Clustering Affiliation Inference from Graph Samples

Jianpeng Zhang, Kaijie Zhu, Yulong Pei, George Fletcher, Mykola Pechenizkiy
Eindhoven University of Technology
The Netherlands
j.zhang.4,k.zhu,y.pei.1,g.h.l.fletcher,m.pechenizkiy@tue.nl

ABSTRACT

Graph sampling is a widely-used approach to address the scalability issue when analyzing large-scale graphs. Several promising cluster-preserving sampling algorithms have been proposed. However, once the clustering structure on a sampled graph is obtained, we may still need a method to infer the clustering affiliations of all other nodes in the original graph from the clustered nodes in the sampled subgraph. In this paper, we present a new two-stage clustering inference (*TCI*) method to infer clustering affiliations of all nodes in the original graph. *TCI* is composed of two stages: 1) initialization of clustering affiliations for unsampled nodes based on computed neighborhood affiliation information; 2) label propagation for the whole graph. Our experimental results demonstrate that the proposed *TCI* method in conjunction with any considered cluster-preserving sampling strategy is capable of inferring the clustering affiliation of the population commendably, and it performs better than the competing methods.

CCS CONCEPTS

• **Networks** → *Network algorithms*; *Network performance evaluation*; *Network structure*;

KEYWORDS

Graph Sampling, Clustering Structure, Population Inference, Representative Sample

ACM Reference Format:
Jianpeng Zhang, Kaijie Zhu, Yulong Pei, George Fletcher, Mykola Pechenizkiy. 2018. Clustering Affiliation Inference from Graph Samples. In *Proceedings of MLG@KDD workshop (MLG@KDD'18).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Graph, as a generic data structure, can be a good representation of the complex relationships (namely, *edges*) among entities (namely, *nodes*) of networks [5]. In a social network, for instance, nodes may represent people and links may represent relationships (e.g., friendships), interactions (e.g., emails transmitted, physical proximity), or homogeneity (e.g., similar books purchased). Nowadays, in a wide range of applications, "big" graphs are becoming prevalent. For instance, Facebook has reported to have more than one billion active users. With 8 bytes for user ID and 100 friends per user, storing the raw edges might take about 1 billion*100*8 bytes =800 GB. Analyzing the clustering structure on such big graphs has become of great importance in various application such as detecting ransomware attacks and abnormal connection in social media.

Since contemporary graphs might contain overwhelming amount of nodes and edges and have obvious characteristics of "big" data, how to effectively process these big graphs is very critical in practice and theory. One direction is to design more efficient and scalable clustering algorithms or make full use of parallelization or distributed computing to speed up the process, but these existing scalable methods are not always easily available and sometimes impractical [15]. Another feasible solution that has received more attention is to take a small sample on the graph and conduct the traditional analysis on the sampled subgraph. Through sampling a representative subgraph, clustering analysis can be performed on the sampled graph to detect the inherent clustering structure instead of the original graph (namely, population) [15, 17]. In this work we address the underlying questions of the latter sampling-based methodology.

In previous studies, several cluster-preserving sampling algorithms [7, 11, 15–17, 23, 30] have been put forward. The sampling quality is typically judged based on how well the sample helps to capture the structural properties (e.g., clustering structure) [15, 29]. Especially, a real-world network often exhibits underlying clustering structure that is not explicitly measurable. Such implicit structure should be inferred for various applications (e.g., inferring the community of drug users from samples). However, the approaches proposed so far typically do not address this underlying problem: how can we infer the clustering affiliations of all the nodes in the original graph that were not sampled? The existing clustering inference methods either have poor inferring quality [13] or are too time-consuming [2, 19] for achieving this task. In this paper, we focus on using the sampled graph to infer clustering affiliations for the unsampled nodes of the original graph. We refer to this task as *population inference* [13, 19].

Population inference has a different emphasis in comparison with *representative subgraph sampling* [24]. The population inference focuses on how a given sampled subgraph can be used best to draw inferences about the larger population while representative subgraph sampling concentrates on how best to construct representative subgraph samples. Note that the nodes in the sample are not independent with respect to the unsampled nodes of the population, which introduces a bias to learning and inference procedures, thus the common practice that divides the nodes into training and test sets is difficult. Sampled nodes with known labels can serve two roles: one is to serve as training set, the other is to give the background knowledge during clustering inference [14].

