Journal of AI and Data Mining
Vol 8, No 1, 2020, 127-148.
DOI: 10.22044/JADM.2019.7145.1842

Using an Evaluator Fixed Structure Learning Automata in Sampling of Social Networks

S. Roohollahi[1], A. Khatibi Bardsiri[2*] and F. Keynia[3]

1. Computer Engineering Department, Kerman Branch, Islamic Azad University, Kerman, Iran.
2. Computer Engineering Department, Kerman Branch, Islamic Azad University, Kerman, Iran.
3. Department of Energy Management and Optimization, Institute of Science and High Technology and Environmental Sciences; Graduate University of Advanced Technology, Kerman, Iran.

Received 06 June 2018; Revised 16 April 2019; Accepted 06 October 2019
**Corresponding author: a.khatibi@srbiau.ac.ir (AK. Bardsiri).*

Abstract
Social networks are streaming, and diverse including a wide range of edges so that they are continuously evolved over time and are formed by the activities such as tweets and emails among the users; each activity, adds an edge to the network graph. Despite their popularity, the dynamically and large size of most social networks make it difficult or impossible to study the entire networks. This paper proposes a sampling algorithm that is equipped with an evaluator unit for analyzing the edges and a set of simple fixed structure learning automata. The evaluator unit evaluates each edge and then decides whether the edge and the corresponding node should be added to the sample set. In the proposed algorithm, each main activity graph node is equipped with a simple learning automaton. The developed algorithm is compared with the best current sampling algorithm reported in the Kolmogorov-Smirnov test, and the normalized L1 and L2 distances in real networks and synthetic networks are presented as a sequence of edges. The experimental results obtained show the superiority of the proposed algorithm.

Keywords: *Evaluator Unit, Social networks, Network Sampling, Streaming Sampling, Fix Learning Automata.*

1. Introduction

With a history of over seventy years, social network analysis is an interdisciplinary subject used in various sciences such as sociology, economics, communication sciences, psychology, physics, and computers. At the present time, the social networks are another means of communication in regular social life patterns that are evident from the popularity of the social networks such as Facebook [1]. Therefore, the analysis of these networks is increasingly important for fraud detection [2], discovering interactive patterns among individuals, studying structural characteristics, evolution of networks over time. Despite their popularity and inclusiveness, the continuing dynamics and large size of most social networks makes it difficult or even impossible to study the networks.

For this reason, it is necessary to sample a small version of sub-graphs from the original network to be used for analyzing a larger network. Sampling algorithms should select sample sub-graphs that have similar properties to the original graphs. In [3], several algorithms have been proposed for checking the quality of sampling algorithms and a set of empirical rules for increasing sample measurements, for estimating the original graph [4]; nine different sampling techniques were evaluated in retrieving the underlying structural features of the social networks; they looked at four of the most favorite features, including the degree, clustering coefficient, betweenness centrality, and closeness centrality. The authors in [5, 6] have investigated the statistical properties of the samples taken from scale-free networks with three sampling algorithms: node, edge, and random walk sampling. By sampling social networks, most works [7-12] assume the network graph as a moderate size and static structure, and only focus on producing the samples that are suitable for graph attributes. However, these assumptions are not very

suitable for many real-world networks. For example, social activity networks include communications between users (such as wall posts, tweets, and emails), while any activity between two users results in an addition of an edge to the network graph. For this reason, these networks are streaming and have plenty of edges.

A streaming graph is considered to be a stream of edges that continuously evolves over time and is clearly too large to fit in the memory [13]. The traditional sampling algorithms cannot be used for streaming graphs. When the main network has too many edges in the main memory, sampling can only be done continuously (one edge at a time), due to the random access to the disk incur large input / output costs. In the static domain, a topology-based sampling method requires a random exploration of the neighboring nodes. (In case of sequential access, it requires many passes over the edge stream.). The node sampling method also requires a random access to the node set of the network graph. Therefore, none of these methods are suitable for sampling such a large-scale network. In addition, in some cases, it is necessary to analyze a dynamic network over time for many reasons, such as reviewing social structures over time and discovering interactive patterns among individuals. In these cases, the static sampling algorithms are not suited because they are never able to update the sampled sub-graph using edges that occur over time. Therefore, several snapshots at different points of time should be taken from the main network and for each snapshot; the sampling process must be fully restarted so that the sample is updated from that point of time.

As a result, sampling algorithms that can consider the complexity of the streaming domain are necessary. There is a lot of research works on the graph streams, [14-20]., Only a small amount of research works [13, 21-23] focuses on sampling representative sub-graphs from the streaming graphs. All of these streaming sampling algorithms run in a single pass over the stream, and take into consideration both the stream evolution and the massive size of the networks. An algorithm called partially-induced edge sampling (PIES) [13] has been tested on several real data sets, and its superiority has been shown over other algorithms. Although learning automata has been successful in many dynamic environments, in the field of graph sampling, another algorithm called fixed structure learning automata-based sampling algorithm (FLAS) [8], has been one of the most successful algorithms, and to our knowledge, it is the best streaming sampling method reported so far.

1.2. Motivation

For analysis of social networks, various parameters and indicators are designed and used. The degree of interaction of each node with other nodes, the difference or similarity of the geographical distribution of the nodes and their digital distribution, the depth and severity of the effect of each node's behavior on other nodes, the position of each node in the center, or the extent of its distance from the network center, the one-way or two-way interaction of the node with other nodes, the variety of information and communication between the nodes, and the entropy in the social network, are hundreds of parameters and criteria that are considered in the analysis of social networks. While analyzing the performance of two well-known algorithms, we will review their weaknesses. The PIES algorithm has two major disadvantages: (1) independent sampling from each streaming edges. Thus, the sampled sub-graphs are less likely to preserve connectivity and clustering of the original graph. Using the concept of a partial graph induction (sampling some edges occurring in sample nodes) can strengthen the algorithm to improve some of the main connections, (2) the incident nodes of the sampled edges are replaced with random selected nodes in the sample set. Therefore, nodes with more activity can be replaced despite the existence of some less activities or even isolated nodes in the sample. Despite the PIES's drawbacks, it can act well for sparse graphs and the performance of the algorithm decreases while the graph becomes denser and more clustered. This is while online social networks tend to depict high levels of clustering, and it also has been observed that the density of these networks increases over time.

The FLAS algorithm, also aims to overcome the PIES drawbacks and produce sample sub-graphs with high quality for dense and highly clustered graphs. Its major disadvantages are: (1) excessive focus on degree of centrality and neglecting other features in the main graph, (2) inability to implement the concept of a partial graph induction. Our proposed algorithm intent to develop a streaming sampling algorithm based on an pre-processing process embedded in an evaluator unit and utilize a simple fix structure learning automata, meanwhile maintaining the advantages of PIES and FLAS (such as running in a single pass over the stream and considering the stream evolution), can overcome their drawbacks and produce sample sub-graphs with a high quality and higher rate of preserving main graph attributes for dense and highly clustered graphs.

1.3. Our contribution

Our contributions in this paper are as follows:

We propose a learning automata-based streaming sampling algorithm that is integrated with an evaluation unit, called Evaluator Fixed Automata Sampling (EFAS), which runs in a single pass over the edge stream and maintains a dynamic sample, while the original graph is streaming. Each node of the original activity graph is equipped with a fixed structure learning automata and the fact that whether its corresponding node should to be added to the sample sub-graph or not is dependent on the output of the unit evaluator. We propose an evaluation unit that evaluates the edges based on the programmable parameters. The importance of sampling from each edge depends on the output of the evaluator unit. Using the evaluation unit as a pre-processing phase and then making use of the learning automata, help our proposed algorithm EFAS to overcome the drawbacks of the PIES [13] and FLAS [8] algorithms (as the best streaming sampling method reported so far) and produce more connected sample sub-graphs with higher rates of preserving a main graph attribute.

We conduct various experiments on the real-world and synthetic networks. It is shown that our proposed EFAS algorithm is better in terms of the quality of the sampled sub-graphs, and competitive in retaining attributes as compared to the FLAS and PIES algorithms. The quality of the developed algorithm is tested in terms of the Kolmogorov-Smirnov (KS) test for degree, clustering coefficient, k–core, and path length distributions and also in terms of normalized L1 and L2 distances, respectively, for eigenvalues and network values.

The results of experiments show, that EFAS obtains an improvement of 9% for degree distribution, 16% for clustering coefficient distribution, 18% for k–core distribution, 5% for path length distribution, 29% for the eigenvalues, and 26% of network values as compared to FLAS for an input graph G (V, E) (where V is the set of nodes and E is the set of edges)., As before, FLAS obtains an improvement of about 66% for degree distribution, 69% for clustering coefficient distribution, 64% for k–core distribution, 48% for path length distribution, 78% for the eigenvalues, and 76% of network values as compared to PIES.

2. Streaming graph sampling

The notion of having a static graph with a size that can be stored in memory is often not consistent with the reality. In fact, today we are dealing with an activity network such as Facebook wall, Twitter posts, and Email communications. The users interact repeatedly over time, the network graph is dynamic, and the stream of edges is considered to be continuously time-varying and include too many edges. Therefore, in this situation, the traditional sampling techniques are not appropriate. Due to the importance of studying such large-scale dynamic networks, the researchers use sampling techniques for sampling representative sub-graphs of these streaming graphs. Authors in [23] have utilized a min-wise hash functions to sample almost uniformly from the set of all edges that have been at any time in the stream graph. The sampled edges were used later to maintain the cascaded summaries of the stream. The authors in [21] have proposed a reservoir sampling method based on min-wise hash sampling of the edges in order to maintain the structural summaries of the underlying graph. These structural summaries are designed to create dynamic and efficient models for detecting outlier in graph streams. The authors in [22] have developed a time-based sampling technique for sampling from activity graphs presented as a sequence of edges ordered over time. This method randomly selects a timestamp on the activity timeline of the graph and samples the nodes incident on edges that have occurred in a time window starting from that timestamp. The algorithm repeats this process in a streaming fashion until the required fraction of nodes is collected. Finally, the sample set includes the selected nodes and any future edges that involve these nodes. Others have compared their method with the traditional sampling algorithms such as node sampling and Forest Fire is sampling. The authors in [13] have dealt with the graph sampling problem by outlining a spectrum of computational models for sampling algorithms, ranging from the simplest model based on the assumption of static graphs to the more challenging model of sampling from graph streams. They have proposed several sampling algorithms based on the concept of graph induction generalized across the spectrum from static to streaming. In static domain, they proposed a sampling algorithm called induced edge sampling (ES-i), which was a combination of edge-based node sampling method and the graph induction. The authors proved the better performance of their proposed algorithm ES-i by comparing it with the traditional sampling algorithms. In streaming domain, the authors in [13] have addressed the massive size of edges (that is too large to fit in memory) and continuously evolving edge stream over time, and adapted static sampling algorithms for streaming graphs. They presented streaming variations of node, edge, and topology-based sampling, as well as a streaming variation in the

