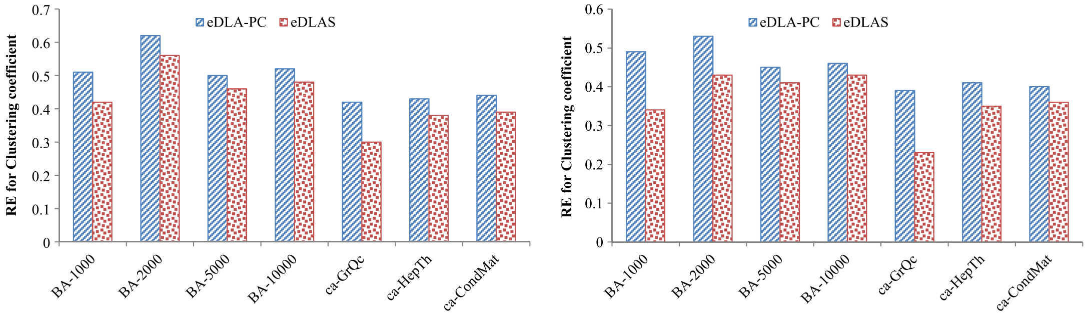
4.2.4. Experiment IV. In this experiment, we study the impact of size of the network on the performance of the proposed sampling algorithm (*e*DLAS) in terms of KS test for degree distribution. The algorithm is tested on synthetic scale-free network with sizes n varying from 100 to 1000 with increment 100 and sampling rates varying from 10% to 30% with increment 5%. The result of this

DOI: 10.1002/dac



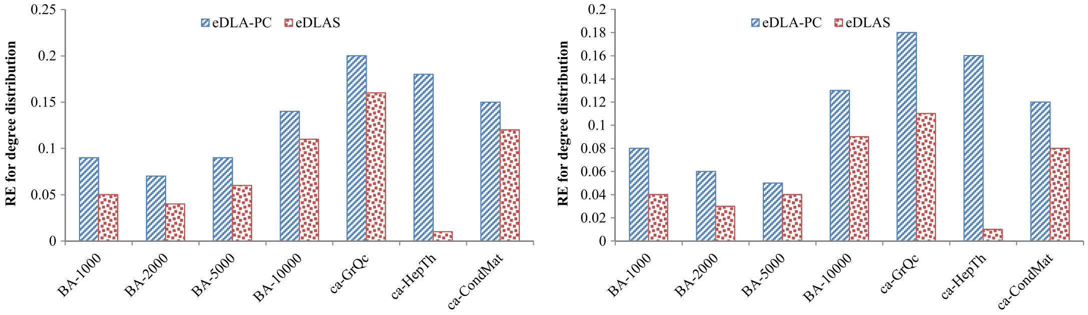
(a) sampling rate=10% (b) sampling rate=20%

Figure 6. Kolmogorov–Smirnov test for degree distribution for different sampling rates.



(a) sampling rate=10% (b) sampling rate=20%

Figure 7. Relative error for clustering coefficient for different sampling rates.



(a) sampling rate=10% (b) sampling rate=20%

Figure 8. Relative error for degree distribution parameter for different sampling rates.

experiment presented in Figure 9 indicates that increasing the size of the input network results in decreasing KS distance for all sampling rates.

4.2.5. Experiment V. This experiment is carried out to study the impact of density of the input network on the performance of the proposed sampling algorithm in terms of RE for CC. In order to perform this experiment, we plot RE for CC versus density for synthetic random network (ER) and also small-world (WS) networks both with 1000 nodes. The result of this experiment for different sampling rates is given in Figure 10. The results indicate that increasing density of input network results in decreasing relative error for both synthetic random and synthetic small-world networks. As can be seen, the slope of the plots for small-world networks is relatively lower than the slope