MLG@KDD'18, August 2018, London, United Kingdom

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

Table 1: Basic notation & definitions

G	The original graph
V	The set of nodes within the graph
E	The set of edges within the graph
N	The number of nodes in V
M	The number of edges in E
S	The sampled subgraph of G
p	The sample rate of nodes
V_s	The set of nodes within the sample
E_s	The set of edges within the sample
n_s	The number of sampled nodes
m_s	The number of sampled edges
$Label_V$	The set of clustering labels of nodes in V
$Label_{V_s}$	The set of clustering labels of nodes in V_s
$Label_V(v_i)$	The clustering label of the i^{th} node in V

In this work, we study how to infer the clustering affiliations of unsampled nodes in the population by using the representative sample. Our main contributions are as follows:

- We present a new two-stage clustering inference (*TCI*) method to infer clustering affiliations of nodes in the original graph. Basically, it is composed of two stages: 1) initialization of clustering affiliations for unsampled nodes; 2) label propagation for the whole graph in which the sample's labels are efficiently propagated to the unlabelled nodes. It provides a feasible solution to infer the clustering structure of the population.
- We provide empirical evidence from an extensive experimentation on a variety of synthetic and real-world graphs demonstrating that the proposed *TCI* method is capable of inferring the clustering affiliation of the original graph commendably. *TCI* performs better than the competing methods. The results are consistent for all considered cluster-preserving sampling strategies.

The rest of the paper is organized as follows. In Section 2 we introduce the problem statement of *population inference* and in Section 3 we provide a brief overview of state-of-the-art methods. In Section 4 we discuss the proposed two-stage clustering inference (*TCI*) method. In Section 5 we discuss its computational complexity. In Section 6 we present and discuss the results of our empirical study. We conclude this paper in Section 7.

2 PROBLEM STATEMENT

Assuming that a representative subgraph S is given which is sampled from the original graph G, and the clustering labels $Label_{V_s}$ of nodes in S are known explicitly[1], our goal is to infer the clustering affiliations for the nodes v where $v \in V - V_s$ by using the sample S. Formally, we define it as follows:

Definition 2.1. (Population inference). Given a partial labelled graph $G = (V, E, Label_V)$ where V is a finite set of nodes, $E \subseteq V \times V$ is a set of edges and $Label_V \subseteq \mathbb{R}^+$ is the set of clustering labels of nodes V. Here $\mathbb{R}^+$ denotes the set of positive real numbers. Specially, $Label_V(v_i) \in Label_V$ is the clustering label of node $v_i \in V$ and partial known label set $Label_{V_s} \subseteq Label_V$ for a subset of nodes $V_s \subseteq V$ in the sample S, the *population inference* is to infer the clustering labels of the unsampled node-set $U = V - V_s$ of the population.

Through the *population inference*, we can obtain the clustering labels $Label_V$ for all the nodes in the original graph. For any two nodes $v_i,\ v_j \in V$, $Label_V(v_i) = Label_V(v_j)$ if and only if v_i and v_j are members of the same cluster.

[1]Note that the clustering labels of nodes in the sample can be obtained from the ground-truth if known. Otherwise, they can be produced by executing a credible clustering algorithm.

3 RELATED WORK

At present, there exist little work in the field of *population inference* and some related researches are briefly summarized. [19] studied how to expand a cluster from few given seed nodes. It is assumed that a sample contains nodes from a single cluster and they proposed expanding methods to grow the sample to include all members of the single cluster, but it can not extend to handle the clustering inference problem of multiple clusters. The kernel spectral clustering (KSC) method was first proposed in [2] and extended to network-structured data in [10] and [18]. It is based on a weighted kernel PCA formulation and the model is built in a primal-dual optimization framework. The model has a good out-of sample extension property which allows for inferring clustering affiliation for unseen nodes. KronEM [8] was proposed to infer missing nodes and edges in graphs. It uses the EM model to estimate the model parameters as well as infer missing nodes and edges of the graph. However, it considers the link-prediction problem where both edges and nodes are missing. In [13] three collective inferencing schemes (i.e., *iterative classification (IC)*, *gibbs inference (GI)* and *relaxation labeling (RL)*) were devised to classify unlabelled nodes in a graph if a subset of labelled nodes is given. Their results demonstrated that *RL* model performs better than others. However, they are too time-consuming to infer the clustering affiliations of nodes in large-scale graphs. For more details about these collective inferences, one may refer to [13].

