Clustering-structure representative sampling from graph streams

Clustering-structure Representative Sampling from Graph Streams

Jianpeng Zhang, Kaijie Zhu, Yulong Pei, George Fletcher, and Mykola Pechenizkiy

Abstract Most existing sampling algorithms on graphs (i.e., network-structured data) focus on sampling from memory-resident static graphs and assume the entire graphs are always available. However, the graphs encountered in modern applications are often too large and/or too dynamic to be processed with limited memory. Furthermore, existing sampling techniques are inadequate for preserving the inherent clustering structure, which is an essential property of complex networks. To tackle these problems, we propose a new sampling algorithm that dynamically maintains a representative sample and is capable of retaining clustering structure in graph streams at any time. Performance of the proposed algorithm is evaluated through empirical experiments using real-world networks. The experimental results have shown that our proposed *CPIES* algorithm can produce clustering-structure representative samples and outperforms current online sampling algorithms.

1 Introduction

An increasing number of networks are large-scale and continuously growing in nature. Hence, modeling and analyzing such data in their entirety is becoming infeasible and impractical. One approach to overcome these features of contemporary graph-structured data collections is to sample a representative subgraph and exploit its characteristics. In the existing literature, many sampling methods [3][6][7] focus on sampling from memory-resident static graphs and assume that the sampling algorithms can access the entire graph by multiple passes. However, many contem-

Jianpeng Zhang
Eindhoven University of Technology, 5600 MB Eindhoven, the Netherlands and NDSC, 450002, Zhengzhou, China, e-mail: j.zhang.4@tue.nl

Kaijie Zhu
National Digital Switching System Engineering & Technological R&D Center (NDSC), 450002, Zhengzhou, China e-mail: kaijiezhu.ndsc@gmail.com

Yulong Pei, George H. L. Fletcher and Mykola Pechenizkiy
Eindhoven University of Technology, 5600 MB Eindhoven, the Netherlands e-mail: {y.pei.1, g.h.l.fletcher,m.pechenizkiy}@tue.nl

porary networks can naturally be represented as fully-dynamic graph streams, i.e., nodes or edges are added or removed arbitrarily at any time [10]. To keep pace with such kinds of graphs, we need an incremental sampling method to analyze the interactions within graphs when new entities arrive in a streaming fashion.

Thus there are two research questions which should be addressed. The first challenge is to design an effective and efficient sampling algorithm on fully-dynamic graph streams where edge insertions/deletions are processed in an incremental manner with limited memory. The second challenge is to generate small, yet representative samples of open-ended graph streams. A sampled graph is representative if it preserves selected properties of the original graph. In general, topological properties, such as degree distribution, clustering coefficient distribution and maximum cluster size, are of interest to data scientists. However, existing sampling techniques are inadequate for preserving the inherent clustering structure. To overcome these challenges, we make the following contributions in this paper:

- We propose a *C*lustering-preserving *P*artially *I*nduced *E*dge *S*ampling (*CPIES*) algorithm to process the fully-dynamic graph streams. *CPIES* employs: (i) a clustering-preserving node replacement, (ii) isolated nodes elimination, and (iii) edge-deletion operation. It can retain the inherent clustering structure well, and eliminate the isolated nodes in the sampled counterpart. Moreover, it can handle the edge-deletion requests which is crucial in fully-dynamic setting.
- The empirical experiments on real-world networks show that *CPIES* is capable of keeping representative/hub nodes rather than peripheral nodes from different clusters in graph streams. It outperforms the state-of-the-art online sampling algorithms in terms of preserving clustering structure.

The remainder of the paper is organized as follows. In Section 2 we provide a brief review of existing work in the areas of graph stream sampling and then give the problem definition in Section 3. In Section 4 the detailed description of the proposed streaming sampling algorithm is given. Experimental evaluation is given in Section 5. The paper is concluded and future work is presented in Section 6.