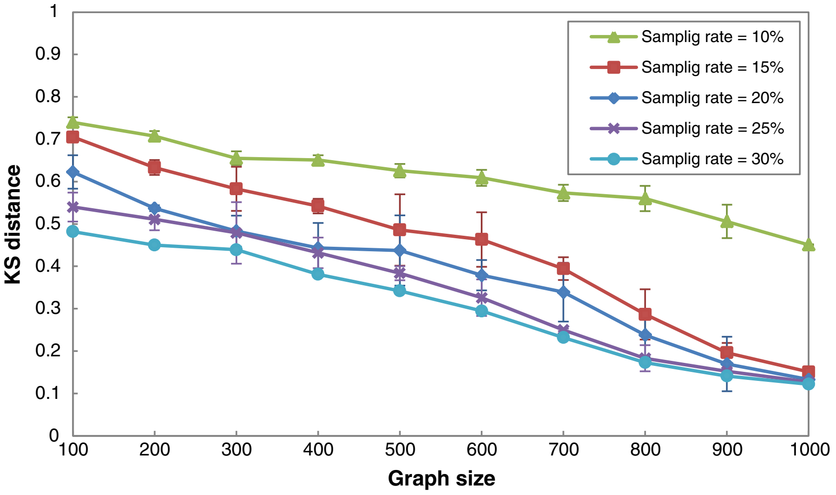


Figure 9. Kolmogorov–Smirnov distance versus network size for different sampling rate.

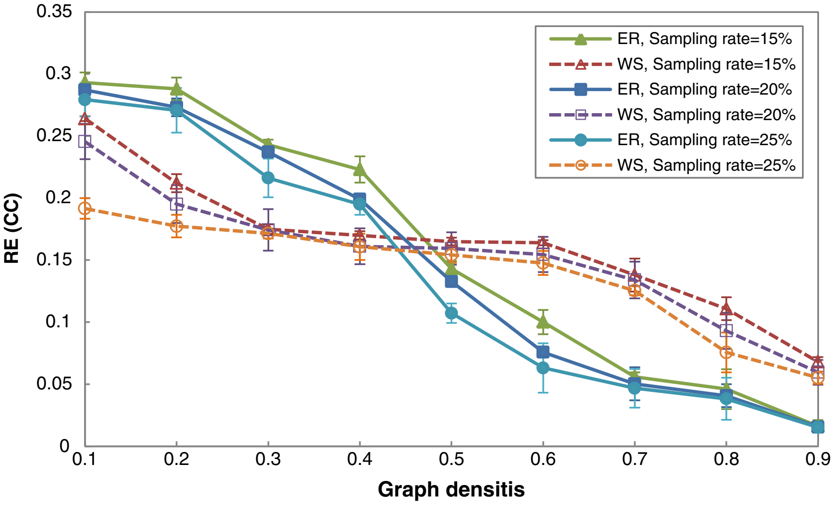


Figure 10. Relative error for clustering coefficient for synthetic test networks with different densities.

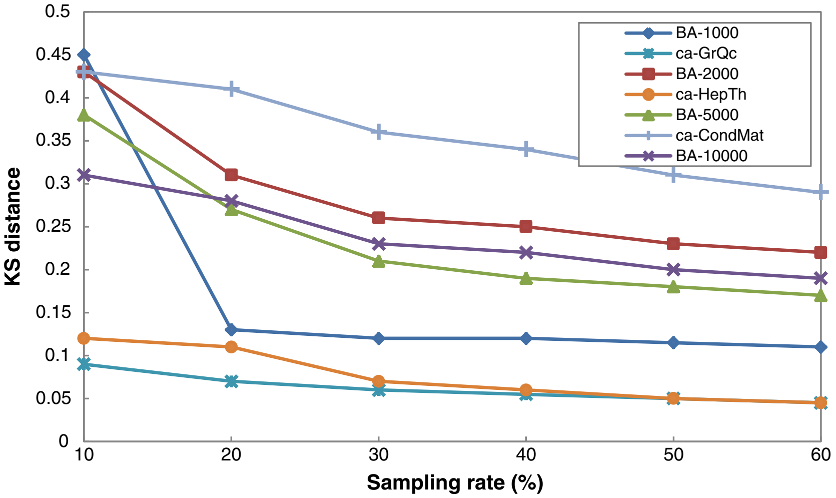


Figure 11. Kolmogorov–Smirnov test versus different sampling rate for different test networks.

DOI: 10.1002/dac

of the plots for synthetic random networks, that is, RE for CC for small-world networks is less elastic to density of the input than RE for CC for synthetic random networks.