4 *TCI*:TWO-STAGE CLUSTERING INFERENCE

As we discussed, the objective of the clustering inference is to assign the clustering affiliations for all the unsampled nodes in the population. However, existing clustering inference methods either have poor inferring quality [13] or are too time-consuming [2, 19]. For this purpose, we propose a new *two-stage clustering inference* (*TCI*) method to label the unsampled nodes. The rationale is that if samples are truly representative of clustering structure of the larger population, the clustering structure on the sample should generalize well to the nodes that are not sampled. That is, using a subgraph S, we attempt to infer the clustering affiliation for all the unsampled nodes v where $v \in V - V_s$. Since low-degree nodes are not vital for inherent clustering structure, the clustering inference method is composed of two stages: 1) initialization of clustering affiliations for unsampled nodes; 2) label propagation for the whole graph.

As Algorithm 1 shows, in the first stage, first we sort unlabelled nodes in a descending order according to the ratio of their labelled neighbors such that the label information of samples are sufficiently utilized (line 1-3). Second for each unlabelled node from the sorted queue U, its label is assigned to the label that has maximum occurrence among the neighbours for which clustering assignments are known already. Then we remove it from the unlabelled set U and

Algorithm 1 Inferring Clustering Affiliation from Samples

Require:
Graph: $G = (V, E)$;
Sampled subgraph: $G_s = (V_s, E_s)$;
Clustering labels of the sampled nodes V_s: $Label_{V_s}$
Ensure:
Clustering labels of V in G: $Label_V$.
1: Obtain the unsampled nodes: $U \leftarrow V - V_s$
2: ###{**The coarse screening**}
3: Sort the nodes of the node-set U according to the fraction of the neighbours in V_s
4: ###{Handle connected components}
5: **while** no more nodes in U can be labelled **do**
6: **for** each node $u \in U$ that pops up from U **do**
7: ###{Set its label to the maximum occurring label among the neighbours in V_s}
8: $Label_V(u) \leftarrow \arg\max_{Label_{V_s}}[Count(Label_{V_s}(ne))]$ s.t. $ne \in N_s(u)$
9: $U \leftarrow U - u$
10: $V_s \leftarrow V_s \cup u$
11: **end for**
12: **end while**
13: ###{Handle isolated nodes}
14: **if** $u \in U$ is a isolated node **then**
15: $Label_V(u) \leftarrow$ single-node cluster
16: **end if**
17: ###{**The fine adjustment**}
18: Set the initial node labels: $initial = Label_V$
19: Set the fixed labels whose labels should not change: $fixed = Label_{V_s}$
20: $Label_V \leftarrow$ Label_Propagation(G, initial, fixed)
21: **return** $Label_V$

add the node into the labelled node-set V_s until no more nodes in V can be labelled (line 4-12). For the isolated nodes which are not connected to the sample, they are designated into its own single-node cluster (line 13-16).

In the second stage we utilize *label propagation* (*LP*) clustering algorithm [22] to optimize the clustering structure for the original graph. *LP* algorithm iteratively simulates a process in which each node in the graph picks the most frequent label in its neighbours in each iteration. In summary, the process of *LP* has five steps:

Step 1. Initialize the clustering labels $Label_V$ of all nodes V in the graph.
Step 2. Set $t = 1$;
Step 3. Arrange the nodes in the graph in a random order and set it to V;
Step 4. For each node $v \in V$ chosen in that specific order, the label function is defined as $Label_V(v) = f(\cdot)$. The function $f(\cdot)$ returns the clustering label occurring with the highest frequency among neighbours. Ties are broken randomly and the order in which the nodes are updated is randomized for each iteration;
Step 5. The process repeats until all the nodes reach a consensus. Otherwise, set $t = t + 1$ and go to Step 3.

The advantages of *LP* algorithm include that (i) it does not need a pre-defined objective function; (ii) its time complexity increases quasilinearly with respect to the size of the graph, which will be detailed described in next Section; (iii) it has the ability to set the initial labels and also allows some labels to be fixed. Thus we set the labeling results of the coarse stage to the initial labels (line 18), and assign the clustering labels of the sampled nodes to fixed labels whose labeling should not change during the iteration (line 19-20). In this way, the *TCI* method is able to assign and infer the clustering labels to the population effectively and accurately.