algorithm ES-I, referred to as the partial-induced edge sampling (PIES), which all run in a single pass over the stream of edges. As reported in [13], PIES preserves more accurately the underlying properties of the testing datasets compared to the other streaming algorithms. This algorithm is biased to high-degree nodes and provides a dynamic sample, while the original graph is streaming. The algorithm receives as input a streaming graph presented as an arbitrarily ordered sequence of edges and adds the first m edges incident to n_s nodes to the sample set. Then, it scans the rest of the stream and randomly samples edges such that any streaming edge is sampled, if at least one of two nodes incident to that edge does not belong to the sample set, and otherwise, that edge is sampled with the probability γ=1. For any sampled edge, the incident nodes replace the former sampled nodes chosen uniformly at random. As noted in [13], PIES achieves a better result for graphs that are sparse and less clustered. PIES has two major disadvantages: (1) independent sampling from each streaming edges, (2) the incident nodes of sampled edges are replaced with randomly selected nodes in the sample set. Therefore, it may that nodes with more activity are replaced despite the existence of some less activity or even isolated nodes in the sample. In the streaming domain, Ghavipour et al. [8] have referred to a streaming sample algorithm for the social activity networks using fixed structure learning automata (FLAS), which runs in a single pass and overcomes the PIES drawbacks and produces sample sub-graphs with high quality for dense and highly clustered graphs. However, its major disadvantages are (1) excessive focus on degree centrality and neglecting other features in the main graph, (2) inability to implement the concept of a partial graph induction.

3. Learning automata

Learning automata (LA) [38, 39] is a reinforcement learning approach. The automaton has a finite set of actions, and it is limited to select one of them at each step. That action as the input to the environment and the environment react to a reinforcement signal with a probability. Based on these interactions between the environment and learning automata, the learning automaton slightly learns the optimal action, which leads to a minimum penalty probability.

The random environment is called a non-stationary environment, if the penalty probabilities vary over time, and it is said to be stationary, if the penalty probabilities are constant. The interaction between a learning automaton and its random environment is shown in figure 1.

The learning automata is used in many applications, such as graph sampling [7, 8], fuzzy membership function optimization [40], and vertex coloring [41]. LA can be divided into two basic groups: if its transition and output function changes over time, a learning automaton is called a variable structure learning automaton (VSLA); otherwise, it is a fixed structure learning automaton (FSLA). An FSLA is a quintuple $\langle \alpha, \beta, \phi, \mathrm{F}, \mathrm{G} \rangle$, , where:

$\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of the actions that the automaton chooses from.

β = {0, 1} is the automaton‘s set of inputs, where, if β = 1, it receives a penalty, and receives a reward otherwise;

$\phi = \{\phi_1, \phi_2, \ldots, \phi_{rN}\}$ indicates its set of states, where N is called the depth of memory of the automaton;

$\mathrm{F} : \phi \times \beta \to \phi$ illustrates the transition of the state of the automaton on receiving an input from the environment. £ can be stochastic.

$\mathrm{G} : \phi \to \alpha$ is the output function of the automaton. The action taken by the automaton is determined according to its current state. This means that the automaton selects action α_i if it is in any of the states $\{\phi_{(i-1)N+1}, \phi_{(i-1)N+2}, \ldots, \phi_{iN}\}$. The state $\phi_{(i-1)N+1}$ is considered to be the most internal state, and ϕ_{iN} is considered to be the boundary state of action α_i, indicating that the automaton has the most and the least certainty in performing the action α_i, respectively.

The action chosen by the automaton is applied to the environment, which leads to emit a reinforcement signal β. On the basis of the received signal β, the state of the automaton is updated, and then a new action is chosen according to the functions £ and ζ, respectively. There exist different types of FSLA based on the state transition function F and the output function G. J, $L_{2N,2}$, $G_{2N,2}$ and Krinsky are famous types of FSLA. All of the mentioned automatons can work on a variety of issues, depending on their performance, and using them, different observers with different behaviors can be implemented. FLAS [8] uses the $G_{2N,2}$ automaton with the best setting of the parameters obtained from a group of experiments (i.e. γ = 0.9 and N = 4). However, since in our proposed algorithm, only the past

behavior of the system will be of great importance for the current issue, there is no need for more complex automata, and so we will use the J automaton in the proposed algorithm with the same settings obtained in the FLAS algorithm.

The ability of learning in a learning automaton is determined by comparison with a pure-chance automaton. If there is no prior information from the environment, there is also no basis on which actions can be selected. In such a case, the automaton at random chooses its actions, i.e. by pure chance. The pure-chance automaton is a standard tool for comparison of learning automatons behavior, such that any automaton that is said to learn is expected to do at least better than a pure-chance automaton. A comparison can be made in terms of the average penalty $M(t)$ received by an automaton $M(t)$ for a pure-chance automaton is a constant value and is denoted by $M(0)$. For a learning automaton to do better than a pure chance automaton, the expected value of its average penalty $E[M(t)]$ must be less than $M(0)$ at least asymptotically as $t \to \infty$.

Definition 1. A learning automaton is expedient only if:

$$\lim_{t\to\infty} E[M(t)] < M(0) \qquad (1)$$

that means the average penalty received by an automaton decreases over time as a result of the increase in learning.

3.1. J automata

The J automaton, which we denote by $J(K, N)$, has KN states (i.e. : $\phi_1, \phi_2, \ldots, \phi_{KN}$) and K actions and attempts to incorporate the past behavior of the system in its decision rule for choosing the sequence of actions.

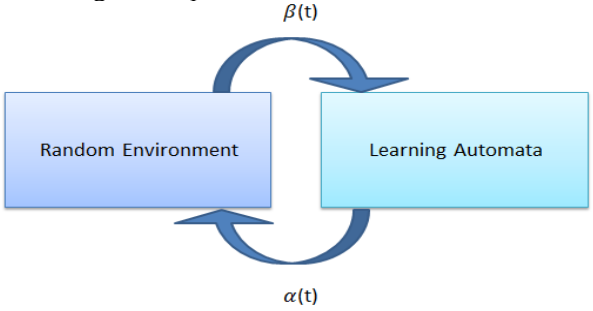

Figure 1. Learning automata and its relationship with environment.

The states with numbers $(k-1)N+1$ through KN correspond to action K. The state transition graph of these automata for a favorable condition and an unfavorable condition is shown in figure 2. As mentioned earlier, the state transition function F can be considered as a stochastic function. In such a case, on receiving a signal from the environment, the transition of the automaton among its states is not deterministic. For instance, when an action results in a reward, the automaton may move one state towards the boundary state with the probability $\gamma 1 \in [0, 1)$, and moves one state towards its most internal state with the probability $1-\gamma 1$, and reverse this procedure using probabilities $\gamma 2$ and $1-\gamma 2$ when an action results in a penalty. In some situations where rewarding a favorable action may be preferable more than penalizing an unfavorable action, one can set $\gamma 1 = 0$ and $\gamma 2 \in [0, 1)$. These settings result in state transitions of J automaton becoming deterministic when the automaton receives a reward ($\beta = 0$), as shown in figure 2. However, in the case of punishment, the J automaton will transit among its states stochastically, as shown in figure 3. Considering $\gamma 1 = 0$, the automaton will be expedient for all values of $\gamma 2$ in interval [0, 1).

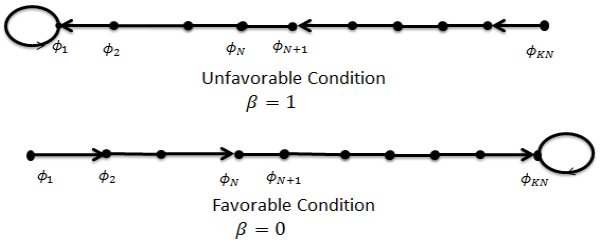

Figure 2. The state transition graph for J automaton.

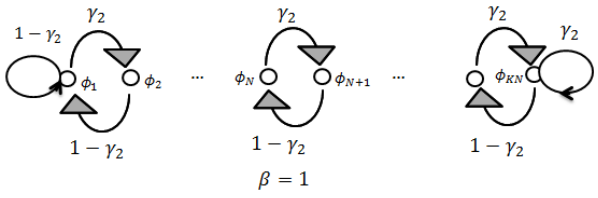

Figure 3. The state transition graph for J in case of punishment.

4. Proposed streaming sampling algorithm

In this section, a fixed structure learning automata-based sampling algorithm, called EFAS is presented for sampling from the streaming graphs. The inputs are an activity graph $G(V, E)$ presented as a sequence of edges in an arbitrary

order, as well as the sample size n_s, and the output of the algorithm is a representative sample $G_s(V_s, E_s)$ $(|V_s| = n_s)$ that will match the properties of the graph G. Figure 4 describes a global flow diagram of the proposed algorithm. Here, we describe the EFAS algorithm on the basis of the state transitions of the J automaton with the stochastic transition function, as depicted in figures 2, 3. The J automaton behaves deterministically when it is rewarded and stochastically when it receives a penalty, i.e. $\gamma 1 = 0$ and $\gamma 2 \in [0, 1)$. Since $\gamma 1$ is set to zero, in the rest of the paper, we ignore it and refer to $\gamma 2$ as γ.

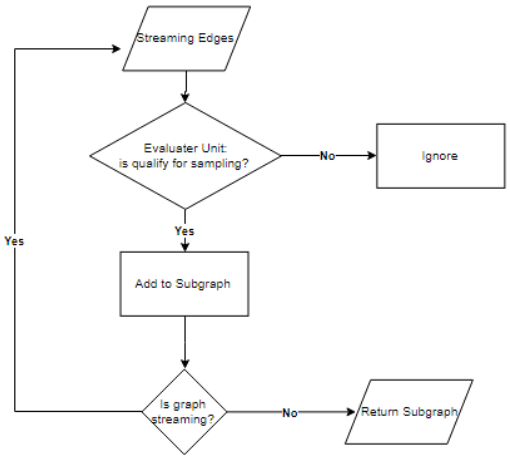

Figure 4. A global flow diagram of the proposed algorithm.