4.2.6. Experiment VI. This experiment is conducted to study the impact of varying sampling rates on the performance of the proposed sampling algorithm in terms of KS test distance for degree distribution. For this purpose, we plot KS test distance versus different sampling rates for each test network as given in Figure 11. From this figure, we can see that KS test distance decreases as the

Table XIII. Comparing DLAS and *e*DLAS in terms of KS test for degree distribution for sampling rates 10% and 20%.

	Sampling rate = 10%				Sampling rate = 20%			
Methods Networks	DLAS	*e*DLAS	Different significance	Performance	DLAS	*e*DLAS	Different significance	Performance
BA-1000	0.59	0.45	1.91E-27	+	0.41	0.13	6.53E-44	+
BA-2000	0.57	0.43	2.00E-19	+	0.30	0.31	2.47E-01	~
BA-5000	0.52	0.38	5.51E-17	+	0.41	0.37	1.84E-03	+
BA-10000	0.49	0.31	2.82E-26	+	0.46	0.28	2.82E-26	+
ca-GrQc	0.24	0.09	3.36E-33	+	0.12	0.07	1.68E-11	+
ca-HepTh	0.15	0.12	4.16E-06	+	0.14	0.11	4.16E-06	+
ca-CondMat	0.49	0.43	6.12E-09	+	0.43	0.41	2.86E-02	+

DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling; Kolmogorov–Smirnov.

Table XIV. Comparing DLAS and *e*DLAS in terms RE for clustering coefficient for sampling rates 10% and 20%.

	Sampling rate = 10%				Sampling rate = 20%			
Methods Networks	DLAS	*e*DLAS	Different significance	Performance	DLAS	*e*DLAS	Different significance	Performance
BA-1000	0.48	0.42	1.64E-11	+	0.47	0.34	8.31E-26	+
BA-2000	0.56	0.56	3.97E-01	~	0.45	0.43	6.25E-02	+
BA-5000	0.49	0.46	1.72E-02	+	0.42	0.41	2.75E-01	~
BA-10000	0.51	0.48	3.87E-03	+	0.44	0.43	2.29E-01	~
ca-GrQc	0.32	0.30	1.85E-03	+	0.29	0.23	2.67E-14	+
ca-HepTh	0.40	0.38	1.40E-03	+	0.34	0.35	8.71E-02	–
ca-CondMat	0.41	0.39	2.86E-02	+	0.38	0.36	2.86E-02	+

DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling; RE, relative error.

Table XV. Comparing DLAS and ***e***DLAS in terms of RE for degree distribution parameter for sampling rates 10% and 20%.

	Sampling rate = 10%				Sampling rate = 20%			
Methods Networks	DLAS	*e*DLAS	Different significance	Performance	DLAS	*e*DLAS	Different significance	Performance
BA-1000	0.09	0.05	1.07E-10	+	0.05	0.04	3.41E-02	+
BA-2000	0.05	0.04	1.55E-01	~	0.04	0.03	1.55E-01	~
BA-5000	0.08	0.06	2.39E-02	+	0.04	0.04	3.97E-01	~
BA-10000	0.10	0.11	1.33E-01	~	0.09	0.09	3.97E-01	~
ca-GrQc	0.14	0.16	1.98E-05	+	0.12	0.11	2.40E-02	+
ca-HepTh	0.08	0.01	1.90E-24	+	0.07	0.01	3.17E-21	+
ca-CondMat	0.12	0.12	3.97E-01	~	0.09	0.08	1.03E-01	~

DLAS, distributed learning automata-based sampling; *e*DLAS, distributed learning automata-based sampling; RE, relative error.

sampling rate increases, and also, KS test distance is higher for small networks. Among the tested networks, KS test distance for *ca-CondMat* is highest and for *ca-GrQc* is lowest.

4.2.7. Experiment VII. This experiment is conducted to compare the performance of the proposed sampling algorithm (*e*DLAS) and other learning automata-based sampling algorithm (DLAS) reported in literature [7]. For this purpose, we used *t*-test to compare the results of this comparison. In this test, the statistical results for comparing *e*DLAS and DLAS algorithms by the two-tailed *t*-test with 58 degrees of freedom at a 0.05 level of significance are reported. In Tables XIII, XIV, and XV, the performance of *e*DLAS versus DLAS is shown as '+', '−', and '~' when *e*DLAS is significantly better than, worse than, or similar to DLAS, respectively. Based on the test results, one may conclude that the process of traversal of the network with the help of *e*DLAS leads to better sampling.