5 COMPUTATIONAL COMPLEXITY

The computational complexities of different components of the *TCI* algorithm are analyzed as follows:

- In the first stage, firstly, the sorting operation aims to maintain the unsampled nodes in descending order based on the fraction of the labelled neighbours in V_s. The minimum time required to perform this sorting is $O((N - n_s) * log(N - n_s))$ where N is the number of nodes in the original graph and n_s is the number of nodes in the sample. Secondly, the label of each unsampled node needs to be assigned to the most frequent label among its neighbours. Here we assume that the average number of labelled neighbours of each node is c, then the complexity of finding the most frequent label is $O(c * (N - n_s))$.
- For the second stage, the time complexity of *LP* algorithm increases quasilinearly with the size of the graph, that is, $O(M + N)$ where M is the number of edges in the original graph. Note that the number of edges M should be much less than $\frac{N(N-1)}{2}$ in real-world graphs, which are usually very sparse.

Overall, the total time complexity of *TCI* is $O(N * logN + M)$. Thus the proposed *TCI* run in quasi-linear time such that it can be employed to process large-scale graphs.

6 EXPERIMENTAL EVALUATION

We present a set of experiments to evaluate the inference quality of the proposed *TCI* method. First we describe the experimental setup for the evaluation. Then we conduct the experiments on benchmark graphs and analyze the obtained results.

6.1 Experimental Setup

The experiments are carried out on a Linux server runing CentOS. Each run employs a single core with 2.40 GHz CPU and at most 32 GB main memory. We have implemented our *TCI* method in *Python* 2.7. For a fair comparison we also consider a diverse collection of collective inferencing methods [13] which includes: *iterative classification (IC)*, *gibbs inference (GI)*, *relaxation labeling (RL)*.

In order to obtain the representative subgraph, we employ several graph sampling algorithms including *induced random edge (IRE)* [1], *induced random vertex (IRV)* [1], *random walk (RW)* [12], *metropolis subgraph (MS)* [7], *metropolized random walk (MRW)* [4, 20], *XSN* [15] and *TLS*[2] sampling [28]. Note that *XSN* is a sampling

[2]Two variants of top-leader sampling algorithms (i.e., *TLS-e* and *TLS-i*) to produce representative samples and they are capable of retaining the clustering structure effectively.

algorithm that explicitly considers the clustering structure, and the sample S is selected such that it maximizes the expansion factor $X(S)$. For *MS* sampling, it produces samples to minimize the difference of the basic property (here we use degree distribution) between the sample and the original population. *RW* sampling cannot work directly on graphs with multiple connected components and easily get trapped in dense clusters. A variant of this approach selects a node at random if the random walk does not discover new nodes after a fixed number of iterations. In our experiments, we use the threshold proposed in [11], which is $100 * |V|$.

6.2 Datasets

We present a series of experiments on widely used *LFR* [9] synthetic graphs and real-world graphs. First, the *LFR* generator is employed as synthetic benchmarks because it takes a given ground-truth clustering as an input and assumes that both node degrees and cluster sizes have power-law distributions. In *LFR* benchmarks, a set of parameters needs to be given as necessary. The mixing parameter μ is vital and reflects the average ratio of external degree to total degree for each node. The larger μ is, the more indistinct the clustering structure of the benchmark is. We generate a set of *LFR* benchmarks in which μ ranges from 0.25 to 0.55 with the interval of 0.1. The other configuration settings are outlined in Table 2.

Table 2: The parameters of the LFR benchmarks

Parameter	Description	Value
N	The number of vertices	1000
D	The average degree	25
$maxD$	The maximum degree	$N/10$
μ	The mixing parameter	[0.25: 0.10: 0.55]
e_1	The degree distribution exponent	-2
e_2	The cluster size distribution exponent	-1
$minC$	The minimum cluster-size	50
$maxC$	The maximum cluster-size	$N/10$

Second, we further evaluate the clustering results on real-world graphs in which the meta-data are available, and the meta-data have been proved to be a good proxy for clustering structure in other studies [26, 27]. They can roughly be classified in two groups: small and large networks. The first five datasets including *Karate*, *Football*, *Dolphin*, *Polbooks* and *Polblogs* networks are relatively small[3]. The other graphs are from the Stanford Large Network Dataset Collection[4]. They are relatively large and numerous graph clustering algorithms [6, 21] are computationally costly for them. The large-scale graphs all have the corresponding meta-data that correlate to the clustering structure to serve as the underlying ground-truth. Here only the top 5000 ground-truth clusters from each network are used for evaluation since the quality metrics degrade considerably after the top 5000.

