SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2023, Vol. 66 112103:1–112103:17
https://doi.org/10.1007/s11432-021-3370-4

Cluster-preserving sampling algorithm for large-scale graphs

Jianpeng ZHANG[1*], Hongchang CHEN[1], Dingjiu YU[2,1], Yulong PEI[3] & Yingjun DENG[4]

[1]*National Digital Switching System E&T Research Center, Information Engineering University, Zhengzhou* 450001*, China;*
[2]*Network Systems Department of the Strategic Support Force, Beijing* 100091*, China;*
[3]*School of Computer Science and Technology, Eindhoven University of Technology, Eindhoven* 5612AE*, the Netherlands;*
[4]*Center for Applied Mathematics, Tianjin University, Tianjin* 300072*, China*

Received 5 February 2021/Revised 8 June 2021/Accepted 30 August 2021/Published online 9 November 2022

Abstract Graph sampling is a very effective method to deal with scalability issues when analyzing large-scale graphs. Lots of sampling algorithms have been proposed, and sampling qualities have been quantified using explicit properties (e.g., degree distribution) of the sample. However, the existing sampling techniques are inadequate for the current sampling task: sampling the clustering structure, which is a crucial property of the current networks. In this paper, using different expansion strategies, two novel top-leader sampling methods (i.e., TLS-e and TLS-i) are proposed to obtain representative samples, and they are capable of effectively preserving the clustering structure. The rationale behind them is to select top-leader nodes of most clusters into the sample and then heuristically incorporate peripheral nodes into the sample using specific expansion strategies. Extensive experiments are conducted to investigate how well sampling techniques preserve the clustering structure of graphs. Our empirical results show that the proposed sampling algorithms can preserve the population's clustering structure well and provide feasible solutions to sample the clustering structure from large-scale graphs.

Keywords graph sampling, clustering structure, top-leader nodes, expansion strategies, large-scale graphs

Citation Zhang J P, Chen H C, Yu D J, et al. Cluster-preserving sampling algorithm for large-scale graphs. Sci China Inf Sci, 2023, 66(1): 112103, https://doi.org/10.1007/s11432-021-3370-4

1 Introduction

A graph is a generic data structure that can characterize complex relationships between the entities in the networks. For instance, in online social networks, users can be represented by nodes, and communication associations or friendships can be represented by links. Nowadays, in various applications, "big" graphs are becoming ubiquitous and prevalent. For example, in the Facebook network, daily active users have increased to 1.84 billion, and the total number of monthly active users has reached close to 2.70 billion users (as of December 2020). Analyzing such large-scale graphs has become of great significance in various kinds of applications (e.g., abnormal connections in social media). To understand the structures of these real-world graphs in various scenarios, effective and efficient graph mining algorithms (e.g., sampling, clustering, and classification, among other algorithms) are in great demand. However, there has been a dramatic increase in "big" graphs, which poses huge challenges to researchers in the fields of graph mining. Moreover, the corresponding techniques typically exhibit high computational complexity. Therefore, effectively processing these big graphs is becoming a crucial issue for data scientists. One feasible solution to these problems is to sample a representative subgraph and explore the inherent structures or characteristics.

We consider the sampled graph as a representative if the sampled graph preserves the selected properties of the original graph. Here, the simple node-level properties (e.g., distributions of degree, betweenness centrality, and closeness centrality are commonly investigated by researchers, but sampling influences on

* Corresponding author (email: zjp@ndsc.com.cn)

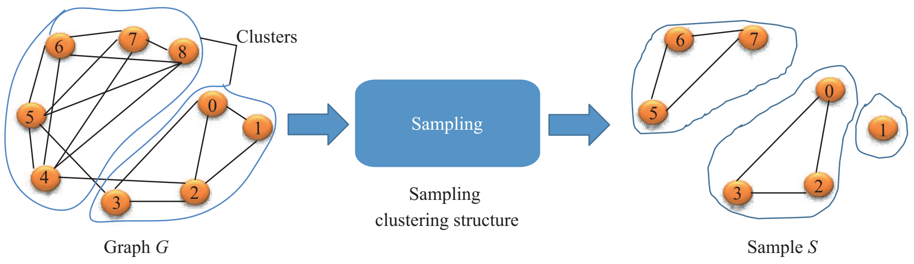


Figure 1 (Color online) A tiny example of sampling the clustering structure.

the intrinsic clustering structure have been overlooked, which is a prevailing characteristic of real-world graphs. As shown in Figure 1, the objective of the sampling is to take a subgraph of the original graph in such a way that the sampled subgraph is representative of the original graph by considering the clustering structure. The basic principle is that if the sample truly represents the original graph in terms of the clustering structure, the clustering structure on the sampled should be well extended to the unsampled nodes. Therefore, it provides a feasible solution in large-scale graph analysis and becomes an important procedure for inferring potential properties in the original graph.

Consequently, there are two research challenges we should address: (i) The first challenge is to construct an effective sampling method on large-scale graphs and ensure that the sample represents the perspective of clustering structures. A sample is representative if it has the original graph's selected properties. Furthermore, a sample is representative if it preserves the selected properties of the original graph. Existing sampling methods, however, are inadequate for keeping the inherent clustering structure. (ii) The second challenge is to evaluate the representativeness of the clustering structure of the sample. There are some studies on how to evaluate the sample quality, but their results are far from solving the problem. To meet these challenges, our main contributions are as follows.

• We devise two novel top-leader sampling algorithms, which are subject to equal-sized expansion (TLS-e) and internal-priority expansion (TLS-i), respectively. These two algorithms can yield representative samples and preserve the clustering structure effectively. The rationale is to take top-leader nodes in each cluster to the sample and then heuristically incorporate the peripheral nodes satisfying the expansion strategies into the subgraph.

• We design the methodology to measure how the sampled graph's clusters with the ground-truth of the original graph can quantitatively differ and utilize effective metrics to evaluate the sampling quality from diverse perspectives.

• Experiments are conducted to investigate how sampling influences the clustering structure of graphs, and empirical results have shown that proposed sampling algorithms (i.e., TLS-i and TLS-e) can maintain the intrinsic clustering structure when sampling from large-scale graphs. Furthermore, we found that TLS-i is more suitable for sampling from graphs in which the distribution of cluster sizes is diverse, and TLS-e is the opposite.

2 Related work

Graph sampling has been widely studied, and it is of great interest to researchers in a wide range of fields. Although several types of graphs may be studied in each field, the key point has been on devising sampling methods to sample representative subgraphs from larger graphs and using subgraphs to simplify downstream tasks (e.g., classification [1] and clustering [2]). Therefore, the related work consists of two parts: (a) sampling methods; (b) sampling the clustering structure.

2.1 Sampling methods

To preserve specific graph properties for domain-specific objects, lots of sampling approaches have been adopted. In [1, 3], these sample methods can approximately be classified into three types: node-based, edge-based, and traversal-based sampling.

2.1.1 *Node-based sampling*

The widely-used induced random vertex (IRV) sampling [4] chooses nodes from the original graph to the desired size with equal probability, and then it includes existing edges among the selected nodes from the population into the sampled counterpart. The strategy is uncomplicated in approximating the direct properties of nodes, e.g., degree distribution, but the connectivity is unlikely to be retained. Some advanced node-based sampling methods are proposed [1], for example, random page-rank node (RPN) and random degree node (RDN) choose sampled nodes according to page-rank weights and node degrees, respectively.

2.1.2 *Edge-based sampling*

Random edge (RE) sampling [3] generates an induced subgraph where each edge is chosen independently and uniformly at random. The sample is a partially induced subgraph, where no edges are added on existing nodes in the sample. Thus, it makes the sampled graph very sparsely connected. Another variant named induced random edge (IRE) first chooses numerous edges from the original graph and adds incident nodes. Then, it incorporates edges from the original graph among the chosen nodes if edges have not been sampled. Both samplings can produce disconnected samples and do not preserve the clustering structure well.

2.1.3 *Traversal-based sampling*

Traversal-based sampling can be divided into two subcategories: sampling without replacement and with replacement. Depth first (DF) and breadth first (BF) sampling [4] are two basic sampling without replacement methods. Both start from a randomly selected node and then select nodes in depth- and breadth-first order, respectively. In both samplings, once the starting node is randomly chosen, the rest of the procedure is deterministic, and it makes the sample very sensitive to the initial seed node. Snow ball (SB) sampling [3] selects a fixed number of neighbors to visit in each loop, but it becomes worse from boundary bias such that many sampled nodes in the final loop lose their neighbors. Forest fire (FF) sampling is a recursive process where a seed node is picked and starts to burn outgoing edges and nodes at the other end get "burned". In each iteration, the picked node selects x of its outgoing neighbors, where x is geometrically distributed with the mean $\bar{x} = p_f/(1-p_f)$, and p_f is the forward burning probability. The iteration continues until the target size is reached.