The schematic diagram and pseudo-code of the sampling algorithm EFAS are given, respectively, in figures 5, 6. EFAS consists of two parts. At first, an initial sample graph is created by successively adding the edges one by one from the beginning of the stream to E_s and their incident nodes to V_s until the required number of nodes is sampled ($|V_s| = n_s$). When a node V_k is visited for the first time, it is equipped with a fixed structure learning automaton LA_k. Initially, the states of learning automata corresponding to the nodes in V_s are set to the boundary state of its current action α_1 ($\forall V_k \in V_s$: the current state of LA_k is set to ϕ_1, i.e. $\phi^k = \phi_1$). In fact, the automation will increase its depth as soon as it receives the appropriate response from the environment, and thus learning is done over time.

At the second part, the algorithm consecutively processes the remaining edges of the stream and keeps on updating the sample as long as the graph is streaming. The decision whether to sample a streaming edge or not is taken based on a pre-processing result that embedded in the evaluator unit. The evaluator unit calculates the value of the edge depending on the automata located in its corresponding nodes;, the process of placement of new nodes in V_s , depends on factors such as the higher value of the new edge., In this case, the new edge will be added to the set E_s, and the new nodes corresponding to it will also be replaced by nodes of the V_s set. Note that deleting a node from the set V_s will always be penalized with the probability $\gamma_2 \in [0,1)$, in which case all the edges of that node will also be deleted from the E_s set; on the other insertion in the set, V_s will also have a definite reward. The placement operation is if the value of the edge seen is greater than the minimum of the edges of the E_s set. Most importantly in our proposed algorithm in comparison with FLAS, by observing each edge, learning will not be made solely by the history of the corresponding nodes of that edge, but the evaluator unit will evaluate the significance of the edge and the corresponding nodes.

4.1. Evaluation unit

4.1.1. Node evaluation

In this section, we evaluate the nodes based on the factor the depth of learning the automata reside in; in this way, each node will be able to find a higher value in its internal state.

$$\omega_{N.E} = \alpha_1 N_{\varphi_{current}} \tag{2}$$

where α_1 is the coefficient of importance of this weight and $N_{\varphi_{current}}$ of the current state of the automata.

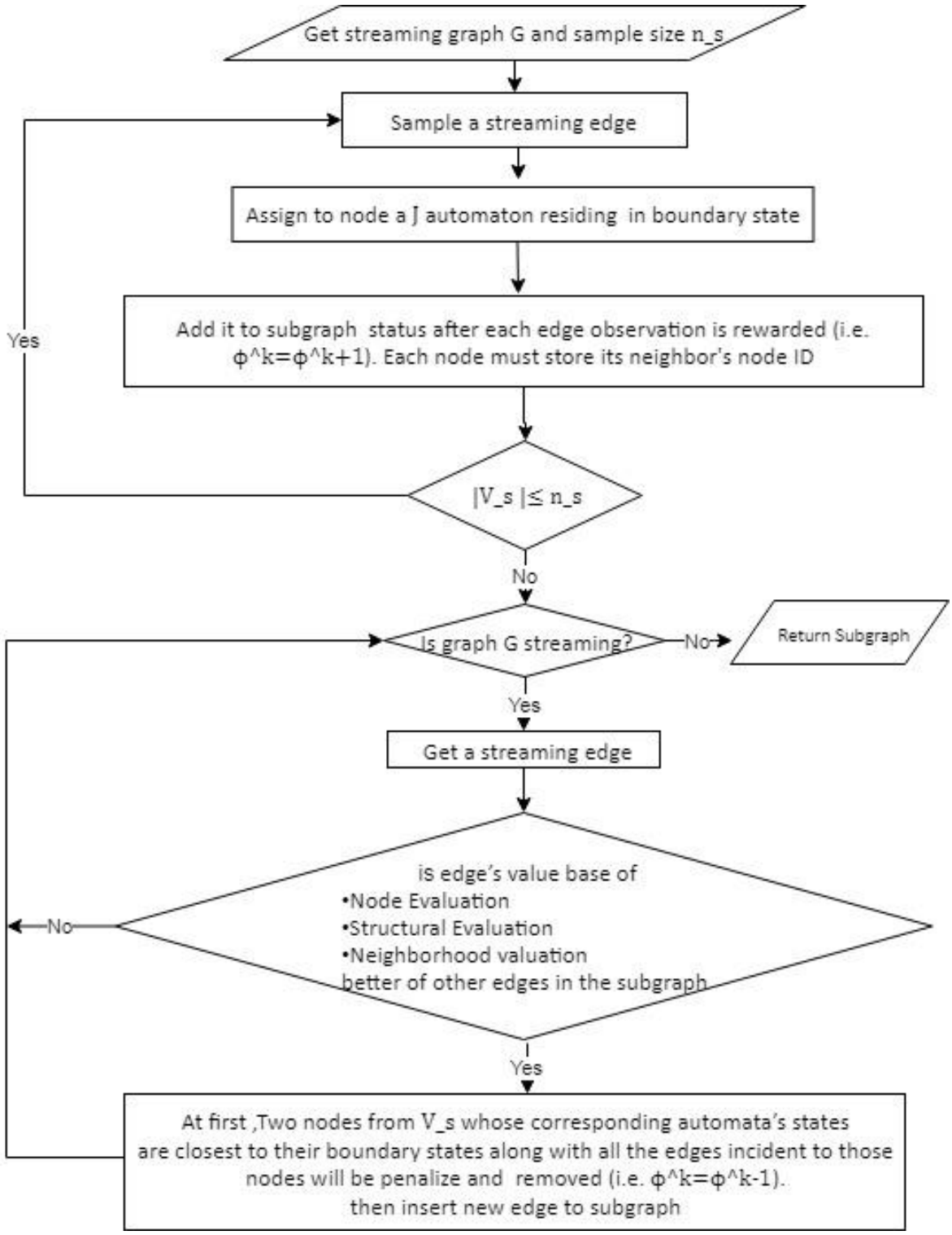

Figure 5. The schematic diagram of FLAS algorithm when L2N−2 automaton is used.

ALGORITHM EFAS Evaluator Fixed Automata Sampling

Input : *Original graph* $G(V,E)$, *Sample size* n_s ***Output*** : *Sampled sub graph* $G_s(V_s, E_s)$

First Phase

$V_s = \emptyset$, $E_s = \emptyset$, $Visited = \emptyset$, $y = 1$;

While $|V_s| < n_s$ ***do***

$e_y = Streaming\ Edge\ (v_i, v_j)$

$E_s = E_s \cup \{e_y\}$

for *each node* v_k *incident to* e_y ***do***

if $v_k \notin V_s$ ***then***

$V_s = V_s \cup \{v_k\}$;

Assign an J automaton (LA_k) *to* v_k , $\phi^K = \phi_1$;

else $\phi^K = \phi^K + 1$; // *reward*

end

$LA_k.visited = LA_k.visited \cup v_{j \neq k}$

end

y += 1;

End

Second Phase

while *graph is streaming* ***do***

$e_y = Streaming\ Edge\ (v_i, v_j)$

min= minimum edge in E_s

For *each node* v_k *incident to* e_y ***do***

$\phi^K = \phi^K + 1$ // *reward*

If *Evaluator* $(e_y) >$ *min* ***then***

$E_s = E_s \cup \{e_y\}$

for *each node* v_k *incident to* e_y ***do***

if $v_k \notin V_s$ ***then***

$V_s = V_s \cup \{v_k\}$;

If $LA_k.visited \cap V_s$ ***then*** $E_s = E_s \cup \{(LA_k.visited,, (LA_k.visited \cap V_s))\}$

Choose sampled node whose corresponding min ,penalize & delete it with all its incident edges and single node after that from V_s *with* $\gamma \in [0,1)$;

end

end

else

penalize node whose corresponding e_y *with* $\gamma \in [0,1)$;

end

Figure 6. The pseudo-code of EFAS algorithm.

4.1.2. Neighborhood evaluation

In this section, node is evaluated and weighed in terms of its neighborhoods structure. In that case, the neighborhood with more valuable nodes will give a higher weight.

$$\omega_{Neigh.E} = \alpha_2(N.Neighbor_{\varphi_{current}}) \quad (3)$$

where α_2 is the coefficient of importance of this weight.

4.1.3. Structural evaluation

In this section, and for simplicity, imagine an edge with two nodes A and B. The node is evaluated in terms of the sub-graph structure that has been evaluated so far. In this way, it is examined whether the neighboring set of A (other than node B) has any share with the neighbors of Node B. If there is a common node (nodes), they will find a higher weight.

$$\omega_{S.E} = \alpha_3(\{N.Neighbors\} \cap \{(N.Neighbor).Neighbors\}) \quad (4)$$

where α_3 is the coefficient of importance of this weight.

4.2. Describing EFAS algorithm

4.2.1. Initialing phase

When a node v_k is visited for the first time, it is equipped with a fixed structure learning automaton (the J automaton) LA_k. Initially, the states of learning automata corresponding to the nodes in V_s are set to the boundary state of its action. This status after each edge observation is rewarded (i.e. $\phi^k = \phi^k + 1$). Each node must store its neighbor's node ID. The state of the action of learning automaton indicates the strength of membership of the corresponding node.

The initial sample graph is created by successively adding the edges one by one from the beginning of the stream to E_s and their incident nodes to V_s until the required number of nodes is sampled. (i.e. $|V_s| = n_s$).

Each time the algorithm is confronted with an edge (an activity) $e_t = (v_i, v_j)$, if its nodes are both in the V_s set, both corresponding automatons LA_i and LA_j are rewarded (i.e. $\phi^k = \phi^k + 1$), and the edge e_t is added to E_s.

4.2.2. Updating phase

At this stage, after meeting the new edge, the evaluator unit determines the importance of the edge for sampling.

If the need for sampling is to be verified , in order to preserve the constraint (i.e. $|V_s| = n_s$), two nodes from V_s whose corresponding automata's states are closest to their boundary states along with all the edges incident to those nodes will be removed. The state of each one of the learning automata of the removed nodes receives penalty and then changes to its boundary state. The isolated nodes in the sub-graph will also be deleted in the same way.

If the need for sampling an edge $e_t = (v_i, v_j)$ is not verified, both corresponding automatons LA_i and LA_j receives penalty and moves one state towards the most boundary state with the probability $\gamma \in [0,1) \phi^k = \phi^k - 1$. In each stage, when an edge $e_t = (v_i, v_j)$ is added to the sub-graph, if each v_i or v_j has a history to a node in V_s set, that edge will be added to the sub-graph. (Using partial graph induction can help the algorithm to recover only some of the original connectivity.)

Using the fixed structure learning automata in our proposed algorithm in comparison with variable structure learning automata (VSLA), we obtain some advantages. The first advantage is that using EFAS reduces the computational cost of our algorithm. EFAS keeps an account of the number of rewards and penalties received for each action.

It continues performing whatever action it was earlier using as long as it is rewarded, and increase or decrease from one state to another with the result of the unit evaluator; the depth of learning is dependent on the death of memory N. However, VSLA chooses its action randomly based on the action probability vector kept over the action set. Therefore, it is probable that the automaton switches to another action even though it has received reward for doing the current action until the selection probability of an action converges to one. As a result, the sample sub-graph will incur many changes, and the complexity of our algorithm will increase. As the second advantage, different behavioral variations of an observer can be modeled by assigning an EFAS to each node of the graph. EFAS plays the role of an observer, which keeps track of the history of the activities of its node with other nodes in order to decide whether it is time to be included or omitted from the sample. Depending on the type of EFAS, different observers with different behaviors can be implemented. For example, when $L_{2N,2}$ is used, we expect a conservative behavior from the observer, whereas when $G_{2N,2}$ and J are used, we expect an optimistic behavior from the observer. However, because in our proposed algorithm, only the past behavior of the system will be of great importance for the current issue, there is no need for more complex automata, and so we will use the J automaton in the proposed algorithm. This possibility of modeling different behavioral variations does not exist when using VSLA. To In order to evaluate the performance of the proposed EFAS algorithm, several experiments are conducted on the real-world networks, described in table 1.

4.3. Discussion

In this section, we will discuss the advantages of the proposed algorithm against the two algorithms FLAS and PIES. Similar to the FLAS algorithm

developed in [8], the proposed EFAS algorithm scans the edge stream in a single pass, has a selection bias to high activity nodes, uses the concept of graph induction, and retains a dynamic sample while the graph G is streaming. The main difference between EFAS and the algorithm FLAS is in the way that the edges are sampled and their incident nodes are added to the sample set. In other words, the most prominent feature of EFAS is in the evaluation unit in a pre-processing role, so that it performs recognition and weighting of the edges for sampling. In EFAS, unlike FLAS, the value of an edge, does not relate solely to the observation history of the edge that leads to biased towards high degree nodes. In EFAS, there are three types of evaluations for weighting, and therefore, there is more probability to accurately retain the properties of the original graph. EFAS attempts to overcome their drawbacks using a simpler fixed structure learning automaton, which reduces the computational burden, and using the evaluator unit that will evaluate the significance of the edge and the corresponding nodes. In EFAS, the addition of an edge to the sample and then selection of nodes of V_s to be replaced by the nodes incident on that edge are done on the basis of the decisions made by the evaluator unit and learning automata. In fact, the learning automaton of each node somehow keeps track of the history of the activities of its node with other nodes and evaluator unit, then use for it, and two other structural features evaluate the significance of the edge. Once an edge is encountered, the evaluator unit decides whether the edge and (or) the incident nodes to be added to the sample or not. Therefore, the probability of being sampled of an edge depends on other edges that have already been observed from the edge's incident nodes, and any edge incident on the nodes with a high activity has a higher probability to be sampled than the edges incident on the nodes with a low activity. In other words, node itself, node neighborhood, and node structure have important roles in evaluating unit decision. In addition, for a sampled edge, the learning automata corresponding to the nodes in V_s determine which sampled nodes must be replaced by the incident nodes to the edge.

5. Experimental evaluation

In this section, we assess the efficiency of the proposed EFAS algorithm on several real-world networks. We utilize social networks from Flickr [31], a collaboration network from CondMAT, and a citation network from ArXiv HepPH [32].

5.1. Network statistics

The performance of a sampling algorithm is measured by determining how well the sub-graphs sampled by it match properties of the original graph. The statistics that we consider in our experiments are: degree, clustering coefficient k–core, and path length distributions, eigenvalues, and network values. We present a formal definition of these properties below:

Degree distribution The degree of a node in graph G is the number of connections or edges the node has to other nodes. The degree distribution $P(d)$ is then considered to be the fraction of nodes in the network with degree d>0. Thus, if there are n_d nodes in the network with degree d, we have:

$$P(d) = \frac{n_d}{|V|} \tag{5}$$

Clustering coefficient distribution The clustering coefficient for a node is defined as the proportion of links between the nodes within its neighborhood to the number of links that could possibly exist between them. The clustering coefficient distribution $P(c)$ is then calculated as:

$$P(c) = \frac{n_c}{|V'|} \ ,0 \le c \le 1 \tag{6}$$

where n_c illustrates the number of nodes in graph G with clustering coefficient c, and V' is the set of nodes with a degree greater than 1.

Table 1. Characteristics of datasets used

Dataset	Nodes	Edges	Density	Avg. Path	Global Clustering
Flickr	820878	66252280	1.9E-5	6.5	0.116
HepPH	34546	420877	7E-4	4.33	0.146
CondMAT	23133	93439	4E-4	5.35	0.264

Table 2. The results of statistical test for different types of algorithms in terms of KS distance for degree distribution.

Dataset	Test Results			
	Mean KS Distance (KS EFAS−KS PIES)	Difference. Sign.	Mean KS Distance (KS EFAS−KS FLAS)	Difference. Sign.
Flickr	-0.1381	2.4367E-81	-0.1	4.8765E-61
HepPH	-0.5099	3.8966E-21	-0.0722	2.7211E-26
CondMAT	-0.3951	6.7916E-57	-0. 0355	3.5643E-41

Table 3. The results of statistical test for different types of algorithms in terms of KS distance for clustering coefficient distribution.

Dataset	Test Results			
	Mean KS Distance (KS EFAS−KS PIES)	Difference. Sign.	Mean KS Distance (KS EFAS−KS FLAS)	Difference. Sign.
Flickr	-0.0585	3.1426E-63	-0.007	3.8396E-21
HepPH	-0.4581	1.7532E-10	-0.0124	2.8739E-9
CondMAT	-0.3301	2.8231E-57	-0. 0752	4.1654E-21

Table 4. The results of statistical test for different types of algorithms in terms of KS distance for path length distribution.

Dataset	Test Results			
	Mean KS Distance (KS EFAS−KS PIES)	Sign. Difference	Mean KS Distance (KS EFAS−KS FLAS)	Sign. Difference
Flickr	-0.3011	2.6574E-23	-0.1945	7.7345E-14
HepPH	-0.3241	1.2598E-8	-0.1552	4.6299E-5
CondMAT	-0.2933	4.9272E-17	-0. 1189	1.7123E-11

Table 5. KS distance for all datasets, at sampling fraction 0.2.

Dataset	EFAS				PIES				FLAS			
	Deg	Clus	K Core	Path	Deg	Clus	K Core	Path	Deg	Clus	K Core	Path
Flickr	0.0445	0.1013	0.1105	0.0018	0.1826	0.1598	0.2145	0.3029	0.1445	0.1087	0.1721	0.1963
HepPH	0.0091	0.0164	0.2041	0.0234	0.5190	0.4745	0.6641	0.3475	0.0813	0.0288	0.1697	0.1786
CondMAT	0.0365	0.0974	0.0619	0.0134	0.3816	0.4275	0.4829	0.3067	0.0720	0.1726	0.0858	0.1323
Avg.	0.0300	0.0717	0.1255	0.012	0.3610	0.3539	0.4538	0.3190	0.0992	0.1033	0.1425	0.1690

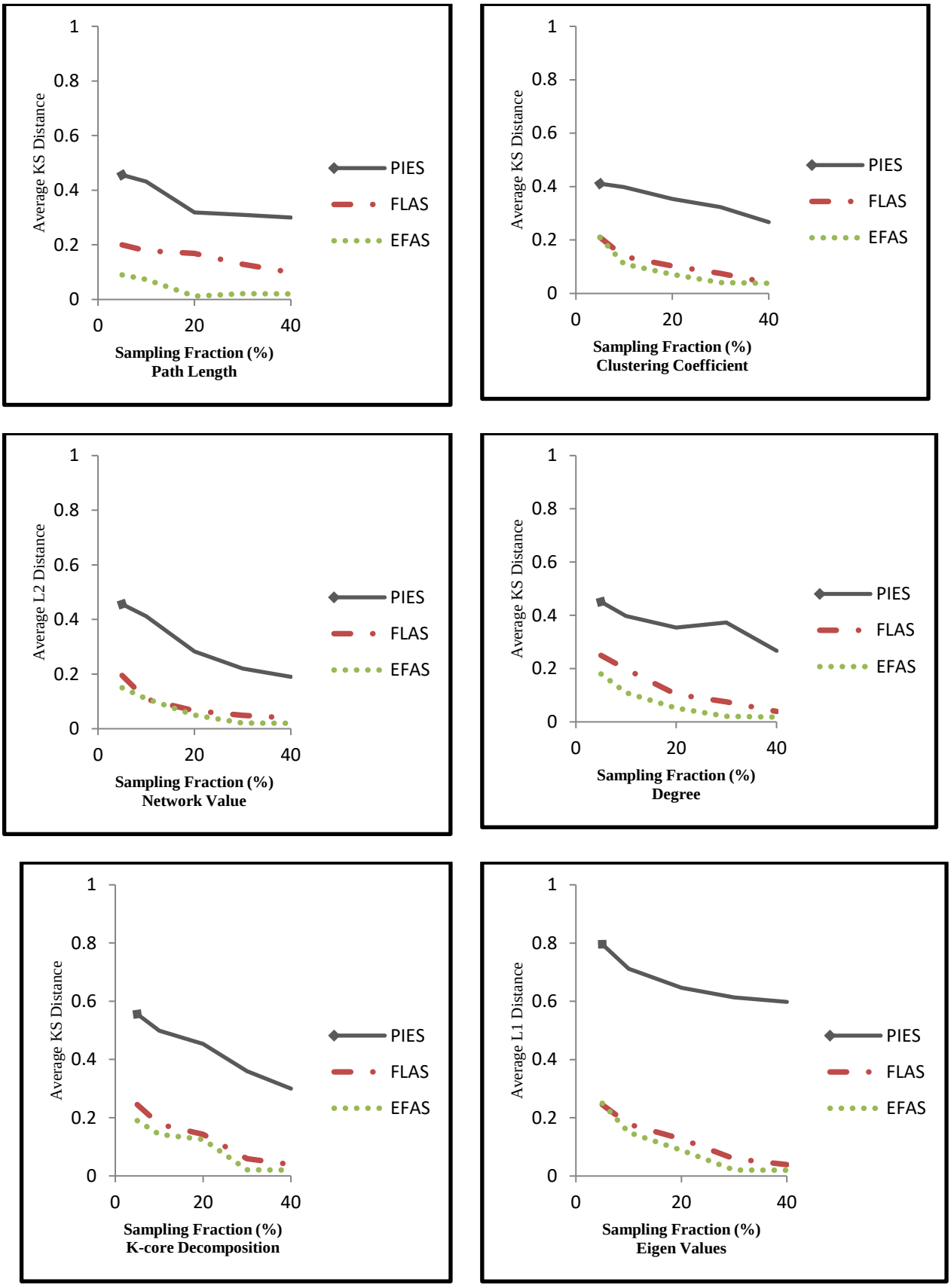

Figure 7. Average KS distance and average L1, L2 distance across 3 datasets.

K−core distribution The k–core of graph G is defined as the largest sub-graph of G in which all nodes have degree at least k . A node has coreness k if it belongs to the $k-core$ but not to $(k+1)-core$. The $k-core$ Distribution $P(k)$ is computed as the fraction of nodes having a coreness $k \geq 0$:

$$P(k) = \frac{n_k}{|V|} \tag{7}$$

where n_k denotes the number of nodes in graph G with coreness k.

Path length distribution The shortest path length denotes the fewest number of hops required to reach from a node to another node. The path length distribution $P(h)$ is considered to be the fraction of pairs of nodes in graph G with the shortest path length $h > 0$:

$$P(h) = \frac{n_h}{|V|^2} \tag{8}$$

where n_h is the number of pairs with shortest path length h .

Eigenvalues The eigenvalue λ of the adjacency matrix A of graph G is computed as

$$Av = \lambda v \tag{9}$$

Where v is the eigenvector of A associated with the eigenvalue λ. Eigenvalues are known as the basis of spectral graph analysis [13]. In the experiments, the largest 25 eigenvalues of a graph G are considered.

Network values Network values denotes the distribution of the eigenvector components corresponding to the largest eigenvalue of the adjacency matrix A of graph G [13]. In the experiments, the largest 100 network values of a graph G are considered.

Table 6. L1/L2 distance for all datasets, at sampling fraction 0.2.

Dataset	EFAS		PIES		FLAS	
	Eigen Val	Net Val	Eigen Val	Net Val	Eigen Val	Net Val
Flickr	0.0221	0.0421	0.5496	0.1923	0.0071	0.0013
HepPH	0.1061	0.0976	0.7151	0.3383	0.2147	0.1203
CondMAT	0.1371	0.0101	0.6743	0.3172	0.1605	0.0768
Avg.	0.0884	0.0499	0.6463	0.2826	0.1274	0.0661

5.2. Evaluation measures

In order to assess the distance between the statistic in the original graph and that in the sampled sub-graph, we use the following distance measures in this paper:

Kolmogorov-Smirnov (KS) statistic The KS statistic is commonly used for measuring the agreement between two cumulative distribution functions (CDFs) [46]. This statistic can be computed as the maximum vertical distance between two distributions:

$$ks = \max_x |F(x) - F'(x)| \tag{10}$$

where x is the range of the random variables, F and F′ denote two CDFs, and $0 \leq KS \leq 1$. In this paper, we use the KS- statistic for computing the distance between the true distribution of the original graph and the estimated distribution from the sampled sub-graph for the degree, clustering coefficient, k–core, and path length distributions.

Normalized L_1 distance The normalized L_1 distance (i.e. normalized manhattan distance) computes the distance between two positive m-dimensional real vectors [12].

$$L_1 = \frac{1}{m} \sum_{i=1}^{m} \frac{|p_i - p_i'|}{p_i} \tag{11}$$

where p and p′ are, respectively, the true vector and the estimated vector. In our experiment, we use the normalized L_1 distance to measure the distance between two vectors of eigenvalues from the original and the sampled graphs.

Normalized L_2 distance The normalized L_2 distance (i.e. normalized euclidean distance) measures the distance between two vectors when

the components of these vectors are fractions [13]. This distance measure is computed as:

$$L_2 = \frac{p - p'}{p} \quad (12)$$

In this paper, the normalized L_2 distance is utilized for computing the distance between two vectors of network values related to the original and the sampled graphs.

5.3. Experimental results

Although all the mentioned automatons depending on their performance can work on a variety of issues, since in our proposed algorithm, only the past behavior of the system will be of great importance for the current issue, there is no need for more complex automata, and so we will use the J automaton in the proposed algorithm with the same settings obtained in the FLAS algorithm (i.e. γ = 0.9 and N = 4). The proposed algorithm, considering the best setting for the parameters, is compared with the FLAS and PIES algorithms. In the experiments, the sampling fraction f varies from 0.1 to 0.3 with an increment of 0.025. For each sampling fraction, we report the average results of 30 independent runs. In each run, we utilize a random permutation of the edge stream as the input to the algorithm to make the results independent from any stream ordering. It is ensured that the different variations in the proposed algorithm and the FLAS and PIES algorithms use the same streaming orders.

Table 7. The maximum number of cores for sampling fraction 0.2 versus the actual value for each set.

Dataset	Real Max. core No.	EFSA	FLAS	PIES
Flickr	406	232	221	162
HepPH	30	21	20	5
CondMAT	25	19	16	6

5.3.1. Experiment 1

In the first experiment, the proposed EFAS algorithm is compared with the FLAS and PIES algorithms according to the KS distance for degree clustering and path length distribution. We perform the test with a 95% confidence interval and report the results for the sampling fraction of 0.2 in tables 2, 3, and 4. Based on the results in these tables, EFAS has a significant performance compared to PIES and FLAS.

5.3.2. Experiment 2

Investigate the ability of preserving the graph, statistics is one of the most important experiments for an algorithm in this field. Thus we assume the statistics such as the degree, clustering coefficient, k–core, path length distributions and report the average KS distance over three data sets for different sampling fractions, and plot the average L1 and L2 distances, respectively, for eigenvalues and for network values in figure 7. From these figures and the results in tables 5 and 6, we can see the superiority of the EFAS algorithm over the other algorithms for all the statistics. Finally, by computing the maximum core numbers (the maximum value of k in the k– core distribution) for sub-graphs sampled by EFAS and comparing those with the real maximum core number for each dataset, we investigate the ability of the EFAS algorithm to maintain the local density in the sampled sub-graphs with other algorithms. As shown in table 7, the sub-graphs that are sampled by EFAS have a maximum number of cores very close to the original graphs, while in FLAS and PIES, such a phenomenon does not happen.

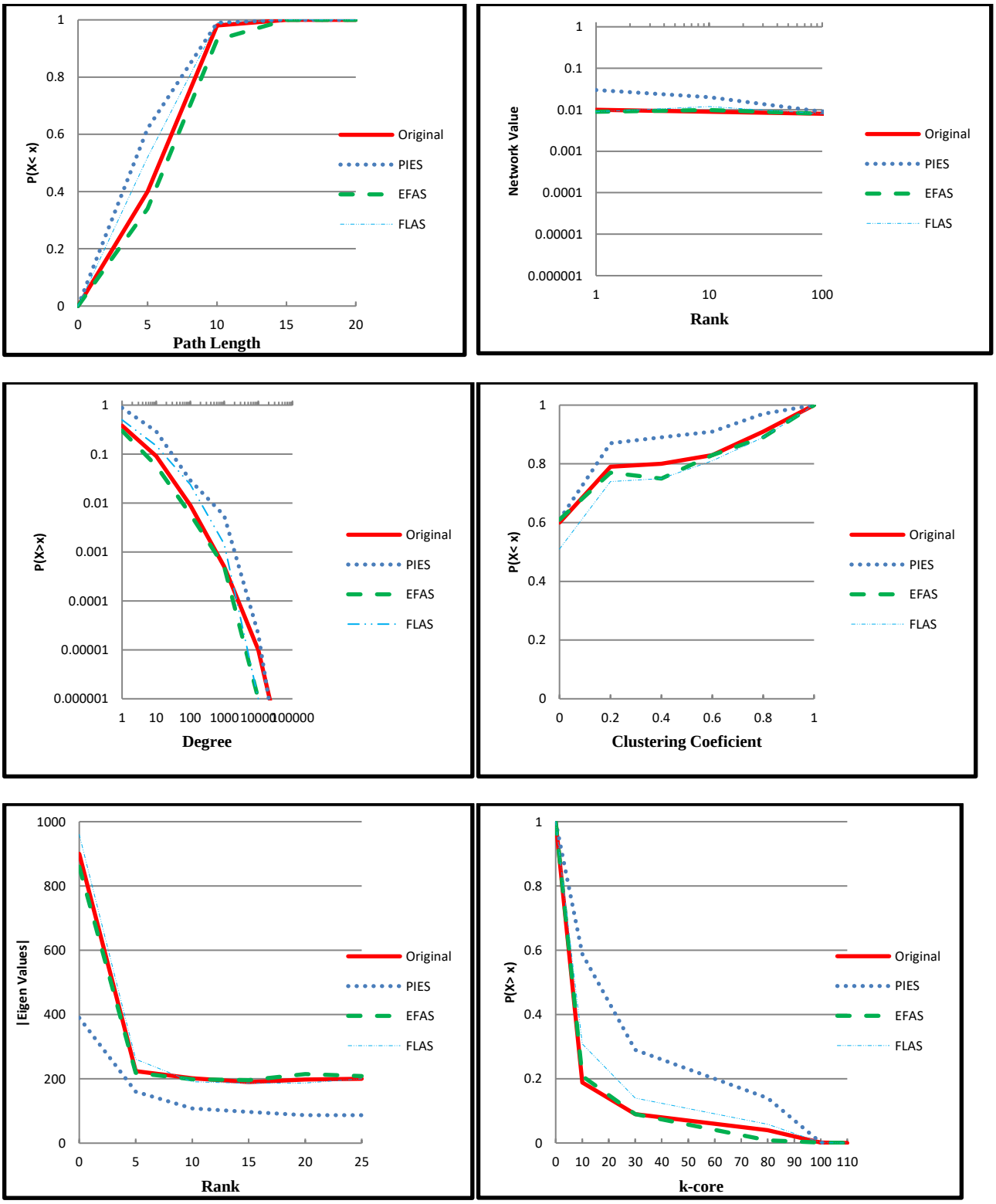

Figure 8. Comparison of sampling algorithms in the Flickr data set for different distribution statistics in the sampling section 0.2.

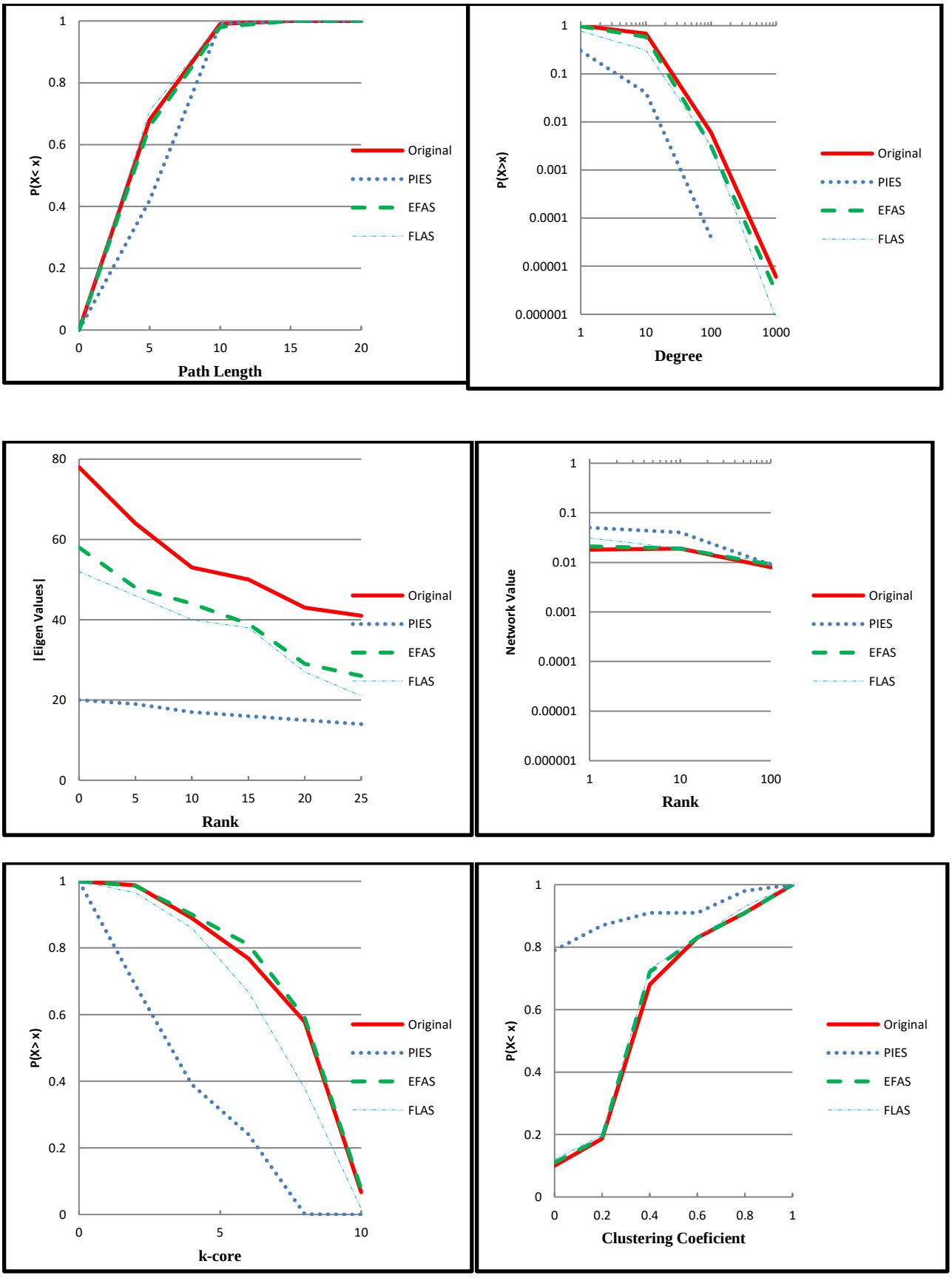

Figure 9. Comparison of sampling algorithms in the HepPH data set for different distribution statistics in the sampling section 0.2.

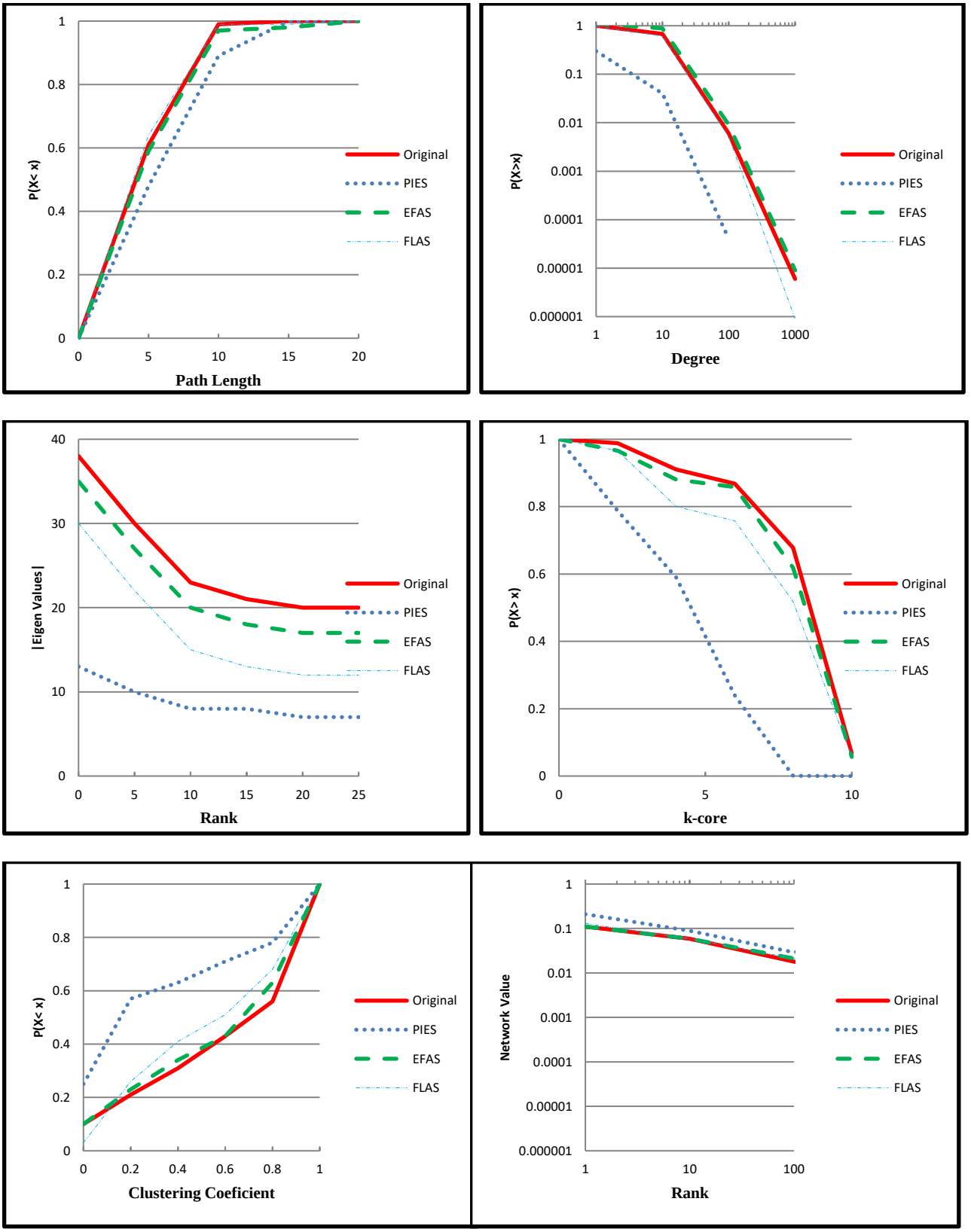

Figure 10. Comparison of sampling algorithms in the CondMAT data set for different distribution statistics in the sampling section 0.2.

5.3.3. Experiment 3

In most graph statistics, both algorithms EFAS and FLAS have a near-closing performance. However, we can see the impact of using the evaluator unit on the relative improvement of EFAS to FLAS. In this experiment, plot the distributions of six graph, statistics for each of the three datasets. Note that for the degree and k–core distributions, CCDF (complementary cumulative distribution function) is plotted, and for clustering coefficient and path length distributions, we plot CDF. While EFAS preserves the k–core distribution in almost all the datasets, FLAS and PIES have different functions either underestimated or overestimated in different distributions. As shown in figures 8, 9, and 10, EFAS preserves the degree distribution more accurately than PIES and a little bit better than FLAS. The PIES algorithm underestimates the degree distribution for all datasets, except for Flickr and clustering coefficient distribution for all datasets. The FLAS algorithm captures the clustering coefficient statistic for HepPH, and underestimates it for CondMAT for the k–core distribution. FLAS generally provides better results compared to the PIES algorithm. PIES overestimate the core structures in Flickr and underestimate them in the other datasets. FLAS captures the path length distribution for all the tested graphs. However, PIES underestimates this statistic for Flickr and overestimates it for CondMAT and HepPH. Both EFAS and FLAS, while, having a far better performance than PIES, are able to accurately estimate the eigenvalues, and especially, the network values for all test graphs.

5.3.4. Experiment VII

In Another experiment, we want to investigate the number of isolated nodes, connected components, and nodes 100 degrees higher (100 higher hubs) in the sub-graphs that are sampled by the proposed EFAS algorithm. Table 8 shows the results of the probability of the isolated nodes as well as the ratio of the top 100 hubs that are collected by sample sub-graphs for three algorithms. We also compare the number of connected components in the sub-graphs that were sample by three algorithms in sampling 0.2 with their actual number for each dataset in table 9. In these tables, it can be seen that, sub-graphs sampled by EFAS have, the less isolated nodes and include the more ratio of the highest degree nodes as compared to FLAS and PIES. In EFAS, the evaluator unit has paid special attention to the history of streaming edges, and high activity nodes have a high probability of counting in the sample sub-graphs, and for this reason, EFAS generates more connected sub-graphs than FLAS. As reported in table 9, sub-graphs that have been sampled by EFAS have fewer connected components than those sampled by FLAS and PIES, and so the graph will have more connections.

5.3.5. Experiment VIII

The purpose of this experiment is to evaluate the performance of EFAS, FLAS, and PIES over time, and compare the performance of these three while the main graph is in streaming. In the first step, an initial sub-graph is sampled from the main graph. Then, with a sampling fraction of 0.2, it is examined how the quality of the sample sub-graph changes after examining the proportions of the edges remaining from the current. The tested ratio varies from 0 to 30 percent. For each ratio, we report the mean KS distance in the data sets for the statistics degree, clustering, and path length distribution in figure 11. It is worth mentioning, that the ratio 0 is related to the initial sub-graph, which has not been checked for any remaining edges of the stream. It is clear that, the increase in the ratio of the revised edges reduces the mean KS distance for all the three algorithms. However this decline is faster for EFAS, so in the same proportion, EFAS always provides much better results than FLAS and PIES. In the following, we also examine the EFAS by setting a stop condition, where the sample update continues until all the learning automata of the node in V_s are in their most internal state. In table 10, the average ratio of the edges to be reviewed, and the mean KS distance for the three graphs, statistics with EFAS with and without the use of stopping conditions are reported. Based on the results obtained, EFAS meets the stop condition in the whole dataset after an average review of 58% of the remaining edges of the current stream and the results close to when there is no stop condition considered in EFAS. Hence, in order to reduce the computational cost, we can use the EFAS sampling algorithm with stopping conditions and observe the results close to EFAS without stopping conditions.

5.3.6. Experiment IX

Downward bias in the degree distribution is one of the challenges in sub-graphs sampled from scale-free networks; in this experiment, we validate how the evaluator unit can offset that. We create two synthetic networks, called S-W1, and S-W2 as small networks that are based on the Watts-Stroggatz model [33], and two latter networks, called S-F1 and S-F2 are scale-free networks based on the Barabasi-Albert model [33]. Table 11 provides information about these synthetic

networks. One of the advantages of our proposed algorithm is to consider the high degree bias in the evaluator unit and use it more in order to preserve more accurately the degree distribution of the scale-free networks. For a sampling fraction of 0.2, we plot the degree distribution for each synthetic network (Figure 12). We can see the effect of using that in the degree distribution. Although it is more difficult to achieve degree distribution in small-world networks, EFAS also has a better performance in these networks. As expected, EFAS is always better than FLAS and PIES due to the use of the evaluator unit, especially in scale-free networks.

Table 8. The probability of isolated nodes and the proportion of hubs.

Dataset	Isolated Node			Hub (Top 100)		
	EFAS	FLAS	PIES	EFAS	FLAS	PIES
Flickr	0.062	0.1301	0.1623	0.77	0.59	0.20
HepPH	0.005	0.0424	0.0970	0.91	0.72	0.68
CondMAT	0.0015	0.1074	0.1514	0.84	0.82	0.55

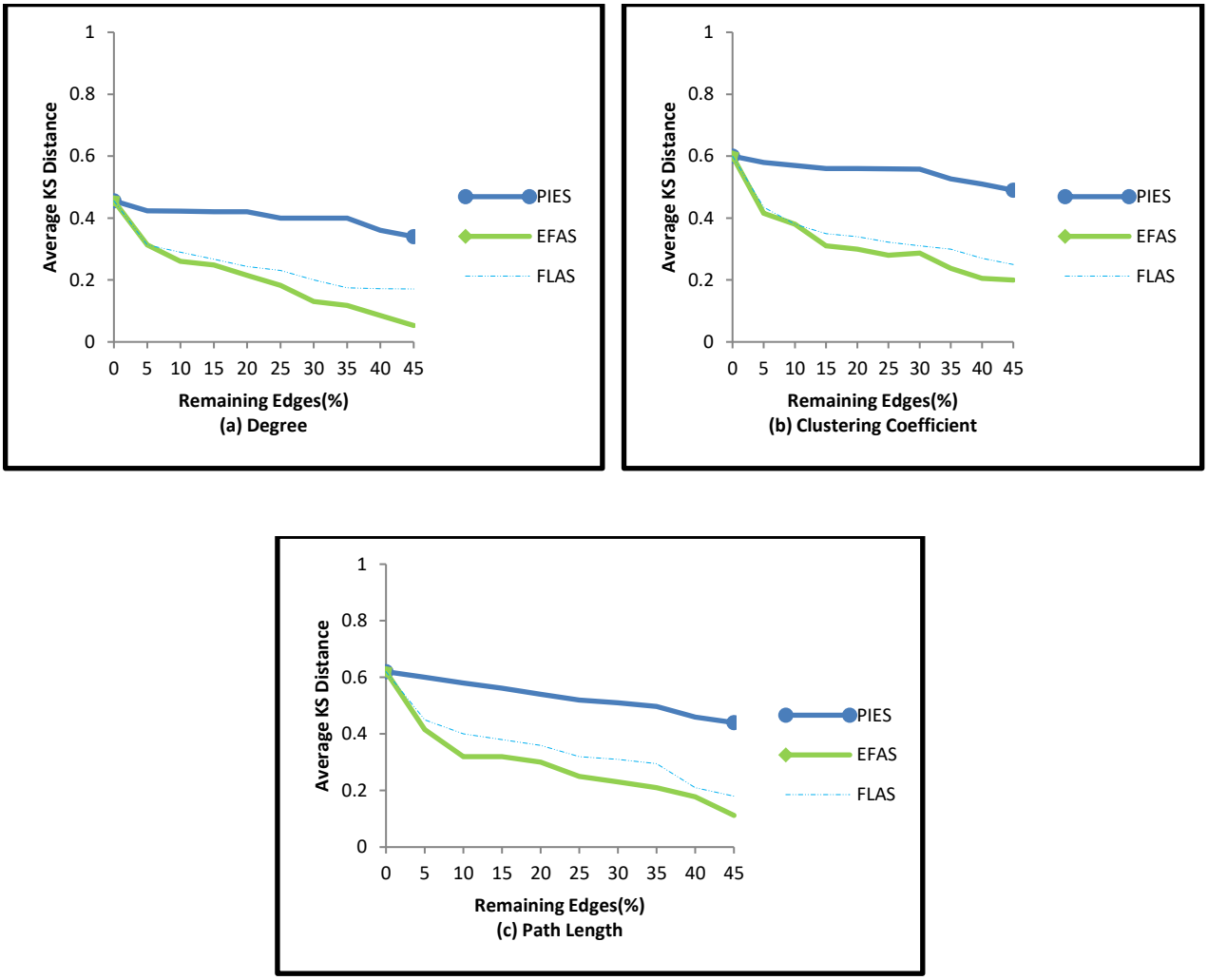

Figure 11. The mean KS distance in all data sets for different edges, in the sampling fraction of 0.2.

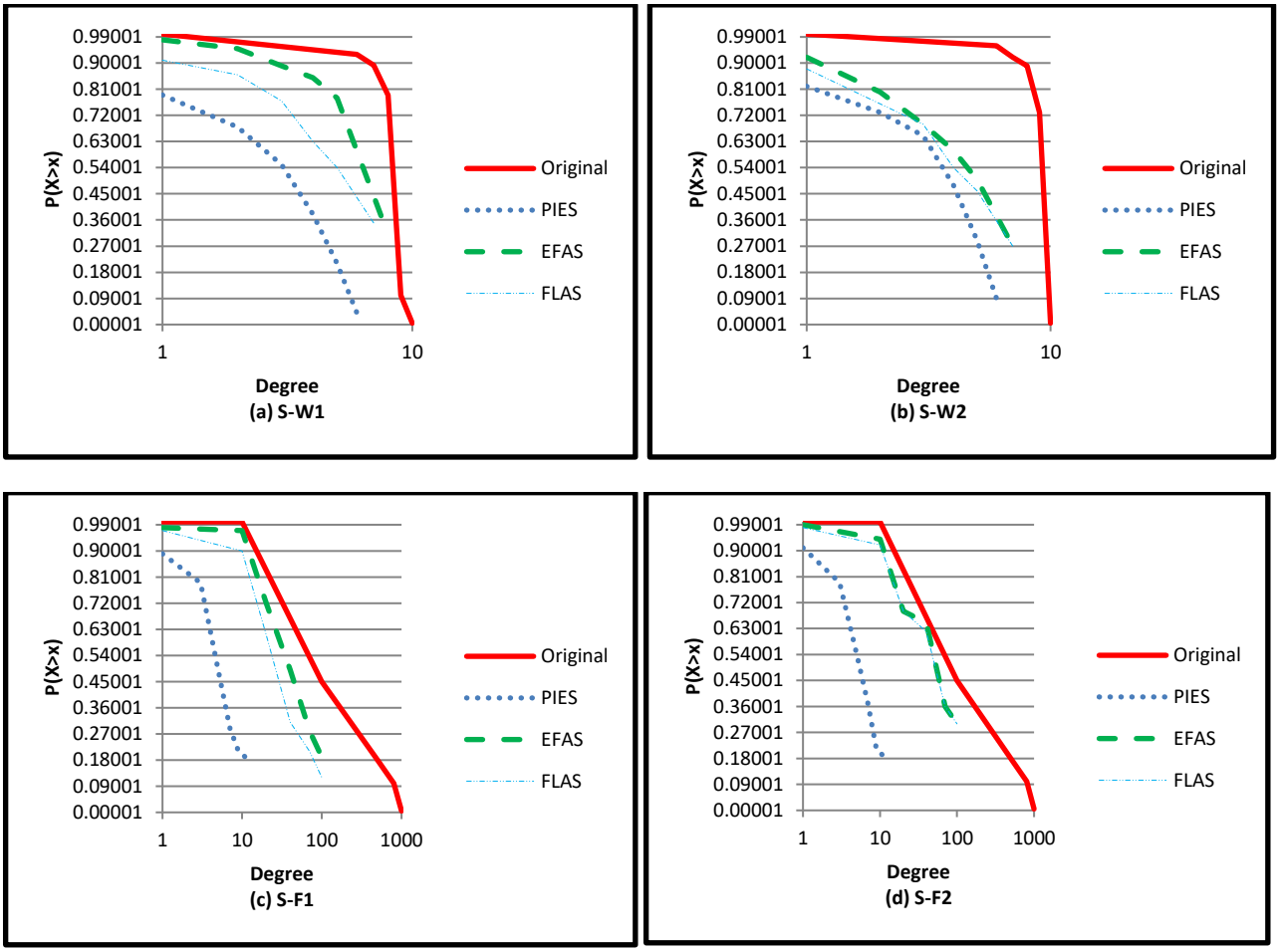

Figure 12. Impact of high degree bias in sampled degree distribution for different types of synthetic networks in sampling fraction 0.2

Table 9. The number of connected components for the sampling fraction of 0.2 compared to its actual value for each dataset.