6.3 Evaluation

In the present study, researchers have only considered the sample quality with respect to basic properties [1] (e.g. degree, clustering coefficient, shortest-path, k-core distributions). However, the basic properties and the corresponding measures of representativeness are inadequate for the new target: how well the clustering structure of the original graph can be recovered by the counterpart of samples. Hence, we describe our methodology on how to evaluate the clustering structure quantitatively.

Table 3: Summary of real-world graphs used in the experiments. Abbreviations are described as follows: N: number of nodes; M: number of edges; C: number of clusters; *# comps*: number of components; *CC*: the average clustering-coefficient for all nodes;

Dataset	N	M	C	# comps	Density	CC
Karate	34	78	2	1	0.139	0.571
Football	115	613	12	1	0.094	0.403
Dolphin	62	159	2	2	0.084	0.259
Polbooks	105	441	2	1	0.081	0.487
Polblogs	1,224	16,718	3	1	0.022	0.320
LiveJournal	84,438	1,521,988	5000	1464	0.006	0.730
Friendster	220,015	4,031,793	5000	669	0.00017	0.442
Orkut	731,514	21,992,510	5000	42	0.00008	0.247
DBLP	93,432	335,520	5000	392	0.00009	0.708
Youtube	39,841	224,235	5000	639	0.0003	0.198

6.3.1 Evaluation Methodology. First we sample a representative subgraph sample S using any sampling strategy, and then execute the clustering algorithm[5] on the subgraph S. Using these clustering assignments of nodes in S, we infer the clustering affiliations for the remaining nodes $U = V - V_s$ by multiple inference methods. Analogously, we execute the same clustering algorithm on the whole graph G and obtain the clustering assignments of nodes in G. Afterwards, we obtain two sets of clustering assignments to the nodes in V: one resulting from the inference method and the other resulting from execution of the clustering algorithm on the whole graph. Thus, for all the nodes V in the original graph, we evaluate these two sets of clustering assignments using multiple metrics to validate the methods' effectiveness.

6.3.2 Evaluation Measurements. We now briefly depict the quality metrics we employ to assess how representative the sample counterpart is with respect to the intrinsic clustering structure in the large population.

δ-precison* & *δ-recall: In general, the recall and precision are two main aspects of the clustering quality, each aspect needs to be handled separately without losing both's significance. Thus, firstly we employ *δ-precison* and *δ-recall* proposed in [28, 29] to capture clustering structure's differences between the sampled counterpart and the original graph. Higher value of *δ-precision* indicates that the obtained clusters of S are more precisely representative of the ground-truth clusters of G while higher value of *δ-recall* means the ground-truth clusters of G are more successfully covered by the obtained clusters of S. Obviously, *δ-precison* and *δ-recall* should be high if nodes grouped together in the clusters of sample are also

[3]They can be obtained in http://www-personal.umich.edu/~mejn/netdata/

[4]The graphs and the corresponding meta-data are complete and publicly available at http://snap.stanford.edu/data

[5]Note that in order to obtain method-independent results, two scalable and credible clustering algorithms including: *Blondel* [3] and *Infomap* [25] are utilized to produce the ground-truth clusters. Here, experimental results using *Blondel* clustering are presented while clustering results using *Infomap* algorithm [25] exhibit similar behaviors and do not report in this experiment.

Table 4: Inferring clustering affiliations for *LFR* benchmarks from the representative samples which are produced by different sampling algorithms at 20% sampling rate. Abbreviations are described as follows: δ-P: δ-precision; δ-R: δ-recall; *NMI*: normalized mutual information; T: the time-consumption of the sampling method.

Networks	Sampling	Quality scores of *TCI*			
		0.5-precision	0.5-recall	NMI	T (s)
LFR_1000 (u=0.25)	TLS-i	**0.841**	**0.972**	**0.866**	3.43
	TLS-e	0.810	0.801	0.279	3.49
	RW	0.396	0.814	0.176	**1.13**
	MS	0.699	0.827	0.265	8.42
	XSN	0.821	0.988	0.865	60.63
	IRV	0.789	0.824	0.253	1.15
	MRW	0.654	0.875	0.739	1.14
LFR_1000 (u=0.35)	TLS-i	**0.744**	**0.920**	**0.833**	3.624
	TLS-e	0.522	0.819	0.207	4.061
	RW	0.228	0.826	0.164	**1.171**
	MS	0.662	0.887	0.797	8.588
	XSN	0.659	0.822	0.232	61.360
	IRV	0.670	0.840	0.196	1.208
	MRW	0.536	0.871	0.717	1.381
LFR_1000 (u=0.45)	TLS-i	0.706	0.909	0.782	4.99
	TLS-e	0.720	0.915	0.775	4.15
	RW	0.113	0.815	0.125	**1.16**
	MS	0.718	0.820	0.230	8.52
	XSN	0.653	0.922	0.744	62.03
	IRV	**0.740**	**0.939**	**0.789**	1.17
	MRW	0.275	0.828	0.513	1.19
LFR_1000 (u=0.55)	TLS-i	0.056	0.823	0.102	5.10
	TLS-e	0.139	0.815	0.113	4.54
	RW	0.008	0.814	0.104	1.24
	MS	**0.429**	0.840	**0.168**	8.45
	XSN	0.017	0.824	0.091	58.38
	IRV	0.353	**0.848**	0.110	1.21
	MRW	0.125	0.825	0.112	**1.14**