5. CONCLUSION

A sampling algorithm based on *e*DLA for complex social networks was proposed in this paper and then eDLAS compared with some existing sampling algorithms including RES, RNS, RWS, MHRW sampling, and the only reported learning automata-based sampling algorithm in the literature called DLAS in terms of RE for clustering coefficient and RE for degree distribution parameter and KS test for degree distribution. The results of experimentation showed the superiority of the proposed sampling algorithm.

REFERENCES

1. Yang K, Cheng X, Hu L, Zhang J. Mobile social networks: state-of-the-art and a new vision. *International Journal of Communication Systems* 2012; **25**(10): 1245–1259.
2. Liu B-H, Hsu Y-P, Ke W-C. Virus infection control in online social networks based on probabilistic communities. *International Journal of Communication Systems* 2014; **27**(12): 4481–4491.
3. Liaqat HB, Xia F, Yang Q, Yang Q, Xu Z, Ahmed AM, Rahim A. Bio-inspired packet dropping for ad-hoc social networks. *International Journal of Communication Systems* 2015. DOI:10.1002/dac.2857. (in-press)
4. Li L, Lin X, Zhai Y, Yuan C, Zhou Y, Qi J. User communities and contents co-ranking for user-generated content quality evaluation in social networks. *International Journal of Communication Systems* 2015. DOI:10.1002/dac.2908 (in-press).
5. Watts DJ, Strogatz SH. Collective dynamics of "small-world" networks. *Nature* 1998; **393**(6684): 440–442.
6. Barabási AL, Albert R. Emergence of scaling in random networks. *Science* 1999; **286**(5439): 509–512.
7. Rezvanian A, Rahmati M, Meybodi MR. Sampling from complex networks using distributed learning automata. *Physica A: Statistical Mechanics and its Applications* 2014; **396**: 224–234.
8. Volz E, Heckathorn DD. Probability based estimation theory for respondent driven sampling. *Journal of Official Statistics-Stockholm* 2008; **24**(1): 79.
9. Gjoka M, Butts CT, Kurant M, Markopoulou A. Multigraph sampling of online social networks. *IEEE Journal on Selected Areas in Communications* 2011; **29**(9): 1893–1905.
10. Papagelis M, Das G, Koudas N. Sampling online social networks. *IEEE Transactions on Knowledge and Data Engineering* 2013; **25**(3): 662–676.
11. Kurant M, Markopoulou A, Thiran P. Towards unbiased BFS sampling. *IEEE Journal on Selected Areas in Communications* 2011; **29**(9): 1799–1809.
12. Murai F, Ribeiro B, Towsley D, Wang P. On set size distribution estimation and the characterization of large networks via sampling. *IEEE Journal on Selected Areas in Communications* 2013; **31**(6): 1017–1025.
13. Leskovec J, Faloutsos C. Sampling from large graphs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006; 631–636.
14. Beigy H, Meybodi MR. Utilizing distributed learning automata to solve stochastic shortest path problems. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems* 2006; **14**(5): 591–615.
15. Meybodi MRM, Meybodi MR. Extended distributed learning automata. *Applied Intelligence* 2014; **41**(3): 923–940.
16. Yoon S, Lee S, Yook SH, Kim Y. Statistical properties of sampled networks by random walks. *Physical Review E* 2007; **75**(4): 046114.
17. Lee CH, Xu X, Eun DY. Beyond random walk and Metropolis–Hastings samplers: why you should not backtrack for unbiased graph sampling. *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, 2012; 319–330.
18. Lee SH, Kim PJ, Jeong H. Statistical properties of sampled networks. *Physical Review E* 2006; **73**(1): 016102.
19. Even S. *Graph Algorithms*, Cambridge University Press: New York, 2011.
20. Kurant M, Markopoulou A, Thiran P. On the bias of BFS (breadth first search). *2010 22nd International Teletraffic Congress (ITC)*, 2010, 1–8.

DOI: 10.1002/dac

21. Frank O. Survey sampling in networks. In *The SAGE Handbook of Social Network Analysis*, Scott J. and Carrington P. J. (eds). SAGE publications: London, 2011; 370–388.
22. Lovász L. Random walks on graphs: a survey. *Combinatorics, Paul Erdos is Eighty* 1993; **2**(1): 1–46.
23. Kurant M, Gjoka M, Butts CT, Markopoulou A. Walking on a graph with a magnifying glass. *Proceedings of ACM SIGMETRICS*, 2011; 1–12.
24. Goel S, Salganik MJ. Assessing respondent-driven sampling. *Proceedings of the National Academy of Sciences* 2010; **107**(15): 6743–6747.
25. Kim P-J, Jeong H. Reliability of rank order in sampled networks. *The European Physical Journal B* 2007; **55**(1): 109–114.
26. Gjoka M, Kurant M, Butts CT, Markopoulou A. Walking in Facebook: a case study of unbiased sampling of OSNs. *2010 Proceedings IEEE INFOCOM*, 2010; 1–9.
27. Ribeiro B, Towsley D. Estimating and sampling graphs with multidimensional random walks. *Proceedings of the 10th annual conference on Internet measurement*, 2010; 390–403.
28. Maiya AS, Berger-Wolf TY. Sampling community structure. *Proceedings of the 19th international conference on World wide web*, 2010; 701–710.
29. Rejaie R, Torkjazi M, Valafar M, Willinger W. Sizing up online social networks. *IEEE Network* 2010; **24**(5): 32–37.
30. Gile KJ, Handcock MS. Respondent-driven sampling: an assessment of current methodology. *Sociological Methodology* 2010; **40**(1): 285–327.
31. Salehi M, Rabiee HR, Nabavi N, Pooya S. Characterizing Twitter with respondent-driven sampling. *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2011; 1211–1217.
32. Jin L, Chen Y, Hui P, Ding C, Wang T, Vasilakos AV, Deng B, Li X. Albatross sampling: robust and effective hybrid vertex sampling for social graphs. *Proceedings of the 3rd ACM international workshop on MobiArch*, 2011; 11–16.
33. Wang J, Guo Y. Unbiased sampling of bipartite graph. *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2011; 357–360.
34. Cooper C, Radzik T, Siantos Y. A fast algorithm to find all high degree vertices in power law graphs. *Proceedings of the 21st international conference companion on World Wide Web*, 2012; 1007–1016.
35. Lu J, Li D. Sampling online social networks by random walk. *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, 2012; 33–40.
36. Salehi M, Rabiee HR, Rajabi A. Sampling from complex networks with high community structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 2012; **22**(2): 023126–023126.
37. Park H, Moon S. Sampling bias in user attribute estimation of OSNs. *Proceedings of the 22nd international conference on World Wide Web companion*, 2013; 183–184.
38. Lu X, Bressan S. Sampling connected induced subgraphs uniformly at random. *Scientific and Statistical Database Management* 2012; **7338**: 195–212.
39. Ribeiro B, Wang P, Murai F, Towsley D. Sampling directed graphs with random walks. *Proceedings IEEE INFOCOM,* 2012; 1692–1700.
40. Yang C-L, Kung P-H, Chen C-A, Lin S-D. Semantically sampling in heterogeneous social networks. *Proceedings of the 22nd international conference on World Wide Web companion*, 2013, 181–182.
41. Gao Q, Ding X, Pan F, Li W. An improved sampling method of complex network. *International Journal of Modern Physics C* 2014; **25**(5): 1440007.
42. Tong C, Niu J, Xie Z, Peng F. Sampling from social network to maintain community structure. *International Journal of Communication Systems* 2014; **27**(9): 1363–1377.
43. Narendra KS, Thathachar MAL. *Learning Automata: An Introduction*, Printice-Hall: New York, 1989.
44. Safavi SM, Meybodi MR, Esnaashari M. Learning automata based face-aware Mobicast. *Wireless Personal Communications* 2014; **77**(3): 1923–1933.
45. Misra S, Chatterjee SS, Guizani M. Stochastic learning automata-based channel selection in cognitive radio/dynamic spectrum access for WiMAX networks. *International Journal of Communication Systems* 2015; **28**(5): 801–817.
46. Krishna PV, Misra S, Joshi D, Gupta A, Obaidat MS. Secure socket layer certificate verification: a learning automata approach. *Security and Communication Networks* 2014; **17**(11): 1712–1718.
47. Kumar N, Lee J-H. Collaborative-learning-automata-based channel assignment with topology preservation for wireless mesh networks under QoS constraints. *IEEE Systems Journal* 2015; **9**(3): 675–685.
48. Kumar N, Lee J-H, Rodrigues JJ. Intelligent mobile video surveillance system as a Bayesian coalition game in vehicular sensor networks: learning automata approach. *IEEE Transactions on Intelligent Transportation Systems* 2015; **16**(3): 1148–1161.
49. Kumar N, Misra S, Obaidat MS, Rodrigues J, Pati B. Networks of learning automata for the vehicular environment: a performance analysis study. *IEEE Wireless Communications* 2014; **21**(6): 41–47.
50. Kumar N, Misra S, Obaidat MS. Collaborative learning automata-based routing for rescue operations in dense urban regions using vehicular sensor networks. *IEEE Systems Journal* 2015; **9**(3): 1081–1090.
51. Saghiri AM, Meybodi MR. A distributed adaptive landmark clustering algorithm based on mOverlay and learning automata for topology mismatch problem in unstructured peer-to-peer networks. *International Journal of Communication Systems* 2015. DOI:10.1002/dac.2977 (in-press).
52. Polatoglou M, Nicopolitidis P, Papadimitriou GI. On low-complexity adaptive wireless push-based data broadcasting. *International Journal of Communication Systems* 2014; **27**(1): 194–200.
53. Nicopolitidis P, Chrysostomou C, Papadimitriou GI, Pitsillides A, Pomportsis AS. On the efficient use of multiple channels by single-receiver clients in wireless data broadcasting. *International Journal of Communication Systems* 2014; **27**(3): 513–520.

54. Nicopolitidis P. Performance fairness across multiple applications in wireless push systems. *International Journal of Communication Systems* 2015; **28**(1): 161–166.
55. Misra S, Krishna PV, Saritha V, Obaidat MS. Learning automata as a utility for power management in smart grids. *IEEE Communications Magazine* 2013; **51**(1): 98–104.
56. Hasanzadeh M, Meybodi MR. Distributed optimization grid resource discovery. *The Journal of Supercomputing* 2015; **71**(1): 87–120.
57. Morshedlou H, Meybodi MR. Decreasing impact of SLA violations: a proactive resource allocation approach for cloud computing environments. *IEEE Transactions on Cloud Computing* 2014; **2**(2): 156–167.
58. Thathachar M. Learning automata with changing number of actions. *IEEE Transactions on Systems, Man, and Cybernetics* 1987; **17**(6): 1095–1100.
59. Alipour MM. A learning automata based algorithm for solving capacitated vehicle routing problem. *International Journal of Computer Science Issues* 2012; **9**(2): 138–145.
60. Akbari Torkestani J, Meybodi MR. Finding minimum weight connected dominating set in stochastic graph based on learning automata. *Information Sciences* 2012; **200**(1): 57–77.
61. Rezvanian A, Meybodi MR. Finding maximum clique in stochastic graphs using distributed learning automata. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 2015; **23**(1): 1–31.
62. Soleimani-Pouri M, Rezvanian A, Meybodi MR. Solving maximum clique problem in stochastic graphs using learning automata. *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, 2012; 115–119.
63. Rezvanian A, Meybodi MR. Finding minimum vertex covering in stochastic graphs: a learning automata approach. *Cybernetics and Systems* 2015; **46**(8): 698–727.
64. Akbari Torkestani J, Meybodi MR. Clustering the wireless Ad Hoc networks: a distributed learning automata approach. *Journal of Parallel and Distributed Computing* 2010; **70**(4): 394–405.
65. Forsati R, Meybodi MR. Effective page recommendation algorithms based on distributed learning automata and weighted association rules. *Expert Systems with Applications* 2010; **37**(2): 1316–1330.
66. Amiri F, Yazdani N, Faili H, Rezvanian A. A novel community detection algorithm for privacy preservation in social networks. *Intelligent Informatics* 2013; **18**: 443–450.
67. Hasanzadeh M, Meybodi MR. Grid resource discovery based on distributed learning automata. *Computing* 2014; **96**(9): 909–922.
68. Goldstein ML, Morris SA, Yen GG. Problems with fitting to the power-law distribution. *The European Physical Journal B-Condensed Matter and Complex Systems* 2004; **41**(2): 255–258.

DOI: 10.1002/dac