For sampling with replacement, random walk (RW) sampling [1] is a well-known method in social network analysis. It begins from a randomly chosen node and picks the neighboring node by simulating a random walk on edges. However, it does not work well in disconnected or loosely connected graphs because the location of the starting node's components cannot be sampled. To solve this problem, random jump (RJ) [3] sampling was proposed, and it gives some probability that the node can randomly jump to any node in the graph, such that it does not get stuck into a local optimum. Also, Metropolized random walk (MRW) [1, 3] sampling is a variant of the Markov chain Monte-Carlo algorithm, and its aim is to draw samples from the target uniform distribution $p(u)$, where $p(u) = 1/N$, N is a normalizing constant, we do not have a priori knowledge. Thus, MRW sampling removes the bias of the RW sampling algorithm so that the target distribution is uniform.

2.2 Sampling the clustering structure

The purpose of this study is distinct from those in the literature. Unlike conventional sampling methods, which concentrate on sampling nodes' properties of a graph, we focus on sampling a graph to preserve non-trivial structural properties (e.g., topological or organizational properties). Though a considerable amount of work on sampling that exists in the complex network and social sciences, most sampling techniques are not specially designed to preserve the essential structural property, i.e., clustering structure [2, 5]. However, the clustering structure of a graph is of importance since they are often associated with common/latent characteristics (e.g., affiliation, role, and common interest).

Leskovec et al. [3] investigated a series of sampling algorithms to sample from large-scale graphs and evaluate various graph properties of interest. They reported that RW and FF samplings perform better than competing sampling algorithms on basic properties. However, experimental measurements only consider the basic properties (e.g., in-degree, out-degree, connected components, and hop-plot distribution), which is insufficient to guarantee that the sample is representative regarding the inherent clustering structure. The work in [6] only assessed which samples are representative of explicit or simple graph properties (e.g., the degree distribution), and it cannot reflect the topology structure as much as possible. Ref. [7] put forward two sampling algorithms based on the idea that the sample with good expansion property tends to be more representative of the clustering structure than others. Ref. [8] proposed a graph sampling based on graph Fourier transform (FGFT). By showing the equivalence of a shifted A-optimal criterion and a function of an ideal low-pass (LP) graph filter, a new sampling method is proposed to minimize the new objective to select sampled nodes greedily. Furthermore, Ref. [9] proposed the SInetL method to deal with the Internet topology specifically. They resolved the Internet topology by employing two normalized Laplacian spectral features and fulfilled a marked reduction in the size of the topology using sampling and merging of these graphs. The number of nodes can be reduced using the proposed method in these graphs effectively while preserving their essential properties. Ref. [10] propounded a sampling approach that is oriented to contextual structures' preservation and employs a graph representation model to extract the contextual structures. A general graph sampling framework called GraphSDH is proposed by [11] using the vertex-centric graph model. They derived a stratified sampling method that was based on vertex-degree and a hierarchical optimization scheme based on sampling position analysis. It substantially accelerates the PageRank computation.

The focus of this work is to design sampling algorithms that represent the inherent clustering structure in the original graph. Generally, there are three main aspects differentiating our work from previous studies on graph sampling: (i) Traditional sampling methods focus on acquiring unbiased estimates related to explicitly measurable node properties, such as degree distributions and clustering coefficients. By contrast, we concentrate on the use of sampling to preserve non-trivial structural properties (i.e., clustering structure) of the graph itself. Such properties are usually considered difficult to identify or measure even with full access to the entire graph. (ii) Unlike most existing research, we analyze the advantages of certain sampling biases and leverage these biases to preserve structural properties. (iii) We exploit the sampling technique as an essential procedure for mining tasks on the sampled graph.

3 Definitions & problem statement

3.1 Basic notations and definitions

Traditional graph analysis considers the graph-structured data as a static graph1), which is either derived from edges' aggregation across all the time or acquired as a snapshot of edges at a special timestamp. The formal definition is as follows.

Definition 1 (Static graph)**.** A static graph G is defined as $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times V \times \mathbb{R}^+$ is a set of weighted edges, such that $N = |V|$ is the number of nodes, and $M = |E|$ is the number of weighted edges. $\mathbb{R}^+$ denotes the set of positive real numbers. Each weighted edge in E is represented as $\langle u, v, w\rangle$, where u and v are the two incident nodes of the edge, where $u \neq v$ and w are edge weights.

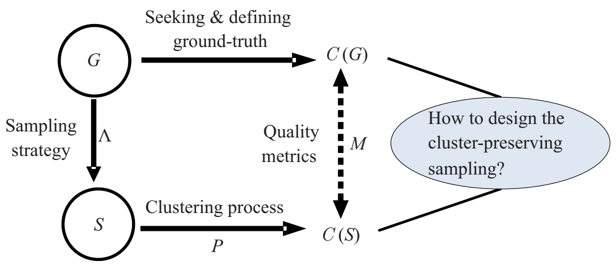
In several real-world graphs, edges in a graph may not have the same weights. For instance, edges usually are related to weights, which differentiate them in terms of quantity, intensity, and strength. Thus, if edges have different weights, we treat the graph as a weighted static graph. Otherwise, it is an unweighted static graph, which is a special case of a weighted graph (i.e., $w = 1.0$). Furthermore, with the pervasive use of social networks, there is a dramatic increase of large-scale static graphs, which is followed by a surge of interest in analyzing such graphs across various disciplines. Therefore, it is necessary to analyze such graphs, which would provide important insights into their structures and formation mechanisms.

Definition 2 (Sample)**.** A sample $S = (V_s, E_s)$ is a sampled subgraph of an original graph $G = (V, E)$, where $E_s \subseteq E$ and $V_s \subseteq V$ such that each node that is noted in E_s is in V_s.

Definition 3 (Sampling rate)**.** A sampling rate p is a sample fraction of nodes V_s from V, i.e., $p = |V_s|/|V|$.

Definition 4 (Clustering)**.** A clustering $\mathcal{C}$ is a finite non-empty node-set $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, where each C_i $(i \in [1, k])$ is a subset of V, called a cluster of G, and it holds that $\bigcup_{C\in\mathcal{C}} C = V$. Note that clusters can be either disjoint or overlapped with each other.

1) In this paper, the network and graph can be used interchangeably, and for ease of analysis, undirected graphs are considered only.



Seeking & defining ground-truth
G
C (G)
Sampling strategy
Λ
Quality metrics
M
How to design the cluster-preserving sampling?
S
Clustering process
P
C (S)

Figure 2 Problem setting. Assume that S is a sample of a graph G, $\mathcal{C}(G)$ is a valid ground-truth of G, and $\mathcal{C}(S)$ is a clustering of S induced by process P, the goal is to preserve the clustering structure $\mathcal{C}(S)$ of sample S well with respect to $\mathcal{C}(G)$ of G.

Definition 5 (Top-leader). For a graph G, a top-leader l_i of a cluster $C_i(G)$ is the representative node of the i-th cluster of $\mathcal{C}(G)$, where i ($i \in [1, k]$) denotes the index of i-th cluster.

Definition 6 (Top-leader list). A top-leader list $L = \{l_1, \ldots, l_k\}$ is the list of top-leaders of the k clusters, where l_i ($i \in [1, k]$) represents the top-leader of the i-th cluster, whereas k is the number of clusters in $\mathcal{C}(G)$.

Definition 7 (Neighbors). Neighbors $N(S)$ of S denote neighbors of the sample S, i.e., $N(S) = \{w \in V - V_s \mid \exists v \in S \text{ s.t. } (v, w) \in E\}$. Specially, when $S = \{u\}$, $N(S)$ can be written as $N(u)$ and $|N(u)|$, which refers to the degree of node u.

Definition 8 (Important neighbors). Important neighbors $H(u)$ of u denote the neighbors with equal or higher degree than node u. That is, $H(u) = \{v \mid v \in N(u) \text{ s.t. } |N(v)| \geqslant |N(u)|\}$.

Definition 9 (Expansion factor). The expansion factor $X(S)$ of S is defined as $X(S) = \frac{|N(S)|}{|S|}$, where $N(S)$ is the neighbors of S, and $|S|$ is the cardinality of S.

Definition 10 (Extra neighbors). Extra neighbors $\mathrm{EN}(v, S)$ are the set of new neighbors contributed by node v once it is sampled into S. That is, $\mathrm{EN}(v, S) = N(v) - N(S) \cup S$.

Definition 11 (Internal nodes). Internal nodes $\mathrm{IN}(S)$ are denoted by the neighbors of S, which do not contribute to any new neighbor. That is, $\mathrm{IN}(S) = \{v \mid v \in N(S) \text{ s.t. } \mid \mathrm{EN}(v, S)| = 0\}$.

Definition 12 (Activated node). A node v is an activated node if it has not been sampled. We call v an activated node and say that node v is activated.

Definition 13 (Deactivated node). A node v is a deactivated node if it has been sampled.

3.2 Problem setting

The primary objective is to design a new sampling algorithm to sample large-scale graphs that preserve the clustering structure. How can we preserve the clustering structure using the sample S from the original graph G? First, we want the sampled subgraph to contain the top-leader $l_i \in C_i$ from most (or even all) clusters of the original graph, and also nodes that are tight around the top-leader nodes should be retained. In this way, it makes the sample preserve the clustering $\mathcal{C}$ of the original graph. Second, the nodes that are grouped together in the sampled counterpart should stem from nodes located in the same cluster of the original graph. As shown in Figure 2, the problem setting can be formally defined as follows.

Definition 14 (Cluster-preserving sample). Given a graph G, a clustering method $\mathcal{P}$, a similarity measurement $\mathcal{M}[\cdot, \cdot]$, a sample S is a cluster-preserving sample that meets the following criteria:

• S maximizes the similarity, i.e., $\max \mathcal{M}(\mathcal{C}(S)), \mathcal{C}(G)$, between the clusters $\mathcal{C}(G)$ of G and clusters $\mathcal{C}(S)$ of S. Note that the clusters of the original and sampled ones are obtainable from the known ground-truth if known. Otherwise, the clusters in G and S are obtained by implementing the same clustering algorithm $\mathcal{P}$;

• The number of non-empty intersections between V_s and each cluster in $\mathcal{C}(G)$ is maximized, where $\mathcal{C}(G)$ is the set of clusters on G.

4 Proposed sampling method

Based on the definition of the cluster-preserving sample, two new top-leader sampling (i.e., TLS-e and TLS-i) methods are presented, which are capable of producing representative subgraphs and retaining clustering structure commendably. Notably, both proposed algorithms are mainly composed of two phases:

• In the first phase, it initializes k nodes, each of which plays a representative role in its corresponding cluster accordingly. That is, initializing of the k top leaders.

• In the second phase, it iteratively expands each cluster by adding eligible peripheral nodes using the specific expansion criterion that needs to be satisfied until the sample reaches the desired size.

By including these top leaders and their surrounding nodes, we expect to yield a representative sample: (1) to accommodate most or all the clusters in the graph, and (2) to be a condensed representation of the clustering structure of the original graph. More specifically, we introduce a new local top-leader initialization method, and then two expansion strategies, which are respectively subjected to TLS-e and TLS-i, are given in detail.

4.1 Initialization of top-leaders

Since the initialization of top-leaders plays an important role in sampling the clustering structure, we discuss existing methods used in the selection of initial top-leaders and present a new local top-leader initialization method.

4.1.1 *Existing global top-leaders*

The global top-leader initialization method [12] only selects top-k nodes with the highest degrees to be top-leaders of clusters in the initialization step. The reason is that top-leaders are usually located at the central part of their clusters. That is, peripheral nodes have a higher probability of connecting with top-leaders than peripheral nodes in the same cluster. However, when it is employed in real-world graphs where degrees of nodes are remarkably diverse and follow the power-law distribution [13], the selected top-leaders are global representative nodes instead of local representative nodes. In fact, the degrees of top-leaders in small clusters are probably less than some ordinary nodes in the big one, which shows that top-leaders of small clusters cannot be selected using this global method. Thus, we need an alternative initialization method to solve this problem.

4.1.2 *Local top-leaders*

Mall et al. [12] have proved that the top-leader selection is NP-hard. Therefore, it motivates us to develop a heuristic approximation approach for this task. Meanwhile, Ref. [14] proposed a top-leader selection method. The method selects high degree nodes, which are not directly connected to top-leaders. However, it is too rigorous since many top-leaders of clusters in real-world graphs should have a "connection" with each other, though some of them may not be strongly connected. For instance, in the US presidential election 2016, Donald Trump and Hillary Clinton are obviously leaders for the Democratic and Republican Parties, respectively. If we consider the relationships between members in two parties, it is unrealistic that party leaders do not connect with each other. For this reason, we put forward a more lenient local top-leaders initialization method. We assume that a top-leader should not have more than $k-1$ neighbors whose degrees are higher than its own degree (k is the number of clusters). That means it may connect other top-leaders of larger clusters (namely, the neighborhood constraint). If the number of neighbors with higher degrees for top-leaders is more than $k-1$, it means this node cannot be a top-leaders of all k clusters, and there are k nodes with higher degrees, which can be classified into the top-leader list. Furthermore, top-leaders should share few common neighbors compared with others (namely, common neighbor constraint). We define the number of common neighbors as ϵ ($\epsilon = 3$ is a good rule of thumb from the perspective of the experiment). Therefore, these nodes that meet the following criteria are given higher priority to be selected as top-leaders of clusters, i.e., each top-leader should satisfy the three constraints simultaneously.

(i) Each top-leader should have as high degrees as possible (i.e., degree constraint).

(ii) The number of neighbors with higher degrees for top-leaders should be no more than $k-1$ (i.e., neighborhood constraint).

(iii) Top-leaders should share few common neighbors with each other (i.e., common neighbor constraint).

As shown in Algorithm 1, the initialization method is simple yet effective. Firstly, we sort the node-set V in a descending order using their degree centrality. Secondly, for each node from the sorted queue

Sorted_V, the method picks the first top-leader with the highest degree, and then we pop up the next node to check whether it satisfies the constraints (i.e., neighborhood and common neighbor constraint). If satisfied, we add the node to the top-leader list L until k top-leaders are found. This process executes iteratively until the target number k is reached. This method allows for possible edge connections among top-leaders, and it can find local top-leaders more accurately. Additionally, we tune on selecting top-leaders into a larger cluster number $c \times k$, where c $(c > 1)$ is a constant, and we observe similar behavior in terms of sample quality.

Algorithm 1 Initialization of k top-leaders

```
Input: Nodes set $V$ in $G$; number of clusters $k$; common neighbor constraint $\epsilon$; desired sample size $n$.
Output: Set of top-leaders $L = \{l_1, l_2, \ldots, l_k\}$.
 1: Sorted_$V$ ← sort the nodes in $V$ by the degree from largest to smallest;
 2: $L \leftarrow \emptyset$;
 3: while $|L| < k$ do
 4:     Pop up a node $v$ from Sorted_$V$ with maximized $|N(v)|$ in $G$;
 5:     if $L = \emptyset$ then
 6:         $L \leftarrow L \cup \{v\}$;
 7:     else
 8:         if $\forall l \in L$ s.t. $|N(l) \cap N(v)| \leqslant \epsilon$ and $|H(v)| \leqslant k - 1$ then
 9:             $L \leftarrow L \cup \{v\}$;
10:         end if
11:     end if
12:     Sorted_$V$ ← Sorted_$V - \{v\}$;
13: end while
14: Return: $L$.
```

4.2 Sampling expansion strategies

After the initialization of k top-leaders, we decide which nodes in neighbor nodes $N(S)$ of S can be preferentially chosen in each loop of the sampling process. The selection directly impacts the properties of the sample being formed. Hence, we propose two expansion strategies to expand peripheral nodes of each cluster into the sample.

4.2.1 *Equal-size expansion*

Salehi et al. [15] derived that when comparing simple node-level properties, e.g., average degree, in a sampled counterpart with the original graph, an equal number of nodes from each cluster is required in order to obtain the optimal result. However, we need to verify this claim on the inherent implicit property: the clustering structure. Thus, the first strategy is to choose an equal number of nodes from each cluster in the original graph. That is, we sample the same number of nodes from each cluster, such that each cluster does not terminate its sampling expansion until its size reaches $B = \lceil \frac{n}{k} \rceil$. The pseudocode for this strategy is presented in Algorithm 2. Initially, we add top-leaders l_i into the i-th cluster C_i $(i \in [1, k])$. Next, we select the neighbor v, which is an activated node, of the cluster C_i such that the $|\mathrm{EN}(v, C_i)|$ (i.e., the number of new neighbors provided by v is minimized once it is sampled into S). This expansion condition ensures that the newly sampled node v is the peripheral node that connects tightly to its top-leaders and sparsely links to other clusters, and then we set the node v to deactivated node. If the cluster size reaches B, we start a new cluster expansion. Note that if the cluster does not have any activated node to be sampled, it becomes an isolated component, and we need to stop expanding this cluster, causing its cluster size to be less than B. Thus, we need to determine whether the target sample size is satisfied. If not, we iteratively add unsampled nodes into V_s in descending order of degree until $|V_s| = n$ is satisfied. Finally, we obtain all the sampled nodes V_s, and then select all the edges E_s that are incident to these sampled nodes from the edge set E.

4.2.2 *Internal-priority expansion*

One important factor that impacts the performance of equal-size expansion is the homogeneity of cluster sizes. In real-world graphs, however, the distribution of cluster sizes is diverse and approximately follows a power law distribution [13]. Therefore, it does not guarantee whether that assumption of an equal number of samples from each cluster is suitable for preserving the clustering structure. Thus, a defect of Algorithm 2 is that the requirement of equal cluster size is too rigorous and unrealistic in real-world

Algorithm 2 Top-leader sampling algorithm with the TLS-e

```
Input: Graph $G = (V, E)$; desired sample size $n$; number of clusters $k$; set of initial top-leaders $L = \{l_1, l_2, \ldots, l_k\}$.
Output: Sampled subgraph $S = (V_s, E_s)$; clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.
 1: $\mathcal{C} \leftarrow \emptyset$, activated nodes $\leftarrow V - L$;
 2: for all $l_i \in L$ do
 3:     $C_i \leftarrow C_i \cup \{l_i\}$;
 4: end for
 5: for all $C_i \in \mathcal{C}$ do
 6:     while $|C_i| < \lceil \frac{n}{k} \rceil$ do
 7:         if no more activated node is found then
 8:             ###{An isolated component};
 9:             Continue;
10:         else
11:             $v \leftarrow$ activatednodes s.t. $\min_{v \in N(C_i)} |\mathrm{EN}(v, C_i)|$;
12:             $C_i \leftarrow C_i \cup \{v\}$;
13:             Set $v$ to deactivated node;
14:         end if
15:     end while
16: end for
17: $V_s \leftarrow C_1 \cup C_2 \cup \cdots \cup C_k$;
18: if $|V_s| < n$ then
19:     Iteratively add nodes to $V_s$ in descending order of degree until $|V_s| = n$ satisfied;
20: end if
21: for all $e = (u, v) \in E$ s.t. $\{u, v\} \subset V_s$ do
22:     $E_s \leftarrow E_s \cup \{e\}$;
23: end for
24: Return: $S$.
```

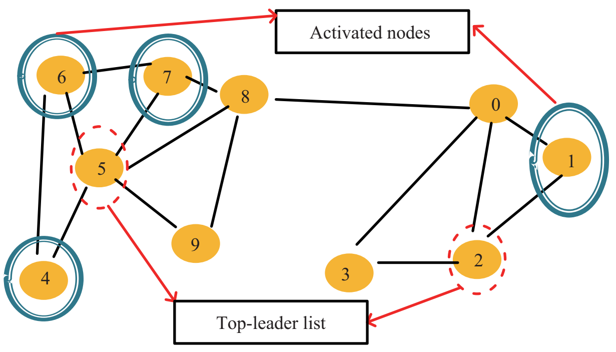


Figure 3 (Color online) Schematic diagram of related definitions.

graphs. To overcome this demerit, we propose a more realistic strategy (i.e., internal-priority expansion) to sample on real-world graphs, which have diverse cluster sizes. The basic rationale is when expanding nodes from the neighborhood of a cluster C_i ($i \in [1, k]$), we can add the nodes that are tightly connected with the cluster by minimizing $|\mathrm{EN}(v, C_i)|$. Since $|\mathrm{EN}(v, C_i)|$ refers to new neighbors contributed to a cluster C_i by v if it is added, the node with lower $|\mathrm{EN}(v, C_i)|$ value can expand the core part of the cluster and help to find potential internal nodes of each cluster. Here, we consider those nodes at $\mathrm{IN}(C_i)$ (i.e., nodes with $|\mathrm{EN}(v, C_i)| = 0$) as internal nodes in cluster C_i, and they do not provide any new neighbors for C_i. These internal nodes of each cluster should be sampled into the subgraph to preserve the inherent clustering structure of the original graph.

The pseudocode for the internal-priority strategy is presented in Algorithm 3. Initially, we add top-leaders l_i into the i-th cluster C_i. Thereafter, it is divided into two phases. In the first phase, we add eligible nodes to each cluster C_i in each iteration. Specifically, if the cluster C_i has potential internal nodes in its neighbors, all will be added into the sample and labeled with C_i. Otherwise, the neighbor node which provides the least new neighbors should be sampled. In the second phase, the number of nodes might be greater than the desired size n after the first phase, so we iteratively prune out the sampled node with the minimum degree until the target number n is satisfied. Compared to the equal-size strategy, the main advantage of internal-priority expansion strategy is that clusters are not fixed to equal sizes, and the size of each cluster depends on potential internal nodes in their own neighbors.

As shown in Figure 3, for ease of understanding, a schematic diagram of the definitions is given. The algorithm includes two stages: the initialization of top-leaders and sampling expansion. (a) Initialization of top-leaders. It initializes k vertices (here, we set $k = 2$), and the selected vertices are the top-leader in

Algorithm 3 Top-leader sampling algorithm with the TLS-i

```
Input: Graph $G = (V, E)$; desired sample size $n$; number of clusters $k$; set of top-leaders $L = \{l_1, l_2, \dots, l_k\}$.
Output: Sampled subgraph $S = (V_s, E_s)$; clustering $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$.
1: $\mathcal{C} \leftarrow \emptyset$, activated nodes $\leftarrow V - L$;
2: for all $l_i \in L$ do
3:     $C_i \leftarrow C_i \cup \{l_i\}$;
4: end for
5: while $|V_s| \leqslant n$ do
6:     for all $C_i \in \mathcal{C}$ do
7:         if no more activated node is found then
8:             ###{An isolated component};
9:             Continue;
10:        else
11:            if IN$(C_i) \neq \emptyset$ then
12:                Internal_nodes $\leftarrow$ IN$(C_i)$;
13:                $C_i \leftarrow C_i \cup$ Internal_nodes;
14:                Set Internal_nodes to deactivated nodes;
15:            else
16:                $v \leftarrow$ activatednodes s.t. $\min_{v \in N(C_i)} |\mathrm{EN}(v, C_i)|$;
17:                $C_i \leftarrow C_i \cup \{v\}$;
18:                Set $v$ to deactivated node;
19:            end if
20:        end if
21:    end for
22: end while
23: $V_s \leftarrow C_1 \cup C_2 \cup \cdots \cup C_k$;
24: if $|V_s| > n$ then
25:     Iteratively remove nodes from $V_s$ in ascending order of degree until $|V_s| = n$ satisfied;
26: end if
27: for all $e = (u, v) \in E$ s.t. $\{u, v\} \subset V_s$ do
28:     $E_s \leftarrow E_s \cup \{e\}$;
29: end for
30: Return: $S$.
```

corresponding clusters. As shown in Figure 2, vertices 2 and 5 are added to the top-leader list. (b) Sampling expansion. Here, we expand the sample according to the proposed internal-priority expansion criterion until the target number of vertices n is reached (here $n = 5$). According to the internal-priority criterion, vertices 4, 6, 7, and 1 are included in the sampled counterpart. Then, the corresponding edges are attached to the vertices to obtain the sample.

5 Time complexity

In this section, the time complexities of TLS-e and TLS-i algorithms are analyzed. For ease of analysis, we consider a connected graph, where there are no isolated nodes and connected components.

In the local top-leader initialization, the sorting operation maintains the node queue in descending order of the nodes' degree. The minimum time required to perform this sorting is $O(N \times \log(N))$, and in the worst case, finding the initial k top-leaders in the queue is $O(k \times N)$, where N is the number of nodes in the original graph.

For the TLS-e, we set the size of the current cluster b_i $(i \in [1, k])$ is size (size $\leqslant \lceil \frac{n}{k} \rceil$), where n is the target number of nodes in the sample. Assume that the average degree of each node in the original graph is d, and the average ratio of external degree to total degree is μ. In the equal-size expansion, for each cluster b_i, the time complexity of finding the neighbor that minimizes the $|\mathrm{EN}(v, b_i)|$ is size $\times\, d \times u$, and the number of loops is $\lceil \frac{n}{k} \rceil$. Thus, the total time complexity of TLS-e is about $O(k \times \mathrm{size} \times d \times u \times \lceil \frac{n}{k} \rceil)$, which is less than $O((\frac{n}{k})^2)$.

Similarly, for the TLS-i, we set the iteration time of reaching the desired sample size to iter, so the complexity of finding internal nodes for each cluster b_i is $O(\mathrm{size}(b_i) \times d \times u \times \mathrm{iter})$. We cannot obtain the number of iterations and size size(b_i) of the current cluster exactly. However, the worst case is that for each cluster in each loop, only one neighbor of cluster b_i is selected, so after each inner loop, k new nodes are added to the sample, so the maximum iteration that can reach the target size is $\lceil \frac{n}{k} \rceil$. Meanwhile, the size for each cluster should be equal-size (size $\leqslant \lceil \frac{n}{k} \rceil$). Thus in the worst case, the time complexity is approximately $O((\frac{n}{k})^2)$.

As a whole, the total time complexities of TCI-i and TCI-e algorithms are both dominated by the sorting operation in top-leader initialization, which costs $O(N \times \log(N))$. Thus, the proposed sampling methods run in quasi-linear time and can be used to sample from graphs at a scale.

6 Experiments and analysis

A series of experiments are introduced to assess the sample quality of different sampling strategies on keeping their structural properties. First, we briefly described the experimental setup and the benchmarks employed. Then, we conducted empirical experiments to evaluate the sample quality using multiple metrics and reported average performances over 5 different runs.

6.1 Experimental settings

Experiments are performed on a powerful server that runs Linux CentOS. Every run utilizes a single kernel with at most 32 GB main memory and 2.40 GHz CPU. The top-leader sampling with TLS-e and TLS-i is implemented by Python 2.7[2]. For comparison, this paper considers a variety of graph sampling algorithms, including: SInetL [9], GraphSDH [11], XSN (snowball expansion sampler) [7], MRW [1], induced random node (IRN) [4], IRE [4], metropolis subgraph (MS) [6], and RW [16] sampling. It is worth noting that XSN is the only method that distinctively samples the clustering structure, and the sampled graph S is formed progressively such that the expansion factor $X(S)$ is maximized. Random walk cannot directly run on graphs with multiple connected components since it is easy to fall into dense clusters and get stuck. A variant of this method chooses a new node randomly if the "random walker" does not find new nodes until a fixed number of iterations are reached. Here, the threshold $100 \times |V|$ is employed, which is proposed in [3]. For MS sampling, it yields samples to diminish the distinction of the basic property between the sampled counterpart and original graph, and we employed the degree distribution in our experiment.

6.2 Datesets

In our experiments, classical Lancichinetti Fortunato Radicchi (LFR) [17] synthetic graphs are used since the clustering structures are explicitly known. Besides, we test on real-world networks where the ground truth relies on the available meta-data. Meanwhile, the meta-data and clustering structures are highly correlated [18, 19].

6.2.1 *Synthetic networks*

Since real-world graphs with clear ground-truth information are yet rare, synthetic networks that generate the clustering structure and hidden ground-truth have been widely used, and it has gained popularity in literature. LFR benchmark proposed in [17] has been widely used to generate artificial graphs. The LFR is based on the planted partition model, and it is capable of yielding synthetic graphs where the node degree and cluster size follow the power-law distributions. The parameter setting of LFR is briefly introduced in this section. The mixing parameter μ is critical and shows the average proportion of the external degree to the total degree for each node. When μ is set small, the clustering structure of the synthetic graph is more distinct. In our study, we choose LFR parameters based on the report in [20] to simulate real-world networks. Many LFR synthetic networks are generated, where μ ranges from 0.15 to 0.55 with the interval of 0.1, and other parameters are given in Table 1.

6.2.2 *Real-world networks*

We have considered various graphs derived from real-world networks, and they can approximately be categorized into two: large and small networks. The first five datasets are small and accurately labeled networks, which consist of Football, Karate, Dolphin, Polblogs, and Polbooks. They are widely-used to measure the accuracy of the clustering structure[3]. The rest are larger networks, and it is impossible (or very time-consuming) for some of them to process in their entirety unless we take a sample on them. The corresponding meta-data of them serve as the genuine ground truth since they are highly correlated with the clustering structure. Only the top 5000 ground-truth clusters from each network are employed since

2) The source codes of proposed methods are implemented and can be found at https://github.com/bigdataTU/sampling_clustering_structure.

3) All the small networks and ground-truth information can be found at http://www-personal.umich.edu/~mejn/netdata/.

Table 1 Default setting of LFR synthetic networks

Variable	Parameter meaning	Value range
N	The number of nodes	[1000, 10000]
μ	The mixing parameter	0.35
D	The average degree	25
$\max D$	The maximum degree	$N/10$
$\exp_1$	Exponent of the node degree distribution	-2
$\exp_2$	Exponent of the cluster-size distribution	-1
$\max C$	The maximum cluster-size	$N/10$
$\min C$	The minimum cluster-size	50

Table 2 Basic structural information of real-world large graphs[a)]

Network	$\|V\|$	$\|E\|$	# comps	C	CC	Density
Football	115	613	1	12	0.403	0.094
Karate	34	78	1	2	0.571	0.139
Dolphin	62	159	2	2	0.259	0.084
Polblogs	1224	16718	1	3	0.320	0.022
Polbooks	105	441	1	2	0.487	0.081
LFR_1000*	1000	12470	1	13	0.210	0.0249
DBLP	93432	335520	392	5000	0.708	0.00009
Friendster	220015	4031793	669	5000	0.442	0.00017
LiveJournal	84438	1521988	1464	5000	0.730	0.006
Orkut	731514	21992510	42	5000	0.247	0.00008
Youtube	39841	224235	639	5000	0.198	0.0003
LFR_10000*	10000	124237	1	29	0.3696	0.0025

a) $|E|$: number of edges; $|V|$: number of nodes; # comps: the quantity of the connected components; CC: the average clustering-coefficient; C: number of clusters; *: synthetic graph.

the quality degrades considerably after the top 5000, and they are acquired from the SNAP datasets[4)]. All graphs used are undirected and unweighted, and the basic structural information of them is shown in Table 2.

6.3 Evaluation methodology & metrics

Although sampling techniques have been evaluated in some literature [4], the sampling evaluation focuses on the basic node-level properties, such as node degree, closeness, and clustering coefficient distributions. However, basic node-level properties and corresponding measures of their representativeness are inadequate enough to evaluate how well the clustering structure of the original graph is preserved by the sample. Therefore, the evaluation methodology and metrics are described briefly on how to quantitatively assess the quality of sampling clustering structure.

6.3.1 *Evaluation methodology*

First, we obtain the clusters of the original graph to serve as the ground-truth by executing a high-quality clustering algorithm if it is not known, and then perform the same algorithm on the samples generated by these sampling baselines. It is worth noting that two accurate clustering algorithms, i.e., Infomap [21], and Blondel [22] are employed to generate clusters and serve as the ground-truth. Second, we evaluate the clustering quality of the subgraph that is produced by each sampling method using multiple metrics to validate the methods' effectiveness. Furthermore, the clustering quality of the subgraph made using each sampling approach is assessed by multiple sample quality metrics to verify the effectiveness and feasibility of algorithms.

6.3.2 *Sample quality metrics*

Concisely, we selected several sampling quality metrics to measure how representative the sample is regarding the clustering structure. Because precision and recall are viewed as two significant parts of

4) Network information can be obtained at http://snap.stanford.edu/data.

the clustering quality, each part should be individually processed without losing its properties. For this reason, firstly, we used δ-precision and δ-recall [2] to assess the sampling quality of the clustering structure between the original graph and sampled subgraph. The parameter δ is a predefined purity threshold, and it dominates the matching level between clusters of original graphs and those of the sampled subgraph. We suppose that if the matching level is not less than δ, two clusters are regarded as a right match. A good δ-precision score implies that the acquired clusters of G_s represent the ground-truth clusters of G more exactly. Meanwhile, a good δ-recall score indicates that ground-truth clusters of G are more successfully covered using the acquired clusters of G_s.

Secondly, for comparison, we have exploited several widely used quality metrics for assessing clusters in the graph. Two supervised metrics, i.e., normalized mutual information (NMI) [23] and adjusted rand index (ARI) [23], are used for assessing the clustering quality in the sampled graph. Furthermore, we considered two popular unsupervised metrics, including modularity [24] and conductance [25], to measure the clustering outcomes of the sampled counterpart. It should be noted that these metrics are only designed to evaluate the clustering quality on the entire graph, but not especially on the sample. For comparison, we used these metrics on the subgraph $G(V_s)$ of G to guarantee that it possesses the same node-set as the sample. Based on these quality metrics, sampling results on these networks are given in detail.

6.4 Overall experimental results

In this subsection, we perform extensive experimentation to explore how sampling algorithms perform in benchmark graphs using multiple metrics. For small real-graphs mentioned above, researchers have a good knowledge of the clustering structure, and a relatively large sample rate ($p = 0.5$) is taken to verify their sample qualities. But for large graphs, they are comparatively massive at scale, and it is difficult, even infeasible, to cluster on them computationally. Thus, a 15% sample rate is used to analyze the performance of sampling approaches. All results obtained on benchmark networks are presented in Tables 3 and 4. All sampling quality results using Blondel clustering are revealed, and the rest results using Infomap method show similar results. These notable results are stated as follows.

• Overall, qualitatively similar results are found across all graphs. Specifically, we can observe that TLS-i performs better than the state-of-the-art algorithms, including SInetL and GraphSDH methods. This is because TLS-i sampling can include more internal nodes into clusters that make each cluster more reliable. By sampling more internal nodes, it incorporates more influential nodes from each cluster into samples. Therefore, it is capable of producing the cluster-representative samples from the original graph. The XSN algorithm can add nodes with good expansion to the sampled subgraph. However, it starts from a single seed node, such that its performance on sparse graphs is not satisfactory. Moreover, we can observe that MRW sampling shows the poorest results for most of the graphs in terms of preserving the clustering structure. Our interpretation is that SInetL, GraphSDH, RW, and IRE sampling are biased towards high degrees' nodes, whereas MRW sampling ensure that each node has the equal probability to be sampled into the sampled subgraph without considering its importance.

• We can observe that relatively good quality scores are obtained in small real-world graphs. It is because the clustering structures are noticeable using most clustering approaches on these small graphs. However, for large graphs, they are much more difficult to obtain satisfying clustering outcomes. For instance, in Youtube and Orkut networks, many detected clusters are not located in any of the meta-data groups (low recall). Meanwhile, a few clusters of the original graph are not detected well (low precision). However, in some networks (e.g., LiveJournal and DBLP), a considerable percentage of clusters are detected well. This finding is consistent with the results of the research [26].

6.5 Impact of cluster sizes

To verify whether the cluster-size distribution has an important impact on the performance of the proposed TLS-i and TLS-e methods, a set of LFR synthetic graphs are generated in which the maximum cluster-size max$C$ ranges from 50 to 210 with the interval of 20. We also fixed the minimum cluster size to 50 and set the mixing parameter to 0.35. Other configuration settings are the same with Table 1. Meanwhile, Figure 4 shows quality scores of various metrics with synthetic graphs with different maximum cluster-size max$C$. Each sampling algorithm is executed for 5 trials, and the average score of each metric is employed. We can observe the following phenomena.

Table 3 Evaluation results of different sampling algorithms on small networks (sampling rate $p = 50\%$). It is worth mentioning that clusters are generated using Blondel clustering on the sampled subgraph and original graph. Moreover, the sample rate p is the maximum value of δ-recall

(a) Polbooks	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**0.909**	0.826	0.870	0.786	0.799	0.733	0.881	0.891	0.882
0.0-recall	**0.332**	0.252	0.146	0.141	0.271	0.166	0.149	0.321	0.315
0.5-precision	0.833	0.771	**0.870**	0.786	0.799	0.733	0.845	0.792	0.811
0.5-recall	**0.332**	0.252	0.146	0.141	0.194	0.166	0.149	0.310	0.301
ARI	0.374	0.389	0.117	0.069	**0.438**	0.169	0.134	0.372	0.370
NMI	**0.543**	0.526	0.518	0.381	0.519	0.451	0.427	0.511	0.503
Conductance	0.617	0.597	0.443	0.397	**0.625**	0.469	0.433	0.601	0.598
Modularity	0.604	**0.618**	0.571	0.517	0.519	0.556	0.462	0.607	0.601

(b) Dolphins	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**1**	1	0.941	0.969	1	1	0.944	1	0.951
0.0-recall	0.095	0.143	0.171	0.248	0.231	**0.296**	0.17	0.101	0.092
0.5-precision	**1**	1	0.941	0.969	1	1	0.944	1	0.951
0.5-recall	0.095	0.143	0.171	0.248	0.231	**0.296**	0.17	0.101	0.092
ARI	0.050	0.050	0.057	0.242	0.252	**0.352**	0.158	0.221	0.235
NMI	0.095	0.095	0.341	0.531	0.58	**0.643**	0.435	0.102	0.091
Conductance	0.316	0.436	0.433	0.492	0.501	**0.556**	0.472	0.312	0.301
Modularity	0.339	0.393	0.49	**0.545**	0.393	0.514	0.503	0.310	0.299

(c) Polblogs	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**0.959**	0.905	0.91	0.917	0.854	0.886	0.945	0.942	0.935
0.0-recall	**0.308**	0.257	0.107	0.007	0.246	0.072	0.211	0.291	0.285
0.5-precision	**0.959**	0.905	0.91	0.917	0.854	0.886	0.945	0.923	0.911
0.5-recall	**0.308**	0.257	0.107	0.007	0.246	0.066	0.211	0.254	0.211
ARI	**0.618**	0.434	0.084	0.005	0.42	0.023	0.299	0.561	0.513
NMI	**0.548**	0.481	0.338	0.31	0.464	0.130	0.399	0.512	0.482
Conductance	**0.372**	0.319	0.216	0.224	0.347	0.281	0.344	0.355	0.326
Modularity	0.413	**0.454**	0.367	0.338	0.191	0.421	0.442	0.411	0.401

(d) Football	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**1**	1	0.775	0.591	0.849	0.879	0.575	0.879	0.828
0.0-recall	0.467	**0.482**	0.229	0.177	0.446	0.446	0.341	0.453	0.412
0.5-precision	**1**	1	0.775	0.591	0.849	0.879	0.575	0.879	0.828
0.5-recall	0.467	**0.482**	0.229	0.177	0.396	0.413	0.341	0.453	0.412
ARI	**1**	1	0.334	0.244	0.780	0.831	0.762	0.780	0.765
NMI	**1**	1	0.741	0.699	0.882	0.909	0.833	0.848	0.831
Conductance	**0.586**	0.578	0.343	0.363	0.485	0.522	0.478	0.554	0.493
Modularity	**0.64**	0.631	0.467	0.456	0.521	0.591	0.61	0.621	0.598

(e) Karate	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	1	**1**	0.95	0.914	1	1	0.933	1	1
0.0-recall	0.385	**0.413**	0.319	0.188	0.382	0.146	0.271	0.382	0.319
0.5-precision	1	**1**	0.95	0.914	1	1	0.933	1	1
0.5-recall	0.385	**0.413**	0.319	0.188	0.382	0.146	0.271	0.382	0.319
ARI	0.607	**0.625**	0.44	-0.053	0.593	0.442	0.247	0.611	0.599
NMI	0.748	0.753	0.578	0.068	**0.778**	0.745	0.511	0.738	0.719
Conductance	0.545	0.542	**0.637**	0.431	0.48	0.5	0.437	0.536	0.519
Modularity	0.443	0.442	**0.494**	0.327	0.292	0.319	0.376	0.439	0.425

(f) LFR_1000	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**1**	0.711	0.666	0.654	0.675	0.919	0.889	0.779	0.737
0.0-recall	**0.497**	0.166	0.078	0.045	0.125	0.486	0.464	0.451	0.418
0.5-precision	**1**	0.711	0.666	0.654	0.675	0.919	0.889	0.715	0.702
0.5-recall	**0.497**	0.166	0.078	0.045	0.124	0.486	0.459	0.451	0.418
ARI	**1**	0.333	0.109	0.041	0.276	0.967	0.945	0.891	0.843
NMI	**1**	0.617	0.613	0.586	0.607	0.972	0.932	0.886	0.792
Conductance	**0.467**	0.22	0.279	0.302	0.21	0.448	0.353	0.413	0.401
Modularity	0.563	0.413	0.422	0.438	0.56	0.381	**0.581**	0.510	0.497

Table 4 Evaluation results of different sampling algorithms on large-scale networks (sampling rate $p = 15\%$). It is worth mentioning that clusters are generated using Blondel clustering on the sampled subgraph and original graph. Moreover, the sample rate p is the maximum value of δ-recall

(a) Livejournal	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	0.948	0.952	0.905	0.864	**0.96**	0.886	0.886	0.949	0.912
0.0-recall	0.097	0.074	0.039	**0.125**	0.106	0.072	0.072	0.093	0.102
0.5-precision	0.948	0.949	0.905	0.953	**0.959**	0.941	0.951	0.949	0.901
0.5-recall	0.094	0.065	0.038	0.065	**0.101**	0.063	0.066	0.090	0.083
ARI	**0.941**	0.837	0.64	0.848	0.66	0.839	0.846	0.891	0.883
NMI	**0.982**	0.963	0.836	0.964	0.961	0.964	0.963	0.942	0.938
Conductance	**0.967**	0.888	0.323	0.895	0.866	0.898	0.897	0.875	0.87
Modularity	**0.966**	0.9	0.806	0.899	0.944	0.897	0.898	0.922	0.897

(b) Friendster	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**0.932**	0.910	0.778	0.825	0.916	0.917	0.904	0.893	0.882
0.0-recall	**0.152**	0.125	0.021	0.031	0.102	0.1	0.089	0.127	0.114
0.5-precision	**0.925**	0.91	0.778	0.915	0.914	0.915	0.904	0.883	0.845
0.5-recall	**0.151**	0.124	0.021	0.057	0.095	0.092	0.069	0.112	0.093
ARI	0.606	0.258	0.065	**0.627**	0.581	0.605	0.598	0.434	0.412
NMI	**0.947**	0.922	0.835	0.945	0.944	0.946	0.945	0.917	0.885
Conductance	**0.610**	0.524	0.337	0.597	0.584	0.590	0.584	0.557	0.534
Modularity	**0.904**	0.766	0.504	0.787	0.755	0.768	0.786	0.581	0.543

(c) DBLP	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	0.864	0.916	0.826	**0.946**	0	0.886	0.847	0.823	0.836
0.0-recall	**0.125**	0.122	0.125	0.25	0	0.072	0.074	0.120	0.115
0.5-precision	0.862	**0.915**	0.890	0.885	0.890	0.890	0.877	0.793	0.756
0.5-recall	**0.12**	0.118	0.071	0.065	0.071	0.067	0.069	0.110	0.105
ARI	**0.034**	0.008	0.023	0.029	0.024	0.023	0.023	0.028	0.025
NMI	**0.783**	0.746	0.773	0.771	0.772	0.768	0.767	0.761	0.753
Conductance	0.62	0.581	0.692	**0.695**	0.684	0.688	0.681	0.593	0.576
Modularity	**0.881**	0.65	0.761	0.764	0.753	0.752	0.751	0.771	0.734

(d) Orkut	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**0.741**	0.695	0.642	0.647	0.662	0.668	0.706	0.639	0.614
0.0-recall	**0.129**	0.075	0.017	0.020	0.085	0.084	0.105	0.111	0.093
0.5-precision	**0.73**	0.689	0.642	0.647	0.651	0.657	0.700	0.591	0.582
0.5-recall	**0.094**	0.069	0.017	0.020	0.051	0.049	0.042	0.087	0.068
ARI	**0.219**	0.126	0.014	0.019	0.092	0.076	0.055	0.101	0.097
NMI	**0.843**	0.826	0.792	0.803	0.801	0.799	0.747	0.779	0.765
Conductance	**0.431**	0.407	0.353	0.355	0.293	0.296	0.332	0.401	0.386
Modularity	**0.791**	0.707	0.509	0.521	0.565	0.572	0.67	0.682	0.667

(e) Youtube	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	**0.8**	0.637	0.47	0.545	0.667	0.627	0.482	0.692	0.643
0.0-recall	0.102	**0.119**	0.026	0.048	0.036	0.034	0.04	0.078	0.067
0.5-precision	**0.788**	0.632	0.466	0.544	0.651	0.611	0.478	0.612	0.603
0.5-recall	0.037	**0.1**	0.025	0.047	0.028	0.028	0.048	0.043	0.035
ARI	0.053	0.039	0.046	0.082	**0.091**	0.032	0.019	0.051	0.047
NMI	0.834	0.786	0.811	**0.846**	0.794	0.8	0.571	0.811	0.791
Conductance	**0.591**	0.356	0.293	0.354	0.518	0.555	0.343	0.329	0.318
Modularity	0.453	0.556	0.395	0.518	0.543	**0.559**	0.544	0.442	0.401

(f) LFR_10000	TLS-i	TLS-e	RW	MRW	MS	IRN	XSN	SInetL	GraphSDH
0.0-precision	0.261	0.262	**0.485**	0.447	0.372	0.272	0.213	0.219	0.235
0.0-recall	0.041	**0.049**	0.012	0.021	0.037	0.032	0.027	0.038	0.032
0.5-precision	0.260	0.260	**0.481**	0.447	0.367	0.267	0.267	0.251	0.231
0.5-recall	0.039	**0.049**	0.012	0.021	0.03	0.033	0.031	0.030	0.027
ARI	**0.311**	0.027	0.011	0.003	0.188	0.238	0.238	0.281	0.238
NMI	0.515	0.537	0.527	**0.551**	0.544	0.494	0.514	0.501	0.488
Conductance	**0.408**	0.403	0.259	0.328	0.287	0.207	0.397	0.392	0.387
Modularity	**0.566**	0.544	0.347	0.458	0.396	0.323	0.556	0.494	0.397

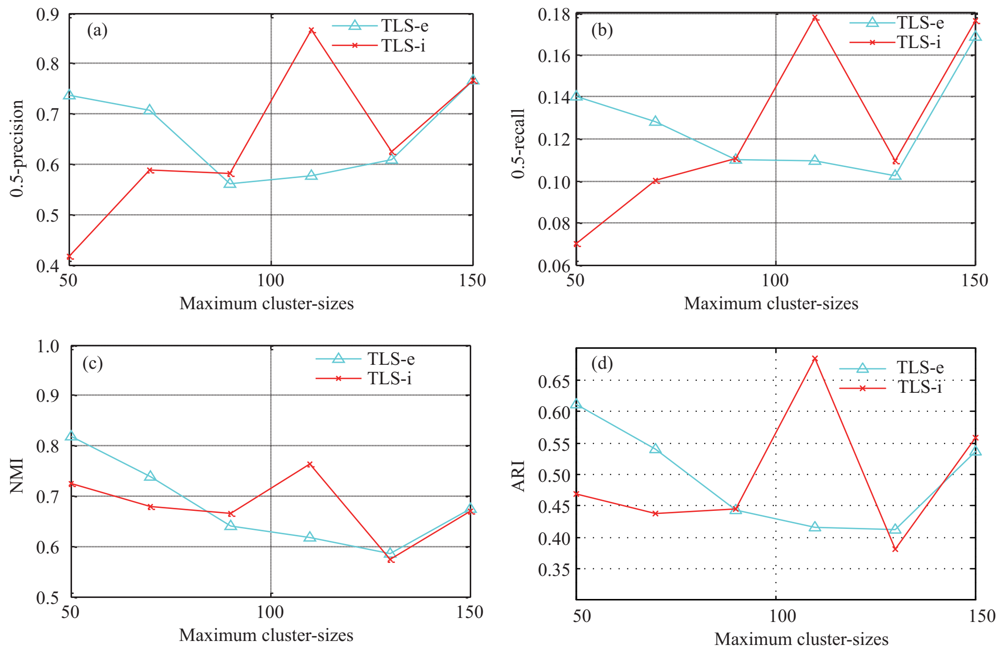


Figure 4 (Color online) Impact of cluster-size on quality metrics on the LFR benchmarks with various maximum cluster-sizes. (a) 0.5-precision; (b) 0.5-recall; (c) NMI; (d) ARI.

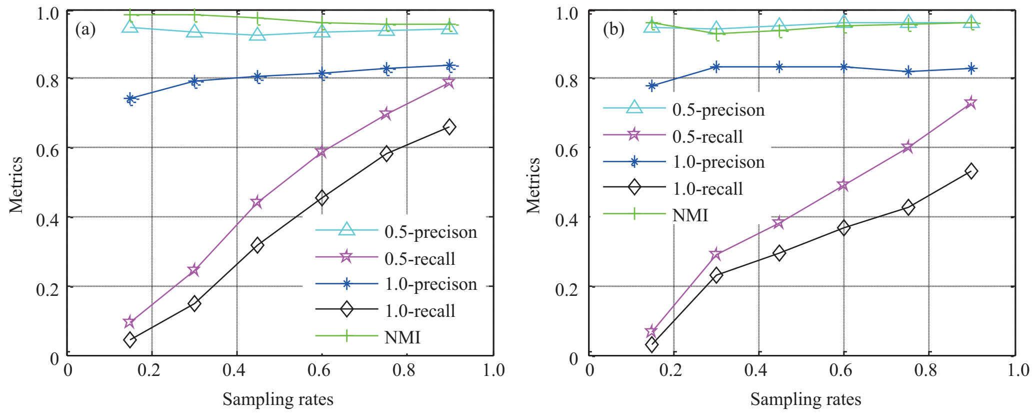


Figure 5 (Color online) Impact curve of quality metrics with the change of sampling rate p on the LiveJournal networks with Blondel clustering. (a) TLS-i; (b) TLS-e.

In general, when the maximum cluster-size max$C$ becomes larger, quality scores of TLS-i increase slightly, while the performance of TLS-e declined and is inferior to TLS-i. This happens because when the cluster sizes become more diverse, the internal-priority expansion strategy can incorporate more internal nodes of each cluster into the sampled graph, and the sample quality is stable. However, TLS-e requires to sample an equal number of nodes from each cluster and ignore the effect of the cluster-size distribution. Thus, it is more suitable for graphs with approximately similar cluster sizes and structures.

6.6 Impact of sampling rates

The sample rate p dominates the proportion of the number of nodes between the sampled and original graph. In this subsection, the impact of the sample rate p on the performance of sampling algorithms is analyzed. We set the range of the sample rate p from 0.15 to 0.90 with a step size of 0.15.

Figures 5 and 6 demonstrate the result of each measurement on DBLP and LiveJournal networks by Blondel clustering. We can see that both TLS-i and TLS-e perform well with different sampling rates. In these large-scale graphs, sample qualities keep stable or even get better with the increase of sampling rates. The increment is more significant in the graph in which the average clustering-coefficient (shown in Table 2) is high. This is because a higher average clustering coefficient makes a more precise selection

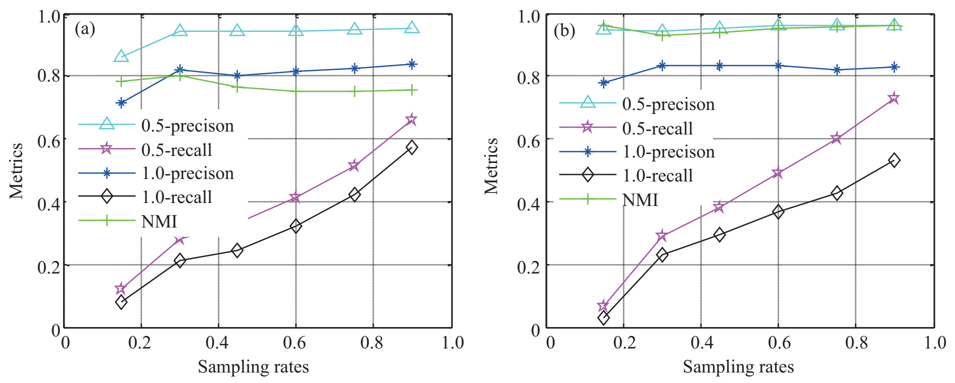


Figure 6 (Color online) Impact curve of quality metrics with the change of sampling rate p on the DBLP networks with Blondel clustering. (a) TLS-i; (b) TLS-e.

of nodes in each cluster, which causes a more competent decision to incorporate each cluster's influential nodes.

7 Conclusion and future work

In this paper, we devised two variants of sampling algorithms to yield samples from large-scale graphs such that sampled subgraphs are representative of the original graph on the clustering structure. The proposed methods greedily select top-leaders from different clusters and then expand eligible nodes, which are near the top-leaders. We empirically evaluate the proposed methods compared with the state-of-the-art sampling algorithms using various quality metrics. Moreover, we demonstrate that the proposed methods can retain the inherent clustering structure well when sampling is performed on these large-scale networks.

In future work, we will investigate more options to sample large-scale graphs while preserving multiple properties simultaneously. Besides, we need to further devise credible quality metrics to assess the sample quality quantitatively, and these metrics could provide more insights into the sample performance from different perspectives.

Acknowledgements This work was supported by Natural Science Foundation Youth Fund Project (Grant No. 62002384), China Postdoctoral Science Foundation Funded Project (Grant No. 47689), and Zhengzhou City Collaborative Innovation Major Project (Grant No. 162/32410218).

References

1 Rozemberczki B, Kiss O, Sarkar R. Little ball of fur: a python library for graph sampling. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM), 2020
2 Zhang J P, Pei Y L, Fletcher G, et al. Evaluation of the sample clustering process on graphs. IEEE Trans Knowl Data Eng, 2020, 32: 1333–1347
3 Leskovec J, Faloutsos C. Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006. 631–636
4 Ahmed N K, Neville J, Kompella R. Network sampling: from static to streaming graphs. ACM Trans Knowl Discov Data, 2014, 8: 1–56
5 Zhang J P, Pei Y L, Fletcher G H, et al. Structural measures of clustering quality on graph samples. In: Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2016. 345–348
6 Hübler C, Kriegel H P, Borgwardt K, et al. Metropolis algorithms for representative subgraph sampling. In: Proceedings of the 8th IEEE International Conference on Data Mining, 2008. 283–292
7 Maiya A S, Berger-Wolf T Y. Sampling community structure. In: Proceedings of the 19th International Conference on World Wide Web, 2010. 701–710
8 Wang F, Cheung G N, Wang Y C. Low-complexity graph sampling with noise and signal reconstruction via neumann series. IEEE Trans Signal Process, 2019, 67: 5511–5526
9 Jiao B, Shi J M, Zhang W S, et al. Graph sampling for Internet topologies using normalized Laplacian spectral features. Inf Sci, 2019, 481: 574–603
10 Zhou Z G, Shi C, Shen X L, et al. Context-aware sampling of large networks via graph representation learning. IEEE Trans Visual Comput Graph, 2021, 27: 1709–1719
11 Hu J, Dai G, Wang Y, et al. Graphsdh: a general graph sampling framework with distribution and hierarchy. In: Proceedings of IEEE High Performance Extreme Computing Conference (HPEC), 2020. 1–7
12 Mall R, Langone R, Suykens J A K. FURS: fast and unique representative subset selection retaining large-scale community structure. Soc Netw Anal Min, 2013, 3: 1075–1095
13 Barabási A L, Albert R. Emergence of scaling in random networks. Science, 1999, 286: 509–512
14 Khorasgani R R, Chen J, Zaiane O R. Top leaders community detection approach in information networks. In: Proceedings of the 4th SNA-KDD Workshop on Social Network Mining and Analysis, 2010

15 Salehi M, Rabiee H R, Rajabi A. Sampling from complex networks with high community structures. Chaos, 2012, 22: 023126
16 Lovász L. Random walks on graphs. Comb Paul Erdos Eighty, 1993, 2: 1–46
17 Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Phys Rev E, 2008, 78: 046110
18 Yang J, Leskovec J. Structure and overlaps of ground-truth communities in networks. ACM Trans Intell Syst Technol, 2014, 5: 1–35
19 Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. Knowl Inf Syst, 2015, 42: 181–213
20 Emmons S, Kobourov S, Gallant M, et al. Analysis of network clustering algorithms and cluster quality metrics at scale. PLoS ONE, 2016, 11: 0159161
21 Rosvall M, Bergstrom C T. Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci USA, 2008, 105: 1118–1123
22 Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks. J Stat Mech, 2008, 2008: 10008
23 Fortunato S, Hric D. Community detection in networks: a user guide. Phys Rep, 2016, 659: 1–44
24 Newman M E J. From the cover: modularity and community structure in networks. Proc Natl Acad Sci USA, 2006, 103: 8577–8582
25 Kannan R, Vempala S, Vetta A. On clusterings: good, bad and spectral. J ACM, 2004, 51: 497–515
26 Hric D, Darst R K, Fortunato S. Community detection in networks: structural communities versus ground truth. Phys Rev E, 2014, 90: 062805