Dataset	Real Connected Components	EFAS	FLAS	PIES
Flickr	1	782	919	21668
HepPH	61	14	27	316
CondMAT	567	113	127	340

Table 10. Average reviewing proportion and KS distance across all datasets for EFAS, in the sampling fraction of 0.2.

EFSA	% Remaining Edges	Deg.	Clust.	Path
Without stop condition	100	0.0300	0.0717	0.012
With stop condition	58	0.1267	0.1056	0.1498

Table 11. Characteristics and parameters of synthetic networks

Graph	Nodes	Edges	Density	Parameters
S-W1	30000	90000	2E-4	K=6, P=0.2
S-W2	20000	100000	5E-4	K=10, P=0.2
S-F1	30000	149987	3.3E-4	m0=10, m=5
S-F2	20000	99985	4.9E-4	m0=6, m=5

6. Conclusion

In this paper, we addressed the problem of sampling from activity networks in which the stream of edges continuously evolves over time. We proposed EFAS, a new streaming sampling algorithm based on a simpler fixed structure learning automata, which overcomes the PIES's drawbacks and an evaluator unit that evaluates nodes based on a variety of aspects that overcomes the FLAS's drawback, while maintaining its advantages. As a result, EFAS not only provides good results on sparse, less clustered graphs, but also can produce high quality sub-graphs for dense and highly clustered graphs.

References

[1] Pinto, P., et al. (2017). Data mining and social web semantics: a case study on the use of hashtags and memes in Online Social Networks. IEEE Latin America Transactions, vol. 15, no. 12, pp. 2276-2281.

[2] Z.Karimi Zandian; M. R. Keyvanpour. (2019). MEFUASN: A Helpful Method to Extract Features using Analyzing Social Network for Fraud Detection. Journal of AI and Data Mining, vol. 7, no. 2, pp. 213-224.

[3] Leskovec, J. and C. Faloutsos. (2006). Sampling from large graphs. in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, 2006.

[4] Ebbes, P., Z. Huang, and A. Rangaswamy. (2012). Subgraph Sampling Methods for Social Networks: The Good, the Bad, and the Ugly. No. hal-02058253. 2010.

[5] Yoon, S., et al. (2007). Statistical properties of sampled networks by random walks. Physical Review E, vol. 75, no. 4, pp. 046-114.

[6] Lee, S.H., P.-J. Kim and H. Jeong. (2006) Statistical properties of sampled networks. Physical Review E, vol. 73, no. 1, pp. 016-102.

[7] Rezvanian, A. and M.R. Meybodi. (2017). A new learning automata-based sampling algorithm for social networks. International Journal of Communication Systems, vol. 30, no. 5, pp. e3091.

[8] Ghavipour, M. and M.R. Meybodi. (2017). Irregular cellular learning automata-based algorithm for sampling social networks. Engineering Applications of Artificial Intelligence, vol. 59, pp. 244-259.

[9] Rezvanian, A. and M.R. Meybodi. (2015). Sampling social networks using shortest paths. Physica A: Statistical Mechanics and its Applications, vol. 424, pp. 254-268.

[10] Kurant, M., A. Markopoulou, and P. Thiran. (2011). Towards Unbiased BFS Sampling. IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1799-1809.

[11] Hübler, C., et al. (2008). Metropolis algorithms for representative subgraph sampling. IEEE International Conference on Data Mining. Pisa, Italy 2008.

[12] Krishnamurthy, V., et al. (2007). Sampling large Internet topologies for simulation purposes. Computer Networks, vol. 51, no. 15, pp. 4284-4302.

[13] Ahmed, N.K., J. Neville, and R. Kompella. (2014). Network sampling: From static to streaming graphs. ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 8, no. 2, pp. 7.

[14] Sarma, A.D., S. Gollapudi, and R. Panigrahy. (2011). Estimating pagerank on graph streams. Journal of the ACM (JACM), vol. 58, no. 3, pp. 13.

[15] Chen L, W.C. (2010). Continuous subgraph pattern search over certain and uncertain graph streams. IEEE Trans Knowl Data Eng, vol. 22, pp. 1093–1109.

[16] Aggarwal CC, Z.Y., Yu PS. (2010). On clustering graph streams, in Proceedings of the 2010 SIAM international conference on data mining SIAM. 2010. p. 478–489.

[17] Aggarwal, C.C., et al. (2010). On dense pattern mining in graph streams. Proceedings of the VLDB Endowment, vol. 3, no. 2, pp. 975-984.

[18] Buriol, L.S., et al. (2006). Counting triangles in data streams. in Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.

[19] Aggarwal, C.C. (2006). On biased reservoir sampling in the presence of stream evolution. in Proceedings of the 32nd international conference on Very large data bases.

[20] Bar-Yossef, Z., R. Kumar, and D. Sivakumar. (2002). Reductions in streaming algorithms, with an application to counting triangles in graphs. in Proceedings of the thirteenth annual ACM-SIAM

symposium on Discrete algorithms. Society for Industrial and Applied Mathematics.

[21] Aggarwal, C.C., Y. Zhao, and P.S. Yu. (2011). Outlier detection in graph streams. in 2011 IEEE 27th International Conference on Data Engineering, Piscataway, NJ, 2011.

[22] Ahmed NK, B.F., Neville J, Kompella R. (2010). Timebased sampling of social network activity graphs Proceedings 8th Work. Min. Learn. with Graphs, vol. 75, no. 4, pp. 1-9.

[23] Cormode, G. and S. Muthukrishnan. (2005). Space efficient mining of multigraph streams, in Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2005, ACM: Baltimore, Maryland. p. 271-282.

[24] Narendra KS, T.M. (2012). Learning automata: an introduction. . 2012: Courier Corporation.

[25] Thathachar MAL, S.P. (2011). Networks of learning automata: techniques for online stochastic optimization. Springer Science & Business Media.

[26] Ghavipour, M. and M.R. Meybodi. (2016). An adaptive fuzzy recommender system based on learning automata. Electronic Commerce Research and Applications, vol. 20, pp. 105-115.

[27] Rezapoor Mirsaleh, M. and M.R. Meybodi. (2016). A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem. Memetic Computing, vol. 8, no. 3, pp. 211-222.

[28] M, T. (1961). On behaviour of finite automata in random medium. Avtom I Telemekhanika, vol. 22, no. 4, pp. 1345-1354.

[29] DF, G. (2006). Graph of flickr photo-sharing social network crawled in May 2006.

[30] Leskovec J, K.A. (2014). SNAP Datasets: Stanford Large Network Dataset Collection. 2014.

[31] Goldstein, M.L., S.A. Morris and G.G. Yen. (2004). Problems with fitting to the power-law distribution. The European Physical Journal B - Condensed Matter and Complex Systems, vol. 41, no. 2, pp. 255-258.

[32] Watts, D.J. and S.H. Strogatz. (1998). Collective dynamics of 'small-world'networks. nature, vol. 393, no. 6684, pp. 440.

[33] Barabási, A.-L. and R. Albert. (1999). Emergence of Scaling in Random Networks. Science, vol. 286, no. 5439, pp. 509.

نشریه هوش مصنوعی و داده کاوی

نمونه گیری از شبکه های اجتماعی به کمک بهره گیری از واحد ارزیاب در ماشین های یادگیر با ساختار ثابت

سعید روح اللهی[1]، عمید خطیبی بردسیری[2*] و فرشید کی‌نیا[3]

[1] گروه مهندسی کامپیوتر، واحد کرمان، دانشگاه آزاد اسلامی ،کرمان، ایران.

[2*] گروه مهندسی کامپیوتر، واحد کرمان، دانشگاه آزاد اسلامی ،کرمان، ایران.

[3] گروه مدیریت و بهینه سازی انرژی، پژوهشگاه علوم و تکنولوژی پیشرفته و علوم محیطی، دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته، کرمان، ایران.

ارسال ۲۰۱۸/۰۶/۰۶؛ بازنگری ۲۰۱۹/۰۴/۱۶؛ پذیرش ۲۰۱۹/۱۰/۰۶

چکیده:

گراف متناظر با شبکه‌های اجتماعی را میتوان گراف‌هایی با ساختار متغیر و توام با طیف گسترده‌ای از تعاملات لحظه‌ای ما بین کاربران تعریف نمود. هر گونه فعالیت بین کاربران در این شبکه‌ها که میتوان به توئیت ها، رایانامه‌ها و بسیاری عناوین مشابه دیگر اشاره کرد منجر به ایجاد یال متناظر مابین آنها خواهد گردید. با وجود عمومیت شبکه‌های اجتماعی، توپولوژی متغیر در کنار حجم و اندازه بالای اینگونه شبکه‌ها امکان مطالعه روی آنها را تا حدودی ناممکن می‌گرداند. بدیهی است نمونه‌گیری سنتی جوابگو نبوده و باید از نمونه‌گیری جریانی متناسب با این نوع شبکه‌ها بهره برد. در این مقاله ، الگوریتم نمونه‌گیری را معرفی خواهیم نمود که در آن هر نود از گراف مجهز به ماشین یادگیر با ساختار ثابت خواهند بود. در این الگوریتم واحدی تحت عنوان واحد ارزیاب وظیفه انتخاب یالها و نودهای متناظر و درج آنها به مجموعه نمونه‌گیری بر عهده خواهد داشت. نتایج حاصله از الگوریتم پیشنهادی بر اساس آزمونهای آماری از جمله آزمون کولموگروف-اسمیرنوف، فاصله اقلیدسی و منهتن بین متریک‌های حاصله از گراف اصلی و گراف نمونه‌گیری شده در چندین شبکه واقعی و شبکه مصنوعی مورد ارزیابی و مقایسه قرار گرفت. نتایج بدست آمده حاکی از برتری الگوریتم پیشنهادی در مقایسه با سایر روشهای نمونه‌گیری ارائه شده در حوزه‌های مشابه می‌باشد.

کلمات کلیدی: واحد ارزیاب، شبکه‌های اجتماعی، نمونه‌گیری شبکه، نمونه‌گیری جریانی، ماشین یادگیر با ساختار ثابت.