2 Related Work

2.1 Sampling from Graph Streams

Graph streams differ from static graphs in three main aspects: (i) the massive volume of edges is far too large to fit into the limited memory; (ii) the topology structure is not fully observable at any point of time (i.e., only sequential access is feasible, not random access); and (iii) efficient, real-time processing is crucial [3]. Motivated by real-world applications, there are some related research on sampling from massive graph streams. Ahmed et al. [1] proposed a generic stream sampling framework for big-graph analytics, called *Graph Sample and Hold* (*gSH*). It samples from massive graphs sequentially in a single pass, one edge at a time, while maintaining a small memory footprint. Ahmed et al. [3] extended node sampling, edge sampling and breadth first sampling into streaming setting, and presented a *partially-induced edge sampling* (*PIES*) algorithm to sample from graph streams, which maintains

a sample reservoir of fixed size. A graph priority sampling (*GPS*) was proposed in [2] for order-based reservoir sampling from massive graph streams. It provides a general way to sample weighted edges according to auxiliary variables so as to accomplish various estimation goals of graph properties. However, it is difficult to use it to sample the inherent clustering structure from fully-dynamic graph streams.

2.2 Clustering Structure in Graphs

A cluster in a graph is defined as a set of densely connected nodes that are sparsely connected to nodes outside of the cluster [4]. The discovery of clustering structure in graphs is important as they often correspond to common/latent properties (e.g., interest, role and affiliation) [5]. Although sampling provides a potential solution for inferring and approximating global, latent properties in original graphs, most sampling techniques are not particularly designed to retain the essential property: the inherent clustering structure [12]. To our knowledge, sampling has not previously been applied to the problem of preserving the clustering structure in graph streams.

The work in [6][7] only assessed the degree to which samples are representative of explicit or simple graph properties (e.g., the degree distribution), which can not fully reflect the topology structure. Maiya et al. [8] proposed two sampling algorithm based on the notion that samples with good expansion properties tend to be more representative of the clustering structure. Newly sampled nodes are chosen either deterministically or probabilistically and the process is continued until we reach the desired subgraph size. However, these approaches are all designed for static graphs to preserve the clustering structure.

3 Problem Statement

We focus on the problem of sampling from a fully-dynamic graph stream where edge insertions and deletions are allowed. Formally, for any discrete time-stamp $t \geq 0$, the input is assumed to be a graph $G(t) = [V(t), E(t)]$, presented as a stream of edges $E(t)$ in arbitrary order, where $V(t)$ is a finite set of nodes and $E(t) \subseteq V(t) \times V(t)$ is a set of edges by time t. Each edge $e(t)$ is in the form of $\langle u, v, t \rangle$, where u and v are the two incident nodes of the edge and t is the associated time-stamp. Initially at time-stamp $t = 0$ we have $V(t) = E(t) = \emptyset$, and for any $t > 0$, at each discrete time-stamp t we receive a new update $e_t = (\bullet, \langle u, v, t \rangle)$ from the edge streams, where $\bullet \in \{+, -\}$. The graph $G(t) = [V(t), E(t)]$ at time-stamp t can be updated as follows:

$$E(t) = \begin{cases} E(t-1) \cup \langle u, v, t \rangle & \text{if } \bullet = \text{“+”} \\ E(t-1) \setminus \langle u, v, t \rangle & \text{if } \bullet = \text{“-”} \end{cases} \tag{1}$$

The aim of graph stream sampling is to generate representative samples, which should obtain a good sample quality, through a single pass on original graph streams. Formally, We denote $\eta(\cdot)$ as any topological graph property, and our objective is to ensure that $G_s(t)$ is representative, in which it matches many of the topological properties of G, i.e., $\eta(G(t)) \approx \eta(G_s(t))$. Specifically, in this paper we mainly consider the property of inherent clustering structure. In addition to the sample rep-

resentativeness requirement, a graph stream sampling algorithm is also required to be computationally efficient (preferably, single-pass) under the limited memory size χ. Formally, we give the problem statement of graph stream sampling. Note that in most cases the sample size of a graph can be defined as the number of nodes in the sample, but it is also feasible to define it as the number of edges in the sample. Here we only consider the former case.