grouped together in the clusters of original graph with high values corresponding to better representativeness, and vice versa.

Normalized Mutual Information (NMI): It is built on the Shannon entropy of information theory. Formally, the entropy $\mathcal{H}$ of a clustering C is

$$\mathcal{H}(C) = -\sum_{C \in C} P(C) * \log_2 P(C),$$

where C is a cluster in C and $P(C) = |C|/N$ and N is the size of nodes. Analogously, for another clustering $\mathcal{D}$, the conditional entropy $\mathcal{H}(C|\mathcal{D})$ is defined as

$$\mathcal{H}(C|\mathcal{D}) = \sum_{C \in C} \sum_{D \in \mathcal{D}} P(C, D) * \log_2 \frac{P(C)}{P(C, D)},$$

where $C \in C, D \in \mathcal{D}$ and the joint probability $P(C, D) = |C \bigcap D|/N$. Then the mutual information $\mathcal{I}$ of C and $\mathcal{D}$ is defined as

$$\mathcal{I}(C, \mathcal{D}) = \mathcal{H}(C) - \mathcal{H}(C|\mathcal{D}). \quad (1)$$

Then, the *NMI* is obtained by dividing the mutual information by the arithmetic average of the entropies of C and $\mathcal{D}$

$$\text{NMI}(C, \mathcal{D}) = \frac{2 * \mathcal{I}(C, \mathcal{D})}{\mathcal{H}(C) + \mathcal{H}(\mathcal{D})}. \quad (2)$$

The *NMI* value ranges the interval [0, 1] and high value of *NMI* means that high correlation between the two clusterings.

6.4 Inferring Clustering Affiliations from Graph Samples

In the first experiment, in order to show that *TCI* method can effectively infer the clustering structure of unsampled nodes from the representative sample, we generated *LFR* benchmarks containing 1000 nodes with different mixing parameter μ. The larger μ is, the more indistinct the clustering structure of the benchmark is. The clustering inference results from the 20% sample of multiple sampling strategies are shown in Table 4. Note that we only present δ-*precision* and δ-*recall* in which ($\delta = 0.5$) and these quality metrics with varying purity thresholds δ exhibit similar behaviors.

From these empirical results, we can derive that the proposed *TCI* method in conjunction with cluster-preserving sampling methods (e.g., *XSN*, *TLS-e* and *TLS-i*) performs well on inferring the clustering structure of the population in most benchmark graphs. Especially, the combination of *TLS-i* sampling and *TCI* inference outperforms other combinations when the clustering structures of graphs are distinct (i.e., $\mu \leq 0.45$). These results verify that (i) *TLS-i* sampling produces the sampled subgraph which preserves the inherent clustering structure well; and (ii) *TCI* inference method provides a feasible solution to infer clustering affiliations of the original graph.

6.5 Comparison of Inference Methods

In the second experiment, multiple inference methods [13], i.e., *IC*, *GI* and *RL*, are employed to infer clustering affiliations for unsampled nodes of the population. For large-scale graphs, we set the sample rate to 20% which has proved to be sufficient to preserve the topology structure [11, 15]. Meanwhile, for small graphs, we take a relatively large sample rate (i.e., $p = 0.5$) to show the inference quality from the samples. The inferring quality of the population is further validated by three quality metrics, i.e., δ-*precision*, δ-*recall*, *NMI*. Besides, we also record the execution time of each method. Please note that these competing methods employed produce a probability distribution over the possible clustering assignments for each node. The node's probability distribution represents the strength of clustering affiliation for the specific node. For a fair comