Irregular Cellular Learning Automata-based Algorithm for Sampling Social Networks

Mina Ghavipour [a], Mohammad Reza Meybodi [b]

[a,b] *Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran*

Abstract: Since online social networks usually have quite huge size and limited access, smaller subgraphs of them are often produced and analysed as the representative samples of original graphs. Sampling algorithms proposed so far are categorized into three main classes: node sampling, edge sampling, and topology-based sampling. Classic node sampling algorithm, despite its simplicity, performs surprisingly well in many situations. But the problem with node sampling is that the connectivity in sampled subgraph is less likely to be preserved. This paper proposes a topology–based node sampling algorithm using irregular cellular learning automata (ICLA), called ICLA-NS. In this algorithm, at first an initial sample subgraph of the input graph is generated using the node sampling method and then an ICLA isomorphic to the input graph is utilized to improve the sample in such a way that the connectivity of the sample is ensured and at the same time the high degree nodes are also included in the sample. Experimental results on real–world social networks indicate that the proposed sampling algorithm ICLA-NS preserves more accurately the underlying properties of the original graph compared to existing sampling methods in terms of Kolmogorov-Smirnov (KS) test.

Keywords: Complex networks, Social networks, Network sampling, Graph mining, Cellular learning automata

1. Introduction

Social network and the analysis of it is an inherently interdisciplinary field which is emerged from computer science, sociology, social psychology, statistical physics, and graph theory (Yang et al., 2013; Li et al., 2013; So and Long, 2013). The research on social network offers a framework for analysing the structure of whole network graph, identifying local and global patterns in these structures and studying dynamical properties on the network. Despite the importance of studying real–world social networks, it would be difficult to capture the structural properties of the whole network since we often face large scale networks with access limitations. To deal with this problem, many sampling methods have been reported in literature. Generally, the main goal of a sampling method is to produce a representative subgraph from the original network which can be used for studying characteristics of the larger network. The term “representative subgraph sampling” defined by Leskovec and Faloutsos (2006) refers to producing a small sample of the original network, whose characteristics represent as accurately as possible the entire network. There exist many characteristics which describe a network structure such as degree, clustering coefficient, and k –core distributions. Authors in (Leskovec and Faloutsos, 2006) proposed a set of empirical rules by which the measurements of the sample can be scaled up, to recover estimates for the original graph. Work in (Ebbes et al., 2013) investigated the ability of nine different sampling methods in preserving the structural properties such as degree, clustering coefficient, betweenness centrality, and closeness centrality of social networks. Lee et al. (2006) exploited three sampling algorithms and investigated the statistical properties of the samples taken by them. They focused on the topological properties such as degree distribution, average path length, assortativity, clustering coefficient, and betweenness centrality distribution.

Existing sampling methods can be categorized into three main classes: node, edge and topology–based sampling. Despite its simplicity, classic node sampling method performs surprisingly well in many situations (Leskovec and Faloutsos, 2006). The problem with node sampling is that connectivity in sampled subgraph is less likely to be preserved (Lee et al., 2006). Considering this in mind, this paper propose a topology–based node sampling algorithm, called ICLA-NS, that utilizes an irregular cellular learning automata (ICLA) to produce representative subgraphs by ensuring the connectivity of sampled subgraphs and sampling the nodes with high degree. ICLA-NS first constructs an initial sample subgraph of the input graph using the node sampling method, and then uses an ICLA isomorphic to the input graph to improve the sample by repeatedly replacing nodes in the sample with the nodes found by exploring the input graph. In order to evaluate the performance of the proposed sampling algorithm, we conduct a number of experiments on real–world networks. Based on our experimental results, the proposed sampling algorithm outperforms the existing sampling algorithms such as node sampling, Random Walk sampling and Forest Fire sampling in terms of Kolmogorov-Smirnov (KS) test for degree, clustering coefficient, and k–core distributions.

The rest of the paper is organized as follows. The next section presents notations and related works on network sampling. In Section 3, learning automata and cellular learning automata are introduced. Section 4 describes the proposed sampling algorithm ICLA-NS. Experimental results are given in section 5. Section 6 concludes the paper.

2. Foundations of Graph Sampling

While the explicit goal of graph sampling algorithms is to produce a smaller subgraph from the original graph, there often exist other implicit goals for a sampling process. Three possible goals of graph sampling algorithms are: *Scale-down sampling*, *Back-in-time sampling*, and *Supervis*ed *sampling*. The Scale-down sampling aims to sample a representative subgraph that have similar (or scaled-down) topological properties to those of the original graph (Leskovec and Faloutsos, 2006). In Back-in-time sampling, the sampled subgraph matches temporal evolution of the original graph (Leskovec and Faloutsos, 2006). That is, the sampled subgraph G_s is similar to what the original graph G looked like when it was of the same size as G_s. Finally, the goal of supervised sampling is to identify nodes belonging to a specific category (Fang et al., 2013; 2015; 2016). For this purpose, a biased sampling is done to sample a subgraph under the requirements related to that category.

In this paper, we focus on the goal of sampling a representative subgraph (Scale-down sampling). This section provides some common notations and a formal definition of the graph sampling problem considered in this paper. In this section, several taxonomies of sampling algorithms reported in the literatures are also given.

2.1. Notations and Definitions

Let $G(V, E)$ be an unweighted and undirected graph with the node set $V = \{v_1, v_2, \dots, v_n\}$ and the set of edges $E = \{e_{ij} \mid v_i \in V, v_j \in V\}$, such that $|V| = n$ denotes the number of nodes, and $|E| = m$ denotes the number of edges. The neighbourhood of node v_i is defined as $N(v_i) = \{v_j \mid e_{ij} \in E, v_j \in V\}$, such that $d(v_i) = |N(v_i)|$ is the degree of node v_i.

In this paper, we consider the following graph sampling problem. Given an input graph $G(V,E)$ and a sampling fraction f, a sampling algorithm samples a subgraph $G_s(V_s,E_s)$ with a subset of the nodes $V_s \subset V$ and a subset of the edges $E_s \subset \{e_{ij} \mid v_i \in V_s, v_j \in V_s\}$, such that $|V_s| = fn$. The goal is to ensure that the sampled subgraph G_s preserves the topological properties of the original graph G.

2.2. Related Works

Graph sampling algorithms can be classified in several ways. We present three such classifications, namely random versus topology –based sampling, static versus streaming graph sampling, and simple versus extended sampling.

2.2.1 Random Versus Topology –based Sampling

In random sampling methods, a sample subgraph is constructed by the random selection of either nodes or edges, and so these methods can be categorized into two main subclasses: node, and edge sampling. Classic node sampling (NS) (Leskovec and Faloutsos, 2006) chooses nodes uniformly at random from the original graph G. For a required fraction f of nodes, each node is independently sampled with a probability of f. The sampled subgraph G_s consists of the chosen nodes V_s as well as all the edges among them (E_s). Despite its simplicity, classic NS performs surprisingly well in many situations (Leskovec and Faloutsos, 2006). Authors in (Stumpf et al., 2005) indicated that classic node sampling does not accurately retain properties for graphs with power–law degree distributions. Although node sampling includes all the edges related to the sampled node set V_s, Lee et al. (2006) shown that the original level of connectivity is less likely to be preserved. Many other variations of NS have been developed in recent years (Krishnamurthy et al., 2005; Leskovec and Faloutsos, 2006; Ahmed et al., 2010; Ahmed et al., 2013).

Classic edge sampling (ES) chooses edges (rather than nodes) independently and uniformly at random from the original graph G. For each edge chosen (and added to E_s), both incident nodes are added to V_s. So, the sampled subgraph G_s is constructed by including the sampled edges in E_s and their incident nodes in V_s. Since classic edge sampling is biased towards high degree nodes and samples both incident nodes of chosen edges, it can accurately preserve the path length distributions (Ahmed et al., 2010; Ahmed et al., 2013). However, ES is less likely to capture the original clustering and connectivity, since it samples the edges independently (Lee et al., 2006). Classic edge sampling generally produces sparse subgraphs. Some improved variations of ES have been proposed so far (Krishnamurthy et al., 2005; Leskovec and Faloutsos, 2006; Ahmed et al., 2010; Ahmed et al., 2013).

There also exist many sampling methods based on topological structure of graph. The common idea in this class of sampling methods is to explore the neighbouring nodes of a given node. These methods can be categorized into two subclasses: *random walks* and *graph traversals*. In the category *random walks*, sampling is performed with replacement, i.e. nodes can be revisited. This category includes classic Random Walk (RW) (Lovász, 1993; Yoon et al., 2007; Lu and Li, 2012) and its variations (Henzinger et al., 2000; Gkantsidis et al., 2006; Stutzbach et al., 2009; Rasti et al., 2009; Ribeiro and Towsley, 2010; Avrachenkov et al., 2010; Kurant et al., 2011; Lee et al., 2012; Rezvanian et al., 2014). In the category *graph traversals*, each node is visited at most once (sampling without replacement). Methods in this category differ in the order in which they visit the nodes. Examples are Breadth –First Search (BFS) (Lee et al., 2006), Depth –First Search (DFS) (Even, 2011), Forest Fire (FF) (Leskovec and Faloutsos, 2006), Snowball Sampling (SBS) (Goodman, 1961; Newman, 2003b; Illenberger et al., 2011), Respondent–Driven Sampling (RDS) (Heckathorn, 1997; Goel and Salganik, 2010), and Expansion Sampling (Maiya and Berger-Wolf, 2010). The sampled subgraph G_s in topology–based sampling methods consists of the explored nodes and edges. Sampling methods based on topology outperform simple methods such as NS and ES (Leskovec and Faloutsos, 2006).

2.2.2 Static Versus Streaming Graph Sampling

The sampling algorithms based on the assumption of a static graph (Goodman, 1961; Lovász, 1993; Heckathorn, 1997; Leskovec and Faloutsos, 2006; Lee et al., 2006; Maiya and Berger-Wolf, 2010; Even, 2011) consider the input graph only at one point in time and assume that it is of moderate size which can fit in the main memory. However, many real–world networks are too large to fit in the memory, and evolve continuously over time and thus are not fully observable at any point in time. Activity networks (e.g. email), social media (e.g. Twitter), and content sharing (e.g. Facebook, YouTube) are the examples of such large dynamic networks. Analysing these networks, called streaming graphs, is increasingly important for identifying patterns of interactions among individuals and investigating how the network structure evolves over time. As a result, the streaming graph sampling has received more attention in recent years. Researchers have developed algorithms for sampling from streaming graphs (Ahmed et al., 2010; Ahmed et al., 2013). Authors in (Ahmed et al., 2013) outlined a spectrum of computational models for designing sampling algorithms, going from static to streaming graphs. They presented the streaming variants of static sampling algorithms for node, edge and topology –based sampling approaches.

2.2.3 Simple Versus Extended Sampling

Simple sampling algorithms consist of one phase which implements the sampling process based on one of three sampling approaches (i.e. node, edge and topology –based sampling) or a combination of them. These algorithms sample each node which they visit for the first time and terminate instantly after visiting the required number of nodes (according to the sampling fraction). Thus, in this group of sampling algorithms, there is a trade –off between the sampling fraction and the accuracy of sampling. The traditional sampling algorithms (Goodman, 1961; Lovász, 1993; Heckathorn, 1997; Leskovec and Faloutsos, 2006; Lee et al., 2006; Maiya and Berger-Wolf, 2010; Even, 2011) fall in this group.

In contrast, extended sampling algorithms consist of two phases: one phase is dedicated to pre/post processing and other phase implements a simple sampling algorithm. As a result, algorithms in this group because of the time needed for pre/post processing have higher time complexity comparing to the simple sampling algorithms but produce samples with higher accuracy. The pre–processing phase is usually done with the goal of finding important nodes/edges to be added to the sample subgraph. Some extended sampling algorithms with pre –processing phase are given in (Rezvanian et al., 2014; Rezvanian and Meybodi, 2015; Yoon et al., 2015). In (Rezvanian et al., 2014), during the pre-processing phase a distributed learning automata is used to find important nodes in the original graph. The algorithm reported in (Rezvanian and Meybodi, 2015), in its pre–processing phase, uses the concept of shortest path for finding important edges of the input graph. The sampling algorithm in (Yoon et al., 2015) extracts the communities of the original graph in its pre–processing phase in order to find important nodes and edges in the graph.

Some extended sampling algorithms with post–processing phase have been also reported in the literature. The post–processing phase is usually utilized with the goal of improving the initial sample created by a simple sampling algorithm. For instance, the work reported in (Ahmed et al., 2013) creates an initial subgraph by sampling edges at the first of the edge stream such that a required percent of nodes is sampled. Then, the initial subgraph is improved by randomly sampling the remaining edges of the stream with the goal of selecting nodes with high degree. In this paper, we also propose an extended sampling algorithm with post–processing phase for sampling representative subgraphs from social networks.

3. Cellular Learning Automata

In this section, we briefly introduce cellular automata (CA), learning automata (LA), and cellular learning automata (CLA) as a combination of cellular automata and learning automata. Then, irregular cellular learning automata (ICLA) is described, which is an extension of traditional cellular learning automata.

3.1. Cellular Automata

Cellular automata (CA) (Wolfram, 1983) mathematically models complex systems which consist of simple identical components with local interactions. It is a non-linear dynamic system with discrete space and time. A cellular automata is made of a regular lattice of simple identical cells which interact locally with each other to generate complicated behavioural patterns. Each cell has a finite set of states. In each discrete step, all the cells are synchronously activated and update their states by interacting with the local environments in the basis of a local rule. The local environment of a cell consists of the cell itself and its neighbourhood. The evolution of CA is determined by the initial states of all cells and the local rule.

3.2. Learning Automata

Learning automata (LA) (Thathachar and Sastry, 2011; Narendra and Thathachar, 2012) is an adaptive decision –making device which attempts to improve its performance by learning the optimal action from a finite set of available actions through repetitive interactions with an unknown random environment. At each step, the automaton chooses an action from its allowable actions based on the probability distribution kept over the set of actions, and performs it on the environment. The environment responses the chosen action in turn with a either a reward or a penalty. At last, the action probability distribution of the automaton is updated depending on the reinforcement signal from the environment. By repeating these interactions, the automaton converges to the optimal action which minimizes the probability of being penalized. Figure 1 shows the relationship between the learning automata and its random environment.

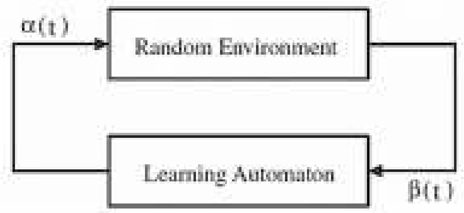


Fig.1. The relationship between the learning automata and the random environment

Learning automata can be categorized into two classes: variable structure learning automata (VSLA) and fixed structure learning automata (FSLA). A variable structure learning automaton can be defined by a quadruple $\langle \alpha, \beta, p, T \rangle$, where:

- $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ presents the action set that automaton chooses from.
- $\beta = \{\beta_1, \beta_2, \dots, \beta_k\}$ presents automaton‘s set of inputs.
- $p = \{p_1, p_2, \dots, p_r\}$ presents the action probability vector, such that p_i indicates the probability of choosing action α_i.

- T presents the learning algorithm that updates the action probability vector depending on the environment's response, i.e. $p(t+1) = T[\alpha(t), \beta(t), p(t)]$, where inputs are respectively the chosen action, the response of the environment and the action probability vector at step t.

Let α_i be the action chosen at instant t. The action probability vector $p(t)$ is updated according to Eq. 1, if the response from the random environment is reward, and $p(t)$ is updated as given in Eq. 2, if the environment's response is penalty.

$$p_j(t+1) = \begin{cases} p_j(t) + a\left[1 - p_j(t)\right] & j = i \\ (1-a)p_j(t) & \forall j \neq i \end{cases} \tag{1}$$

$$p_j(t+1) = \begin{cases} (1-b)p_j(t) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(t) & \forall j \neq i \end{cases} \tag{2}$$

where a and b are respectively reward and penalty parameters. If $a = b$, the learning algorithm is called reward –penalty (L_{R-P}) algorithm, if $a \gg b$, it is called reward –ϵ penalty ($L_{R-\epsilon P}$) algorithm, and if $b = 0$, it is called reward –Inaction (L_{R-I}) algorithm. There are several applications based on LA such as link prediction (Moradabadi & Meybodi, 2016), and fuzzy membership function optimization (Ghavipour & Meybodi, 2016).

3.2.1 Variable Action –Set Learning Automata

A variable action set learning automaton is defined as an automaton for which the number of available actions varies with time. At each instant t, a subset $\hat{\alpha}(t)$ of all the actions is available for the automaton to choose from. The selection of the elements of $\hat{\alpha}(t)$ is randomly made by an external factor. Before selecting an action, the probabilities of all available actions in the subset $\hat{\alpha}(t)$ are scaled as follows. Let $K(t) = \sum_{\alpha_i \in \hat{\alpha}(t)} p_i(t)$ be the sum of the probabilities of the actions in $\hat{\alpha}(t)$. The scaled probability of choosing action $\alpha_i \in \hat{\alpha}(t)$ is defined as

$$\hat{p}_i(t) = p_i(t)/K(t) \quad \forall \alpha_i \in \hat{\alpha}(t) \tag{3}$$

The automaton randomly chooses an action from $\hat{\alpha}(t)$ according to the scaled action probability vector $\hat{p}(t)$. Depending on the reinforcement signal from the environment, the vector $\hat{p}(t)$ of the automaton is updated. Finally, the action probability vector $\hat{p}(t)$ is rescaled as given in Eq. 4.

$$p_i(t+1) = \hat{p}_i(t+1)\, K(t) \quad \forall \alpha_i \in \hat{\alpha}(t) \tag{4}$$

3.3. Cellular Learning Automata

Cellular learning automata (CLA, a combination of CA and LA) (Beigy and Meybodi, 2004) is considered as a powerful mathematical model for many decentralized problems and phenomena. The basic idea of cellular learning automata is to assign a learning automaton to each cell to adjust the state transition of corresponding cellular automaton. At each instant, the learning automaton residing in each cell chooses its action (state for the cell) according to its action probability distribution. The local environment of a learning automaton consists of the learning automata in the adjacent cells. Since the action probability distribution of the neighbouring learning automata varies during the evolution of the CLA, the local environment is considered to be non –stationary. Like CA, the CLA operates under a local rule. The reinforcement signal to the learning automaton is determined on the basis of the local rule and the actions chosen by the neighbouring learning automata. Finally, the learning automaton updates its action probability vector based on the received reinforcement signal. This process continues until the optimal state of each cell is achieved. A d –dimensional cellular learning automata can be defined by a structure $\Lambda = (Z^d, \Phi, A, N, F)$, where:

- Z^d presents a lattice of cells such that each cell is represented by a d –tuple of integer numbers.
- Φ presents a finite set of states (actions).
- A presents the set of learning automata, each of which is assigned to a cell of CLA.
- $N = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_k\}$, called neighbourhood vector, presents a finite set of integer numbers, which indicates the relative position of the neighboring cells, i.e. for a particular cell z, the neighboring cells are the set $\{z + \bar{x}_i | \, i = 1,2, \dots, k\}$.
- $F: \Phi^k \rightarrow \beta$ presents the local rule of the CLA, where β is the value set of the reinforcement signal and Φ^k is the states of the neighboring learning automata. The function F computes the reinforcement signal for each LA according to the current states of the neighboring learning automata.

Cellular learning automata can be categorized into two models: synchronous and asynchronous. In synchronous cellular learning automata, the learning automata in all the cells are activated at the same time. It has been shown that the synchronous CLA converges to a globally stable state for commutative rules. In asynchronous cellular learning automata (ACLA), at a given time only some learning automata are independently activated, rather than all LAs in parallel. The activation of LAs is done in either time –driven or step –driven manner. In time –driven, an internal clock in each cell wakes up the LA associated to that cell. However, the step –driven manner considers a fixed or random sequence for selecting cells.

3.4. Irregular Cellular Learning Automata

Irregular cellular learning automata (ICLA) (Esnaashari and Meybodi, 2008a) is a generalization of traditional CLA which removes the limitation of rectangular grid structure. Such a generalization seems to be necessary since there are many applications such as graph related applications, wireless sensor networks, and immune network systems, which cannot be modelled with regular grids. An ICLA is considered as an undirected graph in which each node is a cell being equipped with a learning automaton, and the neighbouring nodes of any particular node constitute the local environment of that cell. Despite its irregular structure, ICLA behaves equivalently with CLA. ICLA has been found to perform well in a number of applications (Esnaashari and Meybodi, 2008a; Esnaashari and Meybodi, 2008b; Rezapoor Mirsaleh and Meybodi, 2016).

4. The Proposed Graph Sampling Algorithm

In this section, a node sampling algorithm based on topology using irregular cellular learning automata (ICLA), called ICLA-NS, will be proposed. The original graph $G\ (V, E)$ and the sampling fraction f are the inputs and the output is a representative subgraph $G_s(V_s, E_s)$ which satisfies the constraint $|V_s| = f|V|$. The proposed algorithm is an extended sampling algorithm with post–processing phase. In this algorithm, at first an initial sample subgraph is produced by randomly choosing f percent of the nodes in the input graph. Then, an ICLA isomorphic to the input graph attempts to improve the initial sample by repeatedly replacing nodes in the sample with the nodes reached by exploring the input graph with the goal of guaranteeing the connectivity of the sample subgraph and at the same time sampling the high degrees nodes from the input graph. The pseudo code for the algorithm ICLA-NS is given in Fig.2.

```
Algorithm ICLA-NS
Input   Original graph $G(V,E)$ , Fraction of nodes $f$, Convergence threshold $\tau$
Output   Sampled subgraph $G_s(V_s,E_s)$
Begin
    ICLA mapping phase
        Associate each node $v_i$ with a cell equipped with an automaton $A_i$ with $d(v_i)+1$ actions
    Initialization phase
        $V_s \leftarrow$ Fraction $f$ of nodes in $V$ selected at random
        $Q \leftarrow$ cells corresponding to nodes in $V_s$
        Determine initial state for all cells
    Improvement phase
        Set action probability vector of each $A_i$ according to Eq. 5
        Repeat
            $A' \leftarrow$ automaton $A_i$ corresponding to cell at front of $Q$
            Repeat
                Automaton $A'$ chooses one of its actions according to $\hat{p}(t)$
                If  selected action  is its corresponding node $v'$  then
                    If  there is at least one sampled node adjacent to $v'$  then  Reward $A'$  Else  Penalize $A'$
                    $V_s \leftarrow V_s \backslash v_i \cup v'$
                    Remove cell at front of $Q$ and add cell corresponding to $v'$ to the end of $Q$
                Else   \\ Assume selected action is $v_k \in N(v')$
                    If  there is no sampled node adjacent to $v'$  then  Reward $A'$  Else  Penalize $A'$
                    $A' \leftarrow A_k$
                End
            Until selected action is  $v'$
        Until  $\prod_{v_i \in V_s} \hat{p}_i^i(t) \geq \tau$
End Algorithm
```

Fig.2. The pseudo code of the ICLA-NS algorithm

The proposed algorithm ICLA-NS consists of three phases: ICLA mapping phase, Initialization phase and Improvement phase.

ICLA mapping phase: In this phase, an irregular cellular learning automata (ICLA) isomorphic to the input graph G is constructed. The action set of the learning automaton A_i associated with node v_i has $d(v_i)+1$ actions, where $d(v_i)$ denotes the degree of node v_i. The automaton A_i can take one of the actions of giving the opportunity to the node v_i to be joined to the sample set V_s called the action “selection of the corresponding node v_i“ or giving the opportunity to one of the neighbours of v_i, say $v_k \in N(v_i)$, to be joined to V_s called the action “selection of the neighbour v_k of the node v_i“. At the beginning of the algorithm, all the cells of ICLA are considered to be inactive.

Initialization phase: In this phase, an initial subgraph of the input graph G is formed by randomly selecting f percent of the nodes of G (note that the selected nodes must be non – isolated). Every time a node v_i is selected, it is placed in the sample set V_s, and its corresponding cell is added to the queue Q. The action probability vector of each learning automaton A_i is set in such a way that if A_i is in one of the cells in Q, the action “selection of the corresponding node v_i“ will have the selection probability of 1, and otherwise the action “selection of the neighbour v_k of the node v_i“, where v_k is chosen at random from the neighbourhood of v_i, will have the selection probability of 1. After that, all the cells of ICLA are activated synchronously which in turn activate the learning automata to have an action selection based on their action probability vectors. This synchronous activation process is done with the goal of determining the initial state for all the cells of ICLA. The selected action by each learning automaton represents the initial state of its corresponding cell. In this paper, we may use the state of a cell and the action chosen by the learning automaton of that cell interchangeably.

Improvement phase: In this phase, ICLA starts operating asynchronously and attempts to improve the initial sample by repeatedly replacing its nodes with the nodes reached by exploring the graph G in such a way that the high degree nodes are sampled and at the same time the connectivity of the sampled graph is ensured. The order of the cells in Q will be used to activate the cells of the asynchronous ICLA during its operation. That is, the cell located at the front of Q will always be chosen for next cell activation. In this phase, the action probability vector of each automaton A_i is initialized according to Eq. 5.

$$p_i^j(0) = \frac{d(v_j)}{d(v_i)+\sum_{v_k \in N(v_i)} d(v_k)} \quad \forall v_j \in \{v_i \cup N(v_i)\} \tag{5}$$

where $p_i^j(0)$ is the probability that node v_j be chosen by A_i at instant $t = 0$ and $N(v_i)$ denotes the neighbourhood of node v_i. Using equation 5 for initializing the probability vectors of learning automata residing in the cells of ICLA, nodes with high degrees will be initially assigned higher probabilities of selection. This phase repeats the following three steps until the stopping condition is reached.

Step 1. Finding a substitute node by exploring the input graph

This step aims to find a substitute node for the node v_i corresponding to the cell at the front of Q. For this purpose, the input graph G is explored starting from v_i by a series of cell activations in ICLA as follows. At first, the cell located at the front of Q is activated. As a result of this cell activation, the learning automaton A_i in this cell is activated in order to decide whether its corresponding node v_i or one of the neighbouring nodes of v_i to be sampled (as the substitute node) or not. The set of available actions for A_i includes “selection of the corresponding node v_i” and “selection of the neighbour v_k of the node v_i” for any neighbour node $v_k \in N(v_i)$ such that v_k has not been already sampled**,** i.e. $v_i \cup \{v_k \in N(v_i) | \, v_k \notin V_s\}$. The activated automaton A_i selects one of its available actions according to the scaled action probability vector (for more details, see subsection 3.2.1). If A_i chooses the action “selection of the corresponding node v_i”, then the exploration process is finished and node v_i is considered as the substitute node. Otherwise, if A_i chooses the action “selection of the neighbour v_k of the node v_i”, the cell corresponding to v_k is activated which in turn activates the automaton A_k to have an action selection, and so on. This series of cell activations which explores the input graph G continues until the automaton in an activated cell selects the action “selection of the corresponding node”.

Each time an activated automaton, say A_i, selects an action; it receives a reward or penalty signal based on the following rules:

1. If the action chosen by the automaton A_i is “the selection of the corresponding node v_i”, then this action is rewarded if at least one of the neighbouring nodes of v_i has been already sampled otherwise the action will be penalized.

2. If the action chosen by the automaton A_i is "the selection of the neighbour v_k of the node v_i", then this action is rewarded if none of the neighbouring nodes of v_i has been already sampled otherwise the action will be penalized.

Step 2. Updating sample set V_s

The sample set V_s is updated by replacing the node v_i (which corresponds to the cell at the front of Q) with the substitute node found in the previous step. The cell at the front of Q is removed and the cell corresponding to the substitute node is added to the end of Q.

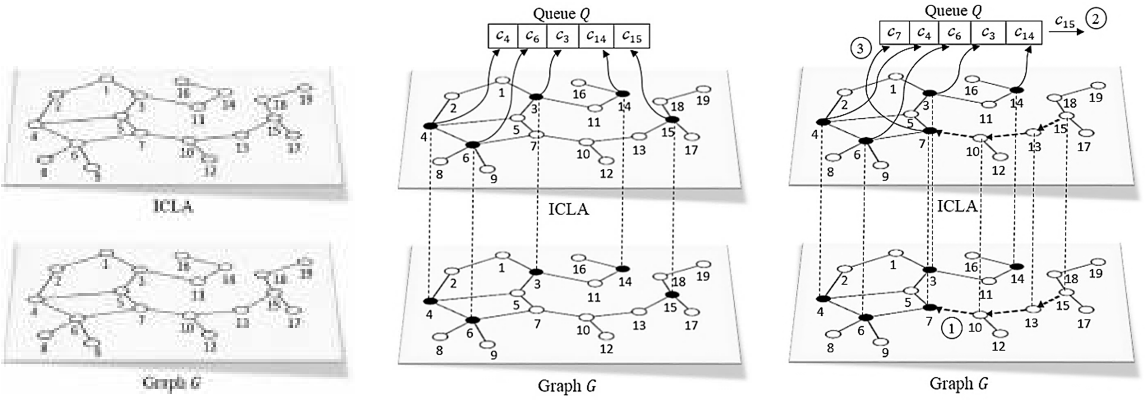
Step 3. Stop condition

The algorithm terminates when the product of the scaled probability of the action "selection of the corresponding node" for all the automata resided in the cells in Q (all the automata associated with the nodes in V_s) is greater than a predefined convergence threshold τ. In other words, the stop condition for the proposed algorithm is defined as

$$\prod_{v_i \in V_s} \hat{p}_i^i(t) \geq \tau \tag{6}$$

where $\hat{p}_i^i(t)$ is the scaled probability of selection of v_i by automaton A_i at instant t, and τ denotes a predefined convergence threshold. The set V_s that satisfies Eq. 6 along with all the edges between nodes in V_s constitute the final sampled subgraph.

Figure 3 illustrates different phases of the proposed algorithm ICLA-NS using a simple example.


ICLA
Graph G
Queue Q
c_4 c_6 c_3 c_{14} c_{15}
ICLA
Graph G
Queue Q
c_7 c_4 c_6 c_3 c_{14}
c_{15}
2
3
1
ICLA
Graph G

a) ICLA mapping phase

b) Initialization phase

c) Improvement phase

Fig.3. Description of three phases of the proposed algorithm ICLA-NS by a simple example: (a) Creating an ICLA isomorphic to the input graph G, (b) Sampling an initial subgraph, and adding the cells corresponding to the sampled nodes to the Queue Q, (c) Finding the substitute node v_7 for the node v_{15} corresponding to the cell at the front of Q by traversing a path in G starting from v_{15}, removing the cell at the front of Q, and adding the cell corresponding to v_7 to the end of Q.

Remark 1. The algorithm ICLA-NS has a selection bias to nodes with high degree for the following reasons:

1. In the algorithm ICLA-NS, the initial assignment to the action probability vector of each automaton gives a higher selection probability to the high degree nodes.
2. The algorithm ICLA-NS during its exploration process allows a node to be visited more than once and hence falls in the category *random walks* (see subsection 2.2.1 for more details). It has been observed empirically that random walks introduce a bias

towards nodes with high degree and explore densely connected parts of graphs (Lovász, 1993; Leskovec and Faloutsos, 2006).

Remark 2. The algorithm ICLA-NS ensures the connectivity in sampled subgraphs for the following reasons:

1. Sampling algorithms which are biased towards the selection of high degree nodes construct more connected subgraphs because of the fact that nodes with high degree often represent hubs in graphs (Leskovec and Faloutsos, 2006; Ahmed et al., 2013).
2. The algorithm ICLA-NS constructs a sample subgraph by sampling all the edges incident to the nodes in V_s (referred as *total graph induction* in (Ahmed et al., 2013)) and hence recovers much of the connectivity in sampled subgraphs (Ahmed et al., 2013).
3. In the algorithm ICLA-NS, each automaton learns to give the opportunity of being sampled to its corresponding node v_i only when there exists at least a sampled node adjacent to v_i. This ensures the connectivity of the sampled subgraph. Experiment VII reported in Section 5 confirms this claim.

Remark 3. The algorithm ICLA-NS preserves the topological properties of the original graph for the following reasons:

1. Since ICLA-NS produces connected subgraphs as representative samples, it preserves more accurately the properties of the original graph (Leskovec and Faloutsos, 2006; Ahmed et al., 2013).
2. Since any sampling algorithm produces sample subgraphs including only a subset of the nodes/edges of the original graph, it underestimates the degrees in the sampled degree distribution. This is a common property of all sampling algorithms, which is referred as *downward bias* in (Ahmed et al., 2013). The algorithm ICLA-NS is biased to high degree nodes which results in an *upward bias* in the original degree distribution, considering the degree of the sampled nodes as observed in the original graph. This upward bias can offset the downward bias of the sampled degree distribution and improve estimates of the sampled degree distribution (Ahmed et al., 2013).
3. The algorithm ICLA-NS is based on the concept of graph induction which recovers much of the connectivity in sampled subgraphs and offsets the downward degree bias (Ahmed et al., 2013).

5. Experimental Evaluation

In this section, the proposed sampling algorithm ICLA-NS is tested on several real –world networks with various structural properties and then results are compared with the results obtained for other well–known sampling algorithms. The datasets which are used in experiments include: *CMU* from the collection of Facebook Networks, *Advogato* and *Hamsterster* from the collection of Social Networks, and *CondMat*, *Erdos992* and *Netscience* from the collection of Collaboration Networks borrowed from (Rossi and Ahmed, 2013). A summary of the global statistics of these datasets has been provided in Table 1.

5.1. Network Statistics

We use three statistics to investigate the representativeness of subgraphs sampled by sampling algorithms in our experiments.

Table 1. Characteristics of used datasets

Dataset	Nodes	Edges	Density	Global Clustering
CondMat	21363	91286	4×10^{-4}	0.642
CMU	6621	249959	1.1×10^{-2}	0.279
Advogato	6551	51332	3.5×10^{-3}	0.287
Erdos992	6100	7515	4×10^{-4}	0.068
Hamsterster	2426	16630	5.7×10^{-3}	0.538
Netscience	379	914	1.3×10^{-2}	0.741

Degree Distribution. The distribution of the number of nodes in the network G with degree d, for every $d \geq 0$

$$p(d) = \frac{|\{v_i \in V | d(v_i) = d\}|}{n} \tag{7}$$

where $|V| = n$, and $d(v_i)$ denotes the degree of node v_i. The degree distribution of many real –world networks follows a power –law distribution (Albert and Barabási, 2002; Newman, 2003a; Dorogovtsev and Mendes, 2003).

Clustering Coefficient Distribution. The distribution of the number of nodes in the network G with clustering coefficient c, for every $0 \leq c \leq 1$

$$p(c) = \frac{|\{v_i \in V \mid d(v_i) > 1, CC(v_i) = c\}|}{|\{v_i \in V | d(v_i) > 1\}|} \tag{8}$$

where $CC(v_i)$ is the clustering coefficient of node v_i with degree greater than 1 and it is defined as the ratio between the number of edges among nodes within the neighbourhood of v_i and the total number of all possible edges between them (Dorogovtsev and Mendes, 2003),

$$CC(v_i) = \frac{2|\{e_{uv} \in E | v_u \in N(v_i), v_v \in N(v_i)\}|}{d(v_i)(d(v_i) - 1)} \tag{9}$$

K–core Distribution. The distribution of the number of nodes in the network G with coreness k, for every $k \geq 0$

$$p(k) = \frac{\{v_i \in V | \, Core(v_i) = k\}}{n} \tag{10}$$

where $Core(v_i)$ is the coreness of node v_i and it denotes the largest value of k such that v_i belongs to the k–core. The k–core of a network is considered as largest subgraph in which all the nodes have at least degree k. The k–core is a metric of the connectivity and community structure of a graph (Alvarez-Hamelin et al., 2005; Carmi et al., 2007; Kumar et al., 2010). The core sizes also demonstrate the localized density of subgraphs in graph (Seshadhri et al., 2013).

5.2. Evaluation Measure

The goal is to sample a representative subgraph G_s from the original graph G such that the distance between the property of G and that of G_s becomes minimum. In our experiments, we use the following distance measure for evaluation:

Kolmogorov-Smirnov (KS) statistic. It is widely used as measure of the agreement between two cumulative distribution functions (*CDF*) (Goldstein et al., 2004). The KS statistic is calculated as the maximum vertical distance between two distributions,

$$KS = \max_x |F(x) - F'(x)| \tag{11}$$

where x denotes the range of the random variable, F and F' are two CDFs, and $0 \leq KS \leq 1$. In this paper, we utilize the KS statistic for computing the distance between the true distribution of the original graph and the approximation distribution obtained from the sampled subgraph for the degree, clustering coefficient, and k–core distributions.

5.3. Experimental Results

In order to investigate the performance of our proposed algorithm ICLA-NS, several experiments are conducted on the real –world datasets described in Table 1. In the experiments I, II and III, the impact of the learning rate, the mechanism used for activating the cells of ICLA, and the sampling rate on the performance of the proposed algorithm ICLA-NS are investigated. In the experiment IV and V, the algorithm ICLA-NS is compared with some well –known sampling algorithms: Node Sampling (NS), Random Walk Sampling (RWS), Forest Fire Sampling (FFS) and also Distributed Learning Automata based Sampling (DLAS) (Rezvanian et al., 2014). We also define a cost function for extended sampling algorithms, which computes the cost spent by an algorithm in the pre/post processing phase. We then study the impact of increasing cost on the performance of ICLA-NS in the experiment VI. Finally, we investigate the impact of the learning process implemented by ICLA on the performance of ICLA-NS in the experiment VII. In ICLA-NS, the learning algorithm for updating the action probability vectors is L_{R-I} and the convergence threshold τ is set to 0.9. Each reported result is an average over 30 independent runs for a range of sampling fractions f from 0.1 to 0.3 with increment 0.05.

Experiment I

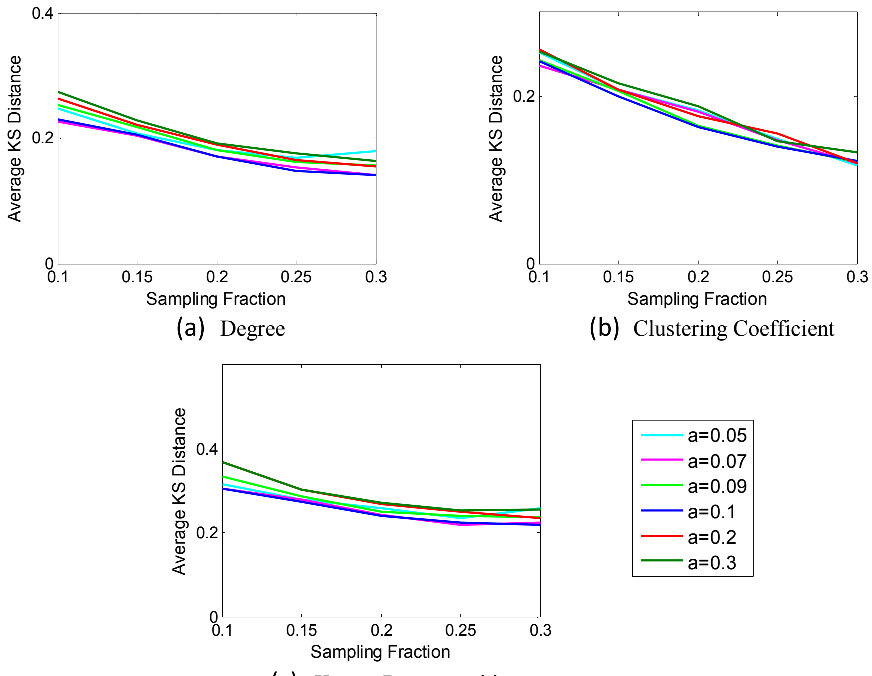
This experiment aims to investigate the impact of the learning rate a on the performance of the proposed algorithm ICLA-NS in terms of Kolmogorov-Smirnov (KS) distance for degree, clustering coefficient and k–core distributions. For this purpose, we test the algorithm for different values of the learning rate a varying from 0.05 to 0.3 and for each value of a, we report the average KS distance over all datasets for different sampling fractions f ranging from 0.1 to 0.3. Figure 4 depicts the results for degree, clustering coefficient and k–core distributions. From the results, one may say that for all three statistics the algorithm ICLA-NS obtains the lowest KS distance for the learning rate of 0.1 and the highest KS distance for the learning rate of 0.3. For this reason, for the experiments that follow we set the learning rate a to 0.1.

Experiment II

In this experiment, we study the impact of the mechanism used for activating the cells of ICLA on the performance of the algorithm ICLA-NS. For this purpose, two following mechanisms are tested:

Mechanism I: The activation of the cells of ICLA according to the order that they appear in the queue.

Mechanism II: The activation of the cells of ICLA by the random selection of a cell from the set of all cells in the queue.



(a) Degree (b) Clustering Coefficient

(c) K-core Decomposition

Fig.4. The impact of learning rate a on ICLA-NS's efficiency in terms of average KS distance over all datasets.

Tables 2 and 3 show the impact of these two mechanisms on the performance of ICLA-NS in terms of the KS distance for degree, clustering coefficient and k–core distributions respectively for the sampling fractions 0.2 and 0.3. From these tables, we observe that for the sampling fraction of 0.2 both the mechanisms provide almost similar results for degree and k–core distributions in all datasets, except for *Erdos992* and *Netscience* that the mechanism I performs considerably better, and for clustering coefficient distribution, the mechanism I outperforms the mechanism II except for *Erdos992*. For the sampling fraction of 0.3, the mechanism I provides the better results than the mechanism II for all or majority of the statistics in all dataset except for *Advogato*. Therefore, we use the mechanism I in our proposed algorithm ICLA-NS in the next experiments.

Table 2. The impact of cell activation mechanism on ICLA-NS's efficiency in terms of KS distance, at sampling fraction 0.2

Statistic	*Deg*		*Clust*		*K Core*	
Dataset	Mechanism I	Mechanism II	Mechanism I	Mechanism II	Mechanism I	Mechanism II
CondMat	**0.0752**	0.0987	**0.2096**	0.3067	**0.1566**	0.1628
CMU	0.1919	**0.1908**	**0.1253**	0.2409	**0.4520**	0.4559
Advogato	**0.3306**	0.3314	**0.1086**	0.1236	**0.3335**	0.3345
Erdos992	**0.1512**	0.3291	0.1770	**0.1104**	**0.1342**	0.3291
Hamsterster	0.1478	**0.1308**	**0.2669**	0.2971	0.1726	**0.1724**
Netscience	**0.1230**	0.4309	**0.0945**	0.1887	**0.1865**	0.6000
Avg. for all Datasets	**0.1700**	0.2520	**0.1637**	0.2112	**0.2392**	0.3424

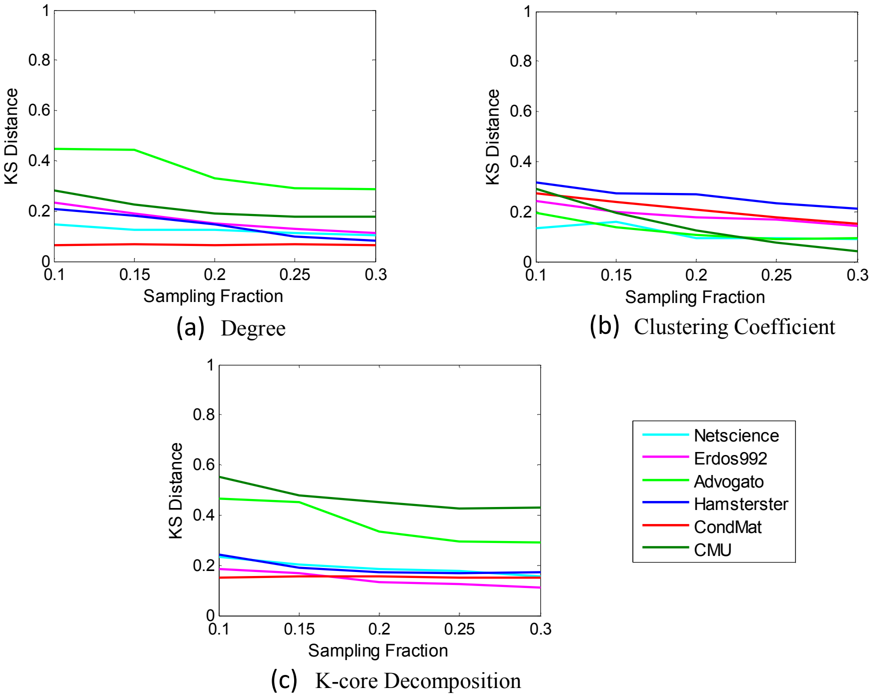
Experiment III

This experiment aims to investigate the impact of the sampling rate f on the performance of ICLA-NS in terms of the KS distance. We consider the statistics degree, clustering coefficient and k–core distributions, and for each statistic, we plot the KS distance for all the datasets at different sampling fractions f ranging from 0.1 to 0.3. As shown in Figure 5, the

KS distance decreases as the sampling rate increases. However, the decrement is not significant for *CondMat* for degree and k–core distributions. The highest KS distance for degree, clustering coefficient and k–core distributions is obtained respectively in *Advogato*, *Hamsterster* and *CMU* datasets.

Table 3. The impact of cell activation mechanism on ICLA-NS's efficiency in terms of KS distance, at sampling fraction 0.3

Statistic / Dataset	*Deg*		*Clust*		*K Core*	
	Mechanism I	Mechanism II	Mechanism I	Mechanism II	Mechanism I	Mechanism II
CondMat	**0.0792**	0.1212	**0.1490**	0.2484	**0.1607**	0.1897
CMU	**0.1782**	0.1804	**0.0419**	0.1430	**0.4283**	0.4298
Advogato	0.2873	**0.2420**	0.0953	**0.0899**	0.2918	**0.2622**
Erdos992	**0.1131**	0.3678	0.1428	**0.1113**	**0.1128**	0.3678
Hamsterster	**0.0801**	0.0825	**0.2138**	0.2179	0.1704	**0.1532**
Netscience	**0.1041**	0.4177	**0.0912**	0.1779	**0.1547**	0.6160
Avg. for all Datasets	**0.1403**	0.2353	**0.1223**	0.1647	**0.2198**	0.3364



(a) Degree

(b) Clustering Coefficient

(c) K-core Decomposition

Fig.5. The impact of sampling rate on ICLA-NS's efficiency in terms of KS distance.

Experiment IV

This experiment aims to investigate the ability of the proposed algorithm ICLA-NS in preserving graph statistics compared to other sampling algorithms, including Node Sampling (NS), Random Walk Sampling (RWS), Forest Fire Sampling (FFS), and also Distributed Learning Automata based Sampling (DLAS). The comparison is made in terms of the KS distance for three graph statistics. We report the results for the sampling fraction of 0.2 in Tables 4, 5 and 6 and for the sampling fraction of 0.3 in Tables 7, 8 and 9. Based on the results, ICLA-NS considerably outperforms other sampling algorithms for degree distribution in all datasets, except for *Advogato* and *Netscience* in which ICLA-NS provides the results close to the best ones. As noted in section 4, ICLA-NS has a selection bias to high degree nodes that improves estimates of the sampled degree distribution. For k–core distribution, ICLA-NS also performs the best in majority of the test datasets and close to the best ones in the rest of datasets. However, ICLA-NS performs slightly better on average than DLAS and FFS for clustering coefficient distribution. ICLA-NS provides the best results for this statistic only in *Advogato*, *Erdos992* and *Netscience* datasets. Generally, DLAS performs almost similar to FFS. NS and RWS perform the worst on average among the five methods for all statistics.

Table 10 reports the average KS distance taken over degree, clustering coefficient and k–core distributions for all datasets. According to the results of this table, we can conclude that the proposed algorithm ICLA-NS preserves more accurately the graph statistics for all test datasets comparing to other sampling algorithms.

In this experiment, we also investigate the ability of ICLA-NS for preserving the local density in sampled subgraphs in comparison to other sampling algorithms. For this purpose, the maximum core number in subgraphs sampled by the sampling algorithms is compared to real maximum core number for each dataset at the sampling fraction 0.2. The maximum core number is defined as the maximum value of k in k–core distribution. As shown in Table 11, ICLA-NS captures the local density in subgraphs sampled from *CondMat*, *Advogato* and *Erdos992*. For *CMU*, ICLA-NS performs better than the other sampling methods. ICLA-NS is as good as FFS for *Hamsterster,* and as good as FFS and DLAS for *Netscience*.

Table 4. KS distance for degree distribution for all datasets, at sampling fraction 0.2

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.5729	0.4358	0.1660	0.4165	**0.0752**
CMU	0.5505	0.4682	0.2945	0.2813	**0.1919**
Advogato	**0.2827**	0.3023	0.4213	0.3467	0.3306
Erdos992	0.6826	0.1714	0.2227	0.2849	**0.1512**
Hamsterster	0.4608	0.4857	0.2990	0.2804	**0.1478**
Netscience	0.6737	0.2201	0.1357	**0.1082**	0.1230
Avg. for all Datasets	0.5372	0.3473	0.2565	0.2863	**0.1700**

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

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.2444	0.7106	**0.1432**	0.1800	0.2096
CMU	0.1484	0.0884	0.2161	**0.0834**	0.1253
Advogato	0.2010	0.7779	0.1678	0.1334	**0.1086**
Erdos992	0.3893	0.3703	0.2055	0.1954	**0.1770**
Hamsterster	**0.2244**	0.8181	0.2893	0.2668	0.2669
Netscience	0.3162	0.3528	0.0947	0.2129	**0.0945**
Avg. for all Datasets	0.2540	0.5197	0.1861	0.1787	**0.1637**

Table 6. KS distance for k–core distribution for all datasets, at sampling fraction 0.2

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.6769	0.6740	0.2406	0.4744	**0.1566**
CMU	0.7297	0.6719	0.5340	0.5111	**0.4520**
Advogato	**0.3163**	0.3979	0.4351	0.3504	0.3335
Erdos992	0.6826	**0.1256**	0.2290	0.2577	0.1342
Hamsterster	0.5531	0.6813	0.3392	0.3180	**0.1726**
Netscience	0.8195	0.4110	0.2096	0.2321	**0.1865**
Avg. for all Datasets	0.6297	0.4936	0.3312	0.3573	**0.2392**

Table 7. KS distance for degree distribution for all datasets, at sampling fraction 0.3

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.4616	0.3823	0.1682	0.3619	**0.0792**
CMU	0.4364	0.3764	0.2334	0.2664	**0.1782**
Advogato	**0.2434**	0.2940	0.4085	0.3257	0.2873
Erdos992	0.5789	0.1337	0.1694	0.1769	**0.1131**
Hamsterster	0.3770	0.4033	0.2847	0.2011	**0.0801**
Netscience	0.5991	0.2044	**0.0724**	0.0810	0.1041
Avg. for all Datasets	0.4494	0.2990	0.2228	0.2355	**0.1403**

Table 8. KS distance for clustering coefficient distribution for all datasets, at sampling fraction 0.3

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.1850	0.6586	**0.1312**	0.1514	0.1490
CMU	0.0914	0.0694	0.1543	**0.0399**	0.0419
Advogato	0.1747	0.7081	0.1171	0.1142	**0.0953**
Erdos992	0.2480	0.2668	0.1573	0.1480	**0.1428**
Hamsterster	**0.1507**	0.7961	0.2466	0.2010	0.2138
Netscience	0.2776	0.3143	0.0936	0.1617	**0.0912**
Avg. for all Datasets	0.1879	0.4689	0.1500	0.1360	**0.1223**

Table 9. KS distance for k–core distribution for all datasets, at sampling fraction 0.3

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.5680	0.6063	0.2432	0.4268	**0.1607**
CMU	0.6320	0.5806	0.4679	0.4705	**0.4283**
Advogato	**0.2639**	0.3277	0.4200	0.3304	0.2918
Erdos992	0.5789	0.1239	0.1745	0.1613	**0.1128**
Hamsterster	0.4488	0.6253	0.3258	0.2134	**0.1704**
Netscience	0.7367	0.3895	0.1404	**0.1196**	0.1547
Avg. for all Datasets	0.5380	0.4422	0.2953	0.2870	**0.2198**

Table 10. Average KS distance for all datasets, at sampling fraction 0.2

Dataset	NS	RW	FFS	DLAS	ICLA-NS
CondMat	0.4981	0.6068	0.1833	0.3570	**0.1471**
CMU	0.4762	0.4095	0.3482	0.2919	**0.2564**
Advogato	0.2667	0.4927	0.3414	0.2768	**0.2576**
Erdos992	0.5848	0.2224	0.2191	0.2460	**0.1541**
Hamsterster	0.4128	0.6617	0.3092	0.2884	**0.1958**
Netscience	0.6031	0.3280	0.1467	0.1844	**0.1347**
Avg. for all Datasets	0.4736	0.4535	0.2580	0.2741	**0.1910**

Table 11. Maximum core number for sampling algorithms versus real maximum core number of original graph, at sampling fraction 0.2

Dataset	Real max core no.	NS	RW	FFS	DLAS	ICLA-NS
CondMat	25	8	4	23	9	**25**
CMU	69	15	20	44	32	**67**
Advogato	25	7	3	23	22	**25**
Erdos992	7	2	3	**7**	**7**	**7**
Hamsterster	24	8	2	**21**	19	**21**
Netscience	8	2	4	**6**	**6**	**6**

Experiment V

In this experiment, we aims to investigate whether the statistics in subgraphs sampled by the proposed algorithm ICLA-NS overestimate or underestimate the statistics of the original graph. For this purpose, we plot degree, clustering coefficient and k–core distributions at sampling fraction 0.2 for the proposed algorithm and other sampling algorithms. Figures $6-11$ indicate the plots for all the distributions across all datasets. Note that in the figures, $P(X > x)$ refers to *CCDF*, and $P(X < x)$ refers to *CDF*.

As shown in these figures, the proposed algorithm ICLA-NS captures the tail of degree distribution for *Advogato* and *Hamsterster* better than other methods. ICLA-NS underestimates the low degrees for *Advogato*, *Erdos992* and *Hamsterster*, overestimates them for *Netscience,* and captures them for *CondMat* and *CMU*. FFS captures the high degrees and underestimates the low degrees for all datasets except for *CMU* and *Netscience.* DLAS performs similar to FFS for *Advogato* and *Hamsterster* and underestimates the degree distribution for other datasets. RWS underestimates the degree distribution for all datasets. NS underestimates the degree distribution for all datasets. ICLA-NS preserves the clustering coefficient distribution for *Netscience* more accurately than other methods. Similar to ICLA-NS, DLAS and FFS underestimate the low clustering coefficients for all datasets except for *CondMat* and *Netscience* and overestimate the high clustering coefficients for all datasets except for *Erdos992.* RWS and NS underestimates the clustering coefficient distribution for all datasets. ICLA-NS captures the k–core distribution for *CondMat*, *CMU*, *Advogato, Erdos992* and *Hamsterster* better than other methods. Similar to ICLA-NS, DLAS and FFS underestimate the core structures for *Netscience* and overestimate them for *Advogato, Erdos992* and *Hamsterster.* RWS underestimates the k–core distribution for *CondMat*, *CMU* and *Netscience*. NS underestimates the k–core distribution for all datasets.

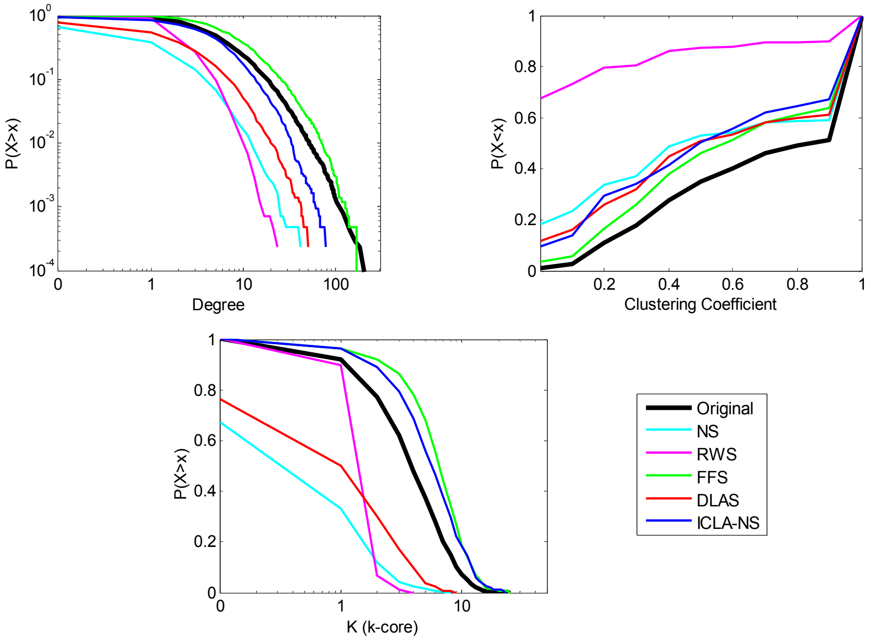


Fig.6. Comparing sampling algorithms on *CondMat* dataset for different distribution statistics, at sampling fraction 0.2.

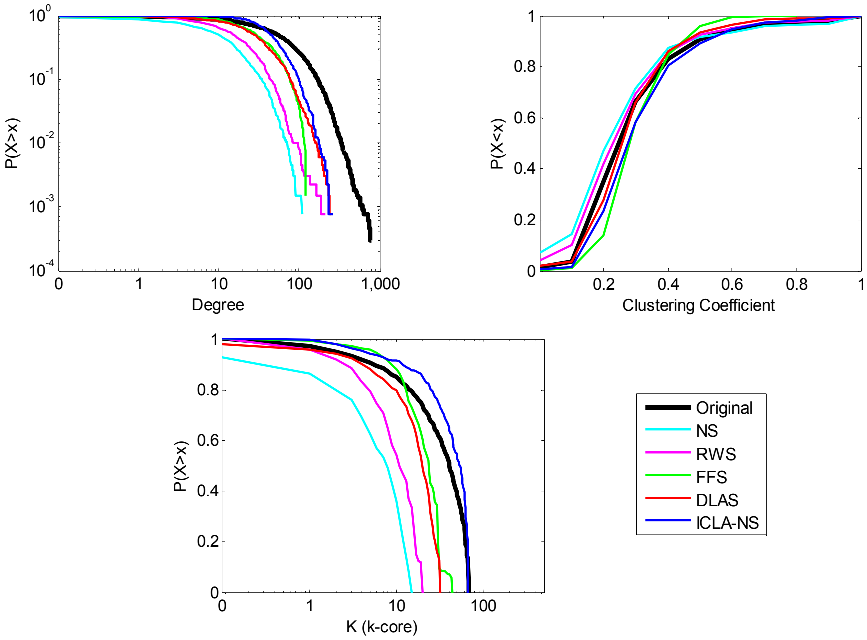


Fig.7. Comparing sampling algorithms on *CMU* dataset for different distribution statistics, at sampling fraction 0.2.

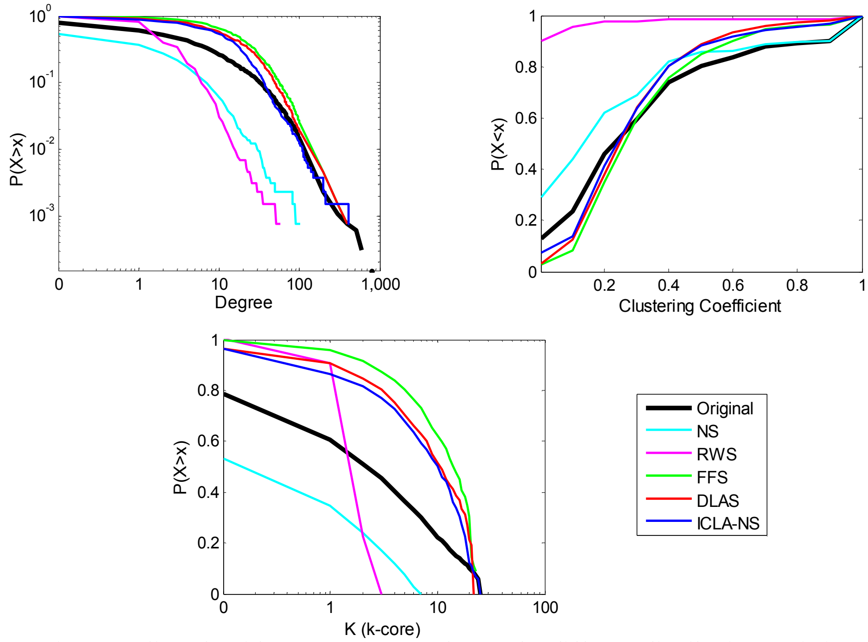


Fig.8. Comparing sampling algorithms on *Advogato* dataset for different distribution statistics, at sampling fraction 0.2.

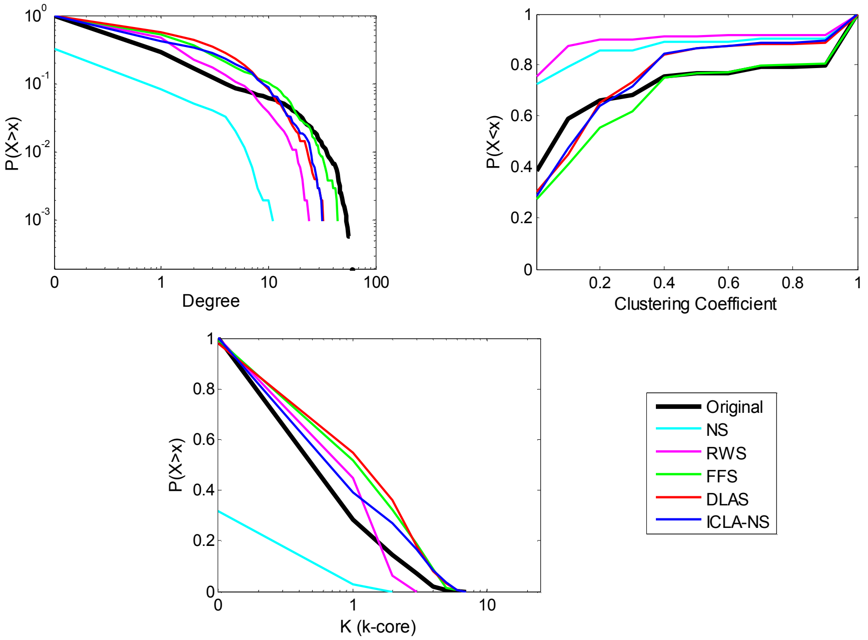


Fig.9. Comparing sampling algorithms on *Erdos992* dataset for different distribution statistics, at sampling fraction 0.2.

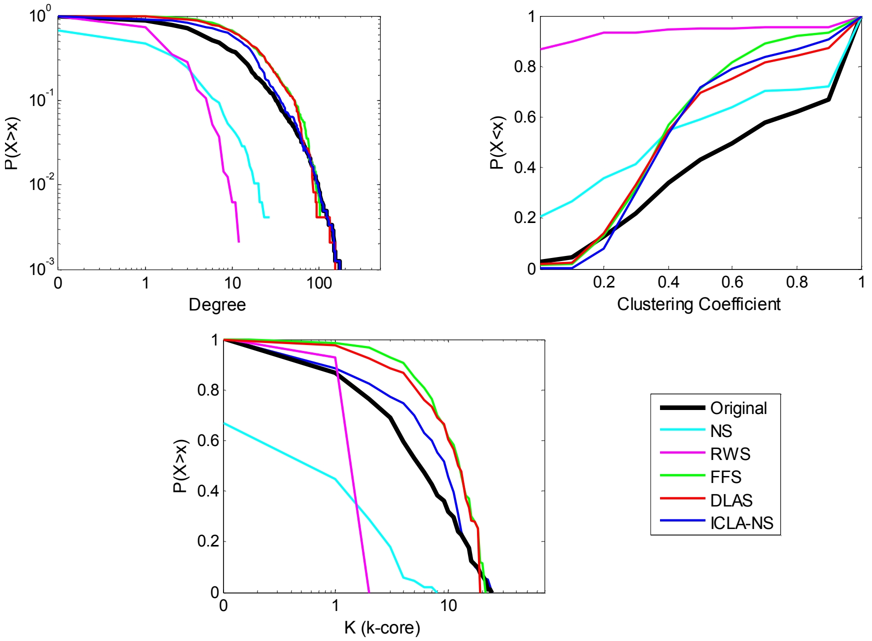


Fig.10. Comparing sampling algorithms on *Hamsterster* dataset for different distribution statistics, at sampling fraction 0.2.

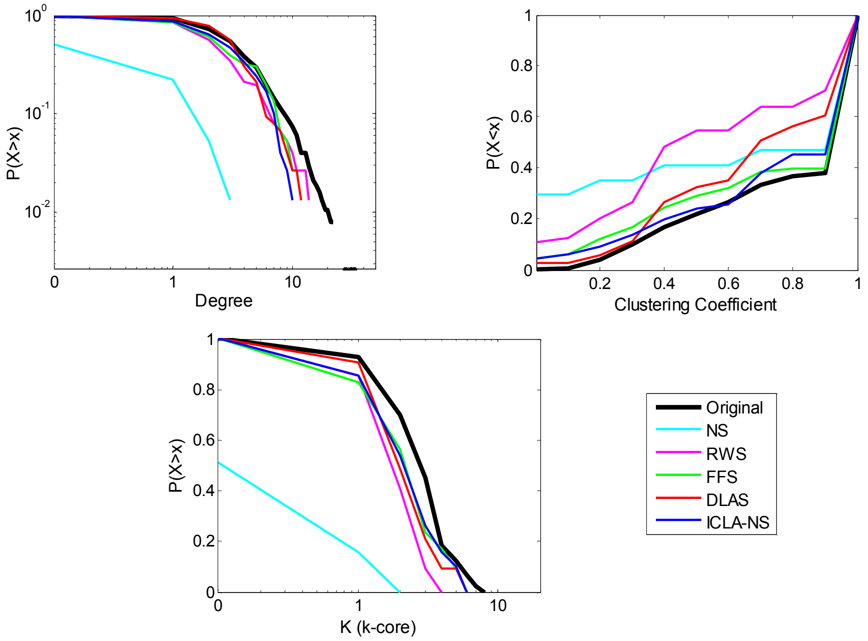


Fig.11. Comparing sampling algorithms on *Netscience* dataset for different distribution statistics, at sampling fraction 0.2.

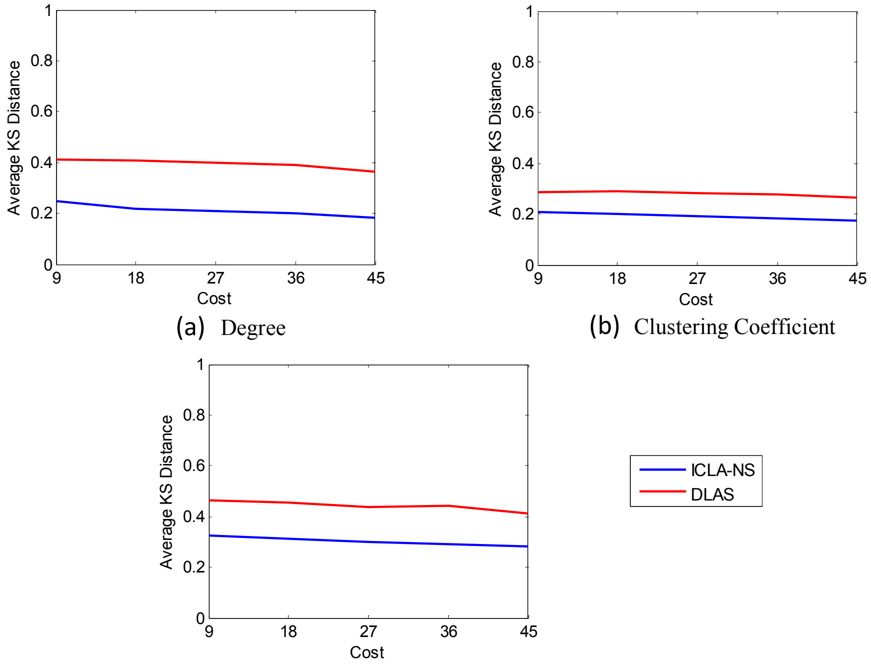
Experiment VI

This experiment aims to show the impact of increasing cost on the performance of ICLA-NS comparing to DLAS algorithm (as an example of an extended sampling algorithm with a pre–processing phase). For this purpose, we use the following definition for the cost of an extended sampling algorithm:

$$Cost = \frac{|V_{visited}|}{|V|} \qquad (12)$$

where $|V_{visited}|$ is the total number of times that the nodes of the input graph are visited during the pre/post processing phase of the algorithm, and $|V|$ is the number of nodes in the input graph. Actually, the cost function computes the additional processing cost that an extended sampling algorithm spends in the pre/post processing phase to produce subgraphs with the higher quality. In this experiment, we change the cost from 9 to 45 with step 9 and for each value of the cost, we report the KS distance averaged over all datasets for degree, clustering coefficient and k–core distributions at the sampling fraction 0.2. From Figure 12, we can see that increasing the cost results in the decrease of the KS distance for both extended sampling algorithms ICLA-NS and DLAS. For a same cost, CLAS-NS always provides better results as compared to DLAS.

We also compare ICLA-NS with the simple sampling algorithms FFS, RWS and NS in terms of average KS distance over all datasets. Table 12 reports the results for degree, clustering coefficient and k–core distributions at sampling fraction of 0.2. According to the results of this table, Even though the proposed algorithm ICLA-NS has higher time complexity comparing to the simple sampling algorithms for a same sample size, it preserves more accurately the graph statistics.



(a) Degree (b) Clustering Coefficient

(c) K-core Decomposition

Fig.12. Comparing extended sampling algorithms in terms of average KS distance versus cost, at sampling fraction 0.2.

Table 12. Comparing ICLA-NS with simple sampling algorithms in terms of average KS distance at sampling fraction 0.2.

Statistic	NS	RW	FFS	ICLA-NS
Deg	0.5372	0.3473	0.2565	**0.1700**
Clust	0.2540	0.5197	0.1861	**0.1673**
K Core	0.6297	0.4936	0.3312	**0.2392**

Experiment VII

This experiment is conducted to study the learning ability of ICLA-NS and its impact on the quality of sampled subgraphs in terms of the KS distance for degree, clustering coefficient and k–core distributions. To do this, we consider another version of ICLA-NS algorithm, called PC-NS, in which each learning automaton in ICLA is replaced by a pure –chance automaton. A pure–chance automaton is an automaton that always selects its actions with equal probabilities and is used as the comparison standard for investigating the efficiency of the learning process (a learning automaton must do at least better than such a pure–chance automaton) (Narendra and Thathachar, 2012). We compare the performances of both algorithms in terms of the KS distance for all datasets and report the results in Tables 13 and 14 respectively for the sampling fractions 0.2 and 0.3. According to the results, ICLA-NS performs considerably better than PC-NS for degree distribution in all datasets except for *Advogato*. For clustering coefficient distribution, ICLA-NS also outperforms PC-NS in all datasets except for *Hamsterster.* ICLA-NS preserves k–core distribution better than PC-NS in all test datasets except for *Advogato* at the sampling fraction of 0.2.

We also investigate the number of isolated nodes in subgraphs sampled by the ICLA algorithm. We compare the probability of isolated nodes for ICLA-NS to that for PC-NS, and other sampling algorithms. As shown in Table 15, for all datasets the algorithms PC-NS, DLAS and NS produce samples which include isolated nodes. This is because these algorithms sample the nodes independently. In contrast, ICLA-NS, FFS and RWS produce connected subgraphs, especially for *Advogato* that includes some isolated nodes.

Table 13. The impact of the learning process on ICLA-NS's efficiency in terms of KS distance, at sampling fraction 0.2

Statistic	*Deg*		*Clust*		*K Core*	
Dataset	ICLA-NS	PC-NS	ICLA-NS	PC-NS	ICLA-NS	PC-NS
CondMat	**0.0752**	0.5527	**0.2096**	0.2550	**0.1566**	0.6207
CMU	**0.1919**	0.5186	**0.1253**	0.1791	**0.4520**	0.6109
Advogato	0.3306	**0.2326**	**0.1086**	0.1972	0.3335	**0.2811**
Erdos992	**0.1512**	0.6021	**0.1770**	0.3824	**0.1342**	0.6021
Hamsterster	**0.1478**	0.4560	0.2669	**0.2057**	**0.1726**	0.5172
Netscience	**0.1230**	0.6374	**0.0945**	0.4049	**0.1865**	0.6670
Avg. for all Datasets	**0.1700**	0.4999	**0.1637**	0.2707	**0.2392**	0.5498

Table 14. The impact of the learning process on ICLA-NS's efficiency in terms of KS distance, at sampling fraction 0.3

Statistic	*Deg*		*Clust*		*K Core*	
Dataset	ICLA-NS	PC-NS	ICLA-NS	PC-NS	ICLA-NS	PC-NS
CondMat	**0.0792**	0.4836	**0.1490**	0.2129	**0.1607**	0.5582
CMU	**0.1782**	0.4602	**0.0419**	0.1297	**0.4283**	0.5756
Advogato	0.2873	**0.2414**	**0.0953**	0.1649	**0.2918**	0.3095
Erdos992	**0.1131**	0.5771	**0.1428**	0.3071	**0.1128**	0.5771
Hamsterster	**0.0801**	0.4045	0.2138	**0.2096**	**0.1704**	0.4590
Netscience	**0.1041**	0.5869	**0.0912**	0.2833	**0.1547**	0.6593
Avg. for all Datasets	**0.1403**	0.4589	**0.1223**	0.2179	**0.2198**	0.5231

Table 15. Comparing sampling algorithms in terms of the probability of isolated nodes, at sampling fraction 0.2

Dataset	Real probability	NS	DLAS	PC-NS	ICLA-NS, FFS, RWS
CondMat	0	0.3267	0.2371	0.5470	**0**
CMU	0	0.0733	0.0211	0.5000	**0**
Advogato	0.2113	0.4672	0.0351	0.4748	**0**
Erdos992	0	0.6811	0.0236	0.8027	**0**
Hamsterster	0	0.3299	0.0082	0.4550	**0**
Netscience	0	0.5208	0.0132	0.5921	**0**

6. Conclusion

In this paper, we proposed a topology–based node sampling algorithm called ICLA-NS for producing representative subgraphs from social networks. The proposed algorithm is in fact an extended sampling algorithm with post–processing phase, since it utilizes an irregular cellular learning automata (ICLA) to guarantee the connectivity and the inclusion of the high degrees nodes in subgraphs initially sampled by classic node sampling method. We investigated the effectiveness of the proposed sampling algorithm by conducting a number of experiments on real–world networks. Our experimental results demonstrate that the properties of sampled subgraphs created by the proposed algorithm ICLA-NS are the more similar to those of the original graph as comparing to the existing sampling methods in terms of Kolmogorov-Smirnov (KS) test for degree, clustering coefficient, and k–core distributions.

For future work, we aim to extend our proposed method ICLA-NS to be able to sample representative subgraphs from weighted graphs, as opposed to the current work in which edge weights are binary.

References

Ahmed, N.K., Berchmans, F., Neville, J., Kompella, R., 2010. Time-based Sampling of Social Network Activity Graphs, in: Proceedings of the Eighth Workshop on Mining and Learning with Graphs, MLG '10. ACM, New York, NY, USA, pp. 1–9. doi:10.1145/1830252.1830253

Ahmed, N.K., Neville, J., Kompella, R., 2013. Network Sampling: From Static to Streaming Graphs. ACM Trans. Knowl. Discov. Data 8, 7:1–7:56. doi:10.1145/2601438

Albert, R., Barabási, A.-L., 2002. Statistical mechanics of complex networks. Rev. Mod. Phys. 74, 47–97. doi:10.1103/RevModPhys.74.47

Alvarez-Hamelin, J.I., Dall'Asta, L., Barrat, A., Vespignani, A., 2005. K-core decomposition of Internet graphs: hierarchies, self-similarity and measurement biases. arXiv:cs/0511007.

Avrachenkov, K., Ribeiro, B., Towsley, D., 2010. Improving Random Walk Estimation Accuracy with Uniform Restarts, in: Kumar, R., Sivakumar, D. (Eds.), Algorithms and Models for the Web-Graph, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 98–109.

Beigy, H., Meybodi, M.R., 2004. A mathematical framework for cellular learning automata. Advances Complex System 3, 295–319.

Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., Shir, E., 2007. A model of Internet topology using k-shell decomposition. PNAS 104, 11150–11154. doi:10.1073/pnas.0701175104

Dorogovtsev, S.N., Mendes, J.F.F., 2003. Evolution of Networks: From Biological Nets to the Internet and WWW (Physics). Oxford University Press, Inc., New York, NY, USA.

Ebbes, P., Huang, Z., Rangaswamy, A., 2013. Subgraph Sampling Methods for Social Networks: The Good, the Bad, and the Ugly (SSRN Scholarly Paper No. ID 1580074). Social Science Research Network, Rochester, NY.

Esnaashari, M., Meybodi, M.R., 2008a. A Cellular Learning Automata Based Clustering Algorithm for Wireless Sensor Networks. Sensor Letters 6, 723–735. doi:10.1166/sl.2008.m146

Esnaashari, M., Meybodi, M.R., 2008b. Dynamic Point Coverage in Wireless Sensor Networks: A Learning Automata Approach, in: Sarbazi-Azad, H., Parhami, B., Miremadi, S.-G., Hessabi, S. (Eds.), Advances in Computer Science and Engineering, Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 758–762.

Even, S., 2011. Graph Algorithms. Cambridge University Press.

Fang, M., Yin, J., Zhu, X., 2013. Active Exploration: Simultaneous Sampling and Labeling for Large Graphs, in: Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13. ACM, New York, NY, USA, pp. 829–834. doi:10.1145/2505515.2505618

Fang, M., Yin, J., Zhu, X., 2015. Active exploration for large graphs. Data Min Knowl Disc 30, 511–549. doi:10.1007/s10618-015-0424-z

Fang, M., Yin, J., Zhu, X., 2016. Supervised sampling for networked data. Signal Processing, Big Data Meets Multimedia Analytics — Containing a selection of papers from the 21st International Conference on Multimedia Modelling (MMM2015) 124, 93–102. doi:10.1016/j.sigpro.2015.09.040

Ghavipour, M., Meybodi, M.R., 2016. An adaptive fuzzy recommender system based on learning automata. Electronic Commerce Research and Applications 20, 105–115. doi:10.1016/j.elerap.2016.10.002

Gkantsidis, C., Mihail, M., Saberi, A., 2006. Random walks in peer-to-peer networks: Algorithms and evaluation. Performance Evaluation, P2P Computing Systems 63, 241–263. doi:10.1016/j.peva.2005.01.002

Goel, S., Salganik, M.J., 2010. Assessing respondent-driven sampling. PNAS 107, 6743–6747. doi:10.1073/pnas.1000261107

Goldstein, M.L., Morris, S.A., Yen, G.G., 2004. Problems with fitting to the power-law distribution. Eur. Phys. J. B 41, 255–258. doi:10.1140/epjb/e2004-00316-5

Goodman, L.A., 1961. Snowball Sampling. The Annals of Mathematical Statistics 32, 148–170.

Heckathorn, D.D., 1997. Respondent-Driven Sampling: A New Approach to the Study of Hidden Populations. Social Problems 44, 174–199. doi:10.2307/3096941

Henzinger, M.R., Heydon, A., Mitzenmacher, M., Najork, M., 2000. On near-uniform URL sampling. Computer Networks 33, 295–308. doi:10.1016/S1389-1286(00)00055-4

Illenberger, J., Kowald, M., Axhausen, K.W., Nagel, K., 2011. Insights into a spatially embedded social network from a large-scale snowball sample. Eur. Phys. J. B 84, 549–561. doi:10.1140/epjb/e2011-10872-0

Krishnamurthy, V., Faloutsos, M., Chrobak, M., Lao, L., Cui, J.-H., Percus, A.G., 2005. Reducing Large Internet Topologies for Faster Simulations, in: Boutaba, R., Almeroth, K., Puigjaner, R., Shen, S., Black, J.P. (Eds.), NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 328–341.

Kumar, R., Novak, J., Tomkins, A., 2010. Structure and Evolution of Online Social Networks, in: Yu, P.S., Han, J., Faloutsos, C. (Eds.), Link Mining: Models, Algorithms, and Applications. Springer New York, pp. 337–357.

Kurant, M., Gjoka, M., Butts, C.T., Markopoulou, A., 2011. Walking on a Graph with a Magnifying Glass: Stratified Sampling via Weighted Random Walks, in: Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '11. ACM, New York, NY, USA, pp. 281–292. doi:10.1145/1993744.1993773

Lee, S.H., Kim, P.-J., Jeong, H., 2006. Statistical properties of sampled networks. Phys. Rev. E 73, 016102. doi:10.1103/PhysRevE.73.016102

Lee, C.-H., Xu, X., Eun, D.Y., 2012. Beyond Random Walk and Metropolis-hastings Samplers: Why You Should Not Backtrack for Unbiased Graph Sampling, in: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '12. ACM, New York, NY, USA, pp. 319–330. doi:10.1145/2254756.2254795

Leskovec, J., Faloutsos, C., 2006. Sampling from Large Graphs, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06. ACM, New York, NY, USA, pp. 631–636. doi:10.1145/1150402.1150479

Li, Y., Wu, C., Luo, P., Zhang, W., 2013. Exploring the Characteristics of Innovation Adoption in Social Networks: Structure, Homophily, and Strategy. Entropy 15, 2662–2678. doi:10.3390/e15072662

Lovász, L., 1993. Random walks on graphs: A survey. Combinatorics, Paul Erdos is Eighty 2, 1–46.

Lu, J., Li, D., 2012. Sampling Online Social Networks by Random Walk, in: Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research, HotSocial '12. ACM, New York, NY, USA, pp. 33–40. doi:10.1145/2392622.2392628

Maiya, A.S., Berger-Wolf, T.Y., 2010. Sampling Community Structure, in: Proceedings of the 19th International Conference on World Wide Web, WWW '10. ACM, New York, NY, USA, pp. 701–710. doi:10.1145/1772690.1772762

Moradabadi, B., Meybodi, M.R., 2016. Link prediction based on temporal similarity metrics using continuous action set learning automata. Physica A: Statistical Mechanics and its Applications 460, 361–373. doi:10.1016/j.physa.2016.03.102

Narendra, K.S., Thathachar, M.A.L., 2012. Learning Automata: An Introduction. Courier Corporation.

Newman, M.E.J., 2003a. The Structure and Function of Complex Networks. SIAM Rev. 45, 167–256. doi:10.1137/S003614450342480

Newman, M.E.J., 2003b. Ego-centered networks and the ripple effect. Social Networks 25, 83–95. doi:10.1016/S0378-8733(02)00039-4

Rasti, A.H., Torkjazi, M., Rejaie, R., Duffield, N., Willinger, W., Stutzbach, D., 2009. Respondent-Driven Sampling for Characterizing Unstructured Overlays, in: IEEE INFOCOM 2009. Presented at the IEEE INFOCOM 2009, pp. 2701–2705. doi:10.1109/INFCOM.2009.5062215

Rezapoor Mirsaleh, M., Meybodi, M.R., 2016. A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem. Memetic Comp. 1–12. doi:10.1007/s12293-016-0183-4

Rezvanian, A., Meybodi, M.R., 2015. Sampling social networks using shortest paths. Physica A: Statistical Mechanics and its Applications 424, 254–268. doi:10.1016/j.physa.2015.01.030

Rezvanian, A., Rahmati, M., Meybodi, M.R., 2014. Sampling from complex networks using distributed learning automata. Physica A: Statistical Mechanics and its Applications 396, 224–234. doi:10.1016/j.physa.2013.11.015

Ribeiro, B., Towsley, D., 2010. Estimating and Sampling Graphs with Multidimensional Random Walks, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10. ACM, New York, NY, USA, pp. 390–403. doi:10.1145/1879141.1879192

Rossi, R.A., Ahmed, N.K., 2013. Network repository, Purdue University, Computer Science Department. Available: http://www.networkrepository.com.

Seshadhri, C., Pinar, A., Kolda, T.G., 2013. An In-depth Analysis of Stochastic Kronecker Graphs. J. ACM 60, 13:1–13:32. doi:10.1145/2450142.2450149

So, R.J., Long, H., 2013. Network Analysis and the Sociology of Modernism. boundary 2 40, 147–182. doi:10.1215/01903659-2151839

Stumpf, M.P.H., Wiuf, C., May, R.M., 2005. Subnets of scale-free networks are not scale-free: Sampling properties of networks. PNAS 102, 4221–4224. doi:10.1073/pnas.0501179102

Stutzbach, D., Rejaie, R., Duffield, N., Sen, S., Willinger, W., 2009. On Unbiased Sampling for Unstructured Peer-to-peer Networks. IEEE/ACM Trans. Netw. 17, 377–390. doi:10.1109/TNET.2008.2001730

Thathachar, M.A.L., Sastry, P.S., 2011. Networks of Learning Automata: Techniques for Online Stochastic Optimization. Springer Science & Business Media.

Wolfram, S., 1983. Statistical mechanics of cellular automata. Rev. Mod. Phys. 55, 601–644. doi:10.1103/RevModPhys.55.601

Yang, L.-X., Yang, X., Liu, J., Zhu, Q., Gan, C., 2013. Epidemics of computer viruses: A complex-network approach. Applied Mathematics and Computation 219, 8705–8717. doi:10.1016/j.amc.2013.02.031

Yoon, S.-H., Kim, K.-N., Hong, J., Kim, S.-W., Park, S., 2015. A community-based sampling method using DPL for online social networks. Information Sciences 306, 53–69. doi:10.1016/j.ins.2015.02.014

Yoon, S., Lee, S., Yook, S.-H., Kim, Y., 2007. Statistical properties of sampled networks by random walks. Phys. Rev. E 75, 046114. doi:10.1103/PhysRevE.75.046114