Definition 1. Graph stream sampling: given a graph stream $G(t)$, the sampling algorithm $\mathscr{S}$ aims to produce a sampled graph $G_s(t)$ by sampling edges of the graph stream $G(t)$ such that:

- the edges are processed in a sequential order (i.e., not random access) through single pass;
- the memory should satisfy the restriction, i.e., $|V_s(t)| \leq \chi$.
- the topological properties, especially the clustering structure, should be representative and preserved well, i.e., $\eta(G(t)) \approx \eta(G_s(t))$;

4 Proposed Sampling Method

Firstly, we will introduce the state-of-the-art *PIES* (Partially Induced Edge Sampling) algorithm. Secondly, we will describe our new *CPIES* algorithm which employs: (i) a clustering-preserving node replacement, (ii) isolated nodes elimination, and (iii) edge-deletion operation to process the fully-dynamic graph streams.

4.1 The Basic PIES Algorithm

Basically, *PIES* [3] is a two-phase sampling algorithm using a single pass for the graph stream: Initially, initial edges in the stream are added to reservoir deterministically in order to accumulate the node reservoir V_s. After $|V_s|$ reaches the desired sample size n, we denote the number of edges in edge reservoir E_s by m. Then it consists of two phases to process the new edge e_t at time t:

- *Selection phase*: the new edge is added with probability $p = \frac{m}{t}$, where t is the current timestamp. If the probability is not satisfied, the new edge does not sample into the reservoir V_s. Otherwise, the process goes directly to replacement phase. The rationale is inherited from reservoir sampling and each edge in $E(t)$ has equal probability (i.e., $\frac{m}{t}$) of being chosen for the edge reservoir E_s.
- *Replacement phase*: If the new edge is selected and at least one incident node has not been sampled into V_s, the previously sampled nodes in the reservoir V_s are replaced based on a certain strategy (namely, $Select\_Replaced\_Node(\cdot)$) in order to maintain the desired size of V_s.

After these two phases, it adds the new edge to the edge reservoir E_s if its two incident nodes have already been in the node reservoir V_s (i.e., partial induction). After careful analysis, we found that the node replacement strategy $Select\_Replaced\_Node(\cdot)$ is an alterable module since it allows adopting various strategies to select node that needs to be replaced (i.e., replaceable node). Nesreen proposes two different replacement strategies in [3]. The first strategy selects the replaceable node uniformly at random from the reservoir V_s, while the second one requires to replace the node kept in V_s for the longest amount of time without acquiring more edges, named *PIES (Min)*. However, the current replacement strategies do not consider the clustering structure into account. Thus, we propose a new clustering-preserving replacement strategy to retain the hub nodes in clusters and replace the peripheral nodes to preserve the inherent structure.

4.2 Clustering-preserving Node Replacement

The basic rationale is that the centrality of the nodes in a cluster depends on their degree, and the hub node tend to have higher degree than that of its neighbors. Thus if a node to be replaced is of high degree, we consider that it has higher probability to be the hub node of its cluster such that the sampling process should take further consideration whether it should be selected as the replaceable node.

Algorithm 1 Proposed Select_Replaced_Node(V_s)

Input: Set of sampled nodes: V_s;
Output: Selected node to be replaced: u''.

1: $i \leftarrow discreteUniform[1, |V_s|]$
2: $u' \leftarrow V_s[i]$
3: $flag \leftarrow 1, u'' \leftarrow u', min \leftarrow |N_s(u')|$
4: **for all** $ne \in N_s(u')$ **do**
5: **if** $|N_s(ne)| > |N_s(u')|$ **then**
6: $flag \leftarrow 0$
7: *break*
8: **else**
9: **if** $|N_s(ne)| < min$ **then**
10: $min \leftarrow |N_s(ne)|$
11: $u'' \leftarrow ne$
12: **end if**
13: **end if**
14: **end for**
15: **if** $flag = 0$ **then**
16: $u'' \leftarrow u'$
17: **end if**
18: **Return:** u''

As shown in Algorithm 1, u' is the selected node uniformly at random, $flag$ is a boolean variable used to record if the selected node u' has higher degree than that of all its neighbors, and $N_s(u')$ is a set of neighbor nodes of u' in V_s. min is used to record the smallest degree of neighbour $N_s(u')$ of node u'. First, we uniformly select the node u' that could be replaced at random. Second, we need to decide whether to replace this node or its neighbour. We compare its degree with that of all its neighbors. If the node's degree is higher than all its neighbors', which indicates the node is probably still a hub node in the cluster, we should not replace the current selected node u' but one of its neighbor with the lowest degree as replaceable node instead. In this manner, hub nodes in the sample are kept on the fly, preserving the inherent clustering structure of original graph stream.

4.3 Isolated Nodes Elimination

We define sampled nodes in the sample as the node reservoir V_s, and denote *isolated nodes* by those nodes kept in V_s with no edges attached. The first reason is that in replacement phase of *PIES*, random selection of the replaceable node may cause the newly added node becomes isolated. Here is a concrete example to illustrate how such kind of isolated nodes are produced.

Example 1: Assuming the new edge $e_t = (u, v, t)$ arrives at timestamp t and is going to be inserted into E_s, when $V_s = [a_1, a_2, ...a_n, u]$ (i.e., u belongs to V_s but v does not). Since node v does not belong to V_s, *PIES* should randomly select a replaceable node for it. However, node u has a probability $p = 1/|V_s|$ to be selected to be the replaceable one so that it will be removed from V_s after v is added. Since u does not belong to V_s any more after the node replacement, e_t will not be inserted such that v becomes an isolated node.

Obviously, this situation happens from time to time that an incident node of new edge is selected as the one to be replaced. To get rid of this problem, we design a stack structure, named *ForbiddenStack* and stipulate that nodes existing in this stack should not be selected as replaceable nodes. For each new edge, its associated nodes will be added into *ForbiddenStack* such that they are forbidden to be chosen as replaceable nodes. When the next edge arrives, nodes in the forbidden stack should be popped up such that it would not influence further node replacement.

The second reason is that after the replaceable node is chosen and needs to be replaced, the neighbours that only connect to the replaceable node become isolated. It is because *PIES* removes all incident edges of the replaceable node such that the neighbours which only connect with it will have no edge attached. The solution is to check the degrees of neighbours of the replaceable node. If the degrees of neighbors are reduced to zero, those isolated nodes should be removed.

4.4 Edge-deletion Operation

Incorporating the ability to delete edges is of crucial importance in a fully-dynamic streaming setting. For example, if the sampling is performed over a sliding window, outdated edges need to be removed from the tail end of the sliding window timely. However, *PIES* does not take the edge-deletion request into consideration. Thus we propose an efficient deletion method to fully support edge-deletion requests. The pseudocode description is shown in Algorithm 2. Once an edge-deletion is required, if the corresponding edge exists in the sample, it should be directly eliminated from the sample. Otherwise it means the edge was not sampled into the reservoir before and we just ignore it. Note that the edge-deletion request deems to be always satisfied since the deletion of an edge means that the influence of the edge and its incidental nodes on the structure of graph should be eliminated as soon as possible. Moreover, we check the degree of incidental nodes of the deleted edge and remove them if their degree equals to zero after the deletion operation. It guarantees this extension would not introduce extra isolated nodes into the sample. Note that because of the edge deletion operation, the node reservoir may not meet the target size and it will be afterwards compensated again by adding new edges.

Algorithm 2 Edge-deletion(e_t, G_s)

Input:
Edge to be deleted: $e_t = (-, \langle u, v, t \rangle)$;
Sampled subgraph: $G_s = (V_s, E_s)$.
Output:
Updated sampled subgraph: G_s.

if $e_t \in E_s$ **then**
  $E_s \leftarrow E_s - \{e_t\}$
  if $|N_s(u)| = 0$ **then**
    $V_s \leftarrow V_s - \{u\}$
  end if
  if $|N_s(v)| = 0$ **then**
    $V_s \leftarrow V_s - \{v\}$
  end if
end if
Return: G_s

4.5 Clustering-structure Representative PIES

Based on these improvements above, we propose the clustering-preserving partially-induced-edge-sampling algorithm, namely *CPIES*, to handle the fully-dynamic graph streams. The algorithmic description is shown in Algorithm 3.

Compared with the basic *PIES* algorithm, the proposed *CPIES* is more likely to keep hub nodes from different clusters and avoid isolated nodes such that *CPIES* should maintain the clustering structure in the samples well.

Algorithm 3 Clustering-structure Representative PIES

Input:
Graph stream by time-stamp t: $G(t) = (V,E)$;
Sample size: n.
Output:
Sampled subgraph by time-stamp t: $G_s(t) = (V_s, E_s)$.
$V_s \leftarrow \emptyset, E_s \leftarrow \emptyset, t \leftarrow 0$
while the edge stream $(symbol, e_t)$ arrives at t **do**
  $e_t = (u,v,t)$
  ###{**Edge addition**}
  if symbol = '+' **then**
    if $|V_s| < n$ **then**
      if $u \notin V_s$ **then**
        $V_s \leftarrow V_s \cup \{u\}$
      end if
      if $v \notin V_s$ **then**
        $V_s \leftarrow V_s \cup \{v\}$
      end if
      $E_s \leftarrow E_s \cup \{e_t\}$
      $m \leftarrow |E_s|$
    else
      $p_e \leftarrow \frac{m}{t}$
      $r \leftarrow Random(0,1)$
      if $r < p_e$ **then**
        $ForbiddenStack \leftarrow ForbiddenStack \cup \{u,v\}$
        if $u \notin V_s$ **then**
          $u' \leftarrow Select\_Replaced\_Node(V_s)$
          $V_s \leftarrow V_s \cup \{u\} - u'$
          for each edge e' incident to u' in E_s **do**
            $E_s \leftarrow E_s - \{e'\}$
          end for
        end if
        if $v \notin V_s$ **then**
          $u' \leftarrow Select\_Replaced\_Node(V_s)$
          $V_s \leftarrow V_s \cup \{v\} - v'$
          for each edge e' incident to v' in E_s **do**
            $E_s \leftarrow E_s - \{e'\}$
          end for
        end if
        $ForbiddenStack \leftarrow ForbiddenStack - \{u,v\}$
      end if
      ###{**Partial edge induction**}
      $E_s \leftarrow E_s \cup \{e_t\}$
    end if
  end if
  ###{**Edge deletion**}
  if symbol = '-' **then**
    $G_s \leftarrow Edge\text{-}deletion(e_t, G_s)$
  end if
  t++
end while

5 Experimental Evaluation

This section presents a series of experiments to evaluate the qualities of various sample strategies on their abilities of preserving structural properties. First of all, we will briefly describe the graphs we used and the methodology for the evaluation. Then we discuss the obtained results. We implement streaming edge sampling (*StreamES*), streaming node sampling (*StreamNS*), *PIES*, *PIES (min)* and proposed *CPIES* using C++ language. For each sample rate p, we experiment with five different runs and calculate various metrics for designated snapshots of real-world networks.

5.1 Real-world graph streams

To validate the effectiveness of our proposed method, the real-world graphs are chosen from different domains and they are obtained from the Stanford Large Network

Dataset Collection [9]. A brief summary of these real networks is shown in Table 1.

Table 1: Summary of real-world graphs used in the experiments. We accumulate graph streams based on specific time interval in order to obtain corresponding snapshots and the statistics of the end streams are given. Abbreviations are described as follows: *# simple edges*: number of non-loop and non-duplicate edges; *# snapshots*: number of snapshots; *# comps*: number of components; *CC*: the average clustering-coefficient for all nodes in the graph.

$\|G\|$	$\|V\|$	$\|E\|$	#simple edges	Time span (days)	Interval (months)	#snapshots	Statistics of the end streams				
							#comps	diameter	radius	density	CC
Enron	151	50,572	1611	1138	6	7	2	4	3	0.14400	0.5210
Email-Eu	986	332,334	16,064	803	2	14	1	7	4	0.03308	0.4070
Col_Msg	1,899	59,835	13,835	193	1	7	4	8	4	0.00772	0.1097
Reality	6,809	52,050	7697	106	0.5	8	1	8	4	0.00033	0.0178
Slashdot	51,068	280,443	117,340	371	2	7	1	17	9	0.00009	0.0201
Fackbook	46,952	876,933	182,384	1560	4	14	842	18	9	0.00019	0.1149

5.2 Evaluation methodology & measurements

Current measures of representativeness are inadequate for our target: how well the clustering structure is represented by the counterpart of samples. Hence, we describe our methodology on how to evaluate the clustering structure quantitatively.

Methodology: Firstly, we generate the clusters of each snapshot in original graph streams using any credible clustering algorithm to serve as ground-truth, and then run the same algorithm on the samples generated by these sampling techniques. Note that in order to obtain method-independent results, we utilize two credible and scalable clustering algorithms including: *Blondel* [4] and *BigClam* [11] to generate the ground-truth clusters. Secondly, we evaluate the clustering quality of the sub-graph generated by each sampling technique using multiple metrics to validate the effectiveness of the methods.

Measurements: We now briefly describe several clustering quality metrics to assess how representative samples are in terms of the inherent clustering structure in the larger network. We consider *precision* and *recall* as two main aspects of clustering quality, and each aspect needs to be handled separately without losing the significance of both. Thus, first of all, we utilize δ-*precison* and δ-*recall* proposed in [12] to capture the differences of clustering structure between the original graph and the sampled counterpart. δ is a predefined purity threshold and which measures the correctness of the relation between clusters of original graph and those of sampled counterpart. Two clusters are considered a match if the degree of match is not less than δ. Higher value of δ-*precision* means that the obtained clusters of G_s are more precisely representative of the ground-truth clusters of G while higher value of δ-*recall* indicates the ground-truth clusters of G are more successfully covered by the obtained clusters of G_s.

Secondly, we also employ several representative metrics widely used for evaluating clusters in the graph including: *adjusted Rand index (ARI)* [5], *normalized*

mutual information (NMI) [5] and *accuracy for number of clusters (ANC)* [11] to evaluate the clustering results of the sampled graphs. Note that those metrics are designed solely to assess the clustering quality on the entire graph, not particularly on the sample. To make a fair comparison, we utilize them on the subgraph $G(V_s)$ (i.e., the same set of nodes in the sample).

5.3 Experimental Results

Overall performance: In the first experiment, we sample 20% of the total number of nodes that appear in the stream as it is progressing. In order to evaluate the sample quality in a streaming setting, we take snapshots at different time points in graph streams based on their lifetime (i.e., the number of edges ordered by time-stamps) and compute multiple quality metrics for each snapshot, and then we calculate the "average" scores over all the snapshots to assess the overall performances. We report the performances of these sampling methods for each network in Table 2. Here we only present the results using *Blondel* clustering, and other results using *BigClam* algorithms exhibit similar behaviors. Some remarkable conclusions can be drawn as follows:

Table 2: The qualities of various sampling strategies on real graphs ($p = 20\%$). Besides, all the metrics are calculated by using *Blondel* clustering algorithms on both original graph and the sampled counterpart. Bold values indicate the best results for corresponding metrics. Please note that the maximum value of δ-*recall* is equal to the sample rate p.

(a) Enron	Metrics	CPIES	PIES	StreamNS	StreamES	PIES (Min)
Measures	1.0-precision	0.683	0.662	0.547	**0.704**	0.684
	1.0-recall	0.114	0.110	0.085	0.103	**0.118**
	0.5-precision	**0.868**	0.840	0.807	0.764	0.862
	0.5-recall	**0.155**	0.144	0.115	0.106	0.150
	0.0-precision	**0.883**	0.852	0.832	0.764	0.879
	0.0-recall	**0.164**	0.151	0.139	0.106	0.154
	ANC	**0.861**	0.857	0.821	0.548	0.825
	NMI	**0.784**	0.766	0.751	0.736	0.770
	ARS	**0.646**	0.606	0.561	0.343	0.595

(b) Email-Eu	Metrics	CPIES	PIES	StreamNS	StreamES	PIES (Min)
Measures	1.0-precision	0.242	0.347	0.326	**0.589**	0.383
	1.0-recall	**0.041**	0.025	0.021	0.028	0.027
	0.5-precision	0.729	**0.736**	0.716	0.690	0.729
	0.5-recall	**0.117**	0.098	0.098	0.031	0.099
	0.0-precision	**0.785**	0.774	0.770	0.708	0.773
	0.0-recall	**0.131**	0.116	0.117	0.031	0.111
	ANC	**0.845**	0.549	0.543	0.106	0.540
	NMI	**0.611**	0.546	0.555	0.504	0.545
	ARS	**0.439**	0.364	0.362	0.043	0.365

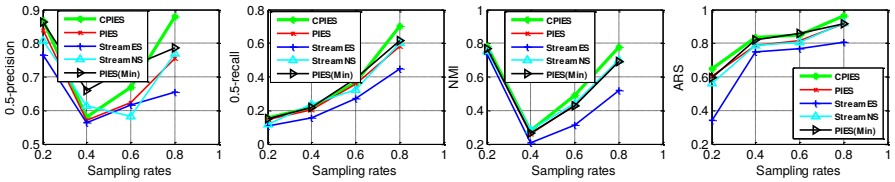
(c) Col_Msg	Metrics	CPIES	PIES	StreamNS	StreamES	PIES (Min)
Measures	1.0-precision	**0.397**	0.149	0.120	0.307	0.192
	1.0-recall	**0.054**	0.002	0.002	0.012	0.011
	0.5-precision	0.455	0.222	0.210	**0.478**	0.281
	0.5-recall	**0.060**	0.008	0.009	0.018	0.019
	0.0-precision	**0.571**	0.417	0.452	0.515	0.480
	0.0-recall	**0.093**	0.046	0.048	0.023	0.054
	ANC	0.437	0.789	**0.913**	0.128	0.809
	NMI	0.235	0.172	0.205	**0.391**	0.171
	ARS	0.043	0.043	0.055	0.017	**0.049**

(d) facebook	Metrics	CPIES	PIES	StreamNS	StreamES	PIES (Min)
Measures	1.0-precision	**0.961**	0.940	0.955	0.903	0.941
	1.0-recall	**0.170**	0.158	0.169	0.145	0.158
	0.5-precision	**0.975**	0.958	0.973	0.935	0.959
	0.5-recall	**0.172**	0.159	0.171	0.146	0.160
	0.0-precision	**0.980**	0.963	0.978	0.940	0.963
	0.0-recall	**0.172**	0.160	0.172	0.146	0.161
	ANC	**0.951**	0.928	0.856	0.795	0.931
	NMI	**0.781**	0.759	0.779	0.742	0.763
	ARS	**0.699**	0.666	0.686	0.597	0.669

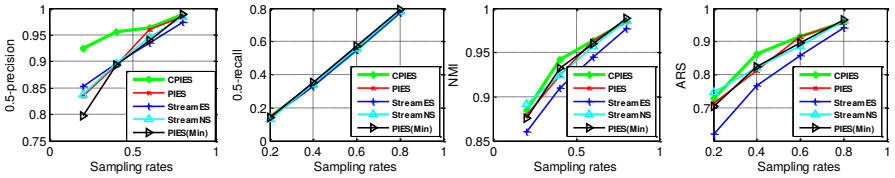
(e) Slashdot	Metrics	CPIES	PIES	StreamNS	StreamES	PIES (Min)
Measures	1.0-precision	**0.877**	0.867	0.851	0.834	0.811
	1.0-recall	**0.022**	0.021	0.021	0.020	0.020
	0.5-precision	**0.892**	0.882	0.866	0.875	0.842
	0.5-recall	0.027	0.027	0.028	0.024	**0.030**
	0.0-precision	**0.911**	0.903	0.895	0.888	0.873
	0.0-recall	0.058	0.059	0.062	0.041	**0.066**
	ANC	0.205	0.220	0.244	0.140	**0.322**
	NMI	0.381	0.362	0.419	**0.419**	0.345
	ARS	**0.158**	0.147	0.154	0.120	0.157

(f) Reality	Metrics	CPIES	PIES	StreamNS	StreamES	PIES (Min)
Measures	1.0-precision	0.514	0.498	**0.549**	0.485	0.471
	1.0-recall	**0.083**	0.077	0.073	0.075	0.080
	0.5-precision	**0.925**	0.835	0.832	0.852	0.797
	0.5-recall	**0.147**	0.144	0.134	0.140	0.140
	0.0-precision	0.861	0.868	**0.892**	0.880	0.842
	0.0-recall	**0.184**	0.180	0.140	0.164	0.184
	ANC	**0.886**	0.897	0.652	0.869	0.792
	NMI	0.885	0.879	**0.892**	0.860	0.876
	ARS	0.728	0.714	**0.748**	0.620	0.705

- The total charts of these results are conclusive. *CPIES* algorithm outperforms other algorithms in most metrics. This is because that *CPIES* is biased towards



(a) *Enron* network



(b) *Reality* network

Fig. 1: The impact of varying sample rate p on quality metrics on *Enron* and *Reality* networks using *Blondel* clustering.

the central nodes with high degree, and they are good representatives of the underlying clustering structure. For *PIES* algorithm, the replaced nodes may be the hub nodes of clusters. It will destroy the topology structure and make the clusters in the sample loss of hub nodes and critical connection. Since *StreamES* and *StreamNS* sampling uniformly sample the edges/nodes without considering the inherent structure of network, their performances highly depend on the structures of graphs.

- Besides, we found that sampling algorithms that include an induced graph step (i.e., *CPIES*, *PIES* and *StreamNS*) in their process perform better than *StreamES* (under-sample of edges) in most case because they contain more edges incident to the sampled nodes.

Note that the number of isolated nodes is also analyzed for both *CPIES* and *PIES*. Because *PIES* replace nodes at random, it is inevitable that its sampled counterpart contains some nodes with zero degree (i.e., isolated nodes). Each time a new edge is sampled from the stream, its incident nodes replace randomly selected nodes from the reservoir. This random replacement policy could replace high-degree nodes while isolated nodes remain in the reservoir. However, the proposed *CPIES* completely avoid isolated nodes and achieves better sample quality.

The impact of sampling rates: In the second experiment, we analyze the impact of the sample rate p. The sample rate p controls the ratio of the number of nodes between the original graph and the sampled counterpart. The sample rate p increases gradually from 0.20 to 0.80 with the interval of 0.20. We run each sampling algorithm 3 trials and take the average of each metrics.

Fig. 1 shows the the average results of various metrics on *Enron* and *Reality* networks using *Blondel* clustering, and we can observe that *CPIES* performs consistently well with different sampling rates. Note that *StreamES* does not include an induced graph step so it underestimates the number of edges of the samples such that it fails to preserving the clustering structure well.

6 Conclusion

In this work, we present a new clustering-structure representative sampling method to produce samples from fully-dynamic graph streams. It is capable to retain the influential nodes of clusters and discard isolated nodes such that the clustering structure of original graph is preserved. We empirically demonstrate that *CPIES* can represent inherent clustering structure of graph streams in an online fashion, and it outperforms current online sampling algorithms in most properties, especially in terms of clustering performance.

In future work, one interesting direction is to explore more general ways to sample the fully-dynamic graph while preserving the clustering structure. Another direction is to design credible measurements to quantitatively evaluate the quality of sample processes. Such quality measures will guide our understanding and study of improved sampling methods.

Acknowledgements This research of Zhang is supported by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 61521003) and the National key Research and Development Program of China (Grant No. 2016YFB0800101). The research of Pei is supported by the Netherlands Organisation for Scientific Research (NWO).

References

1. Ahmed, N.K., Duffield, N., Neville, J., Kompella, R.: Graph sample and hold: A framework for big-graph analytics. In: KDD, pp. 1446–1455. ACM (2014)
2. Ahmed, N.K., Duffield, N., Willke, T., Rossi, R.A.: On sampling from massive graph streams. VLDB (2017)
3. Ahmed, N.K., Neville, J., Kompella, R.: Network sampling: From static to streaming graphs. TKDD **8**(2), 7 (2014)
4. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **10**, 008 (2008)
5. Fortunato, S., Hric, D.: Community detection in networks: A user guide. Physics Reports **659**, 1–44 (2016)
6. Hübler, C., Kriegel, H.P., Borgwardt, K., Ghahramani, Z.: Metropolis algorithms for representative subgraph sampling. In: ICDM, pp. 283–292. IEEE (2008)
7. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: KDD, pp. 631–636. ACM (2006)
8. Maiya, A.S., Berger-Wolf, T.Y.: Sampling community structure. In: WWW, pp. 701–710. ACM (2010)
9. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: WSDM, pp. 601–610. ACM (2017)
10. Stefani, L.D., Epasto, A., Riondato, M., Upfal, E.: Trièst: Counting local and global triangles in fully dynamic streams with fixed memory size. TKDD **11**(4), 43 (2017)
11. Yang, J., Leskovec, J.: Community-affiliation graph model for overlapping network community detection. In: ICDM, pp. 1170–1175. IEEE (2012)
12. Zhang, J., Pei, Y., H. L. Fletcher, G., Pechenizkiy, M.: Structural measures of clustering quality on graph samples. In: ASONAM, pp. 345–348. IEEE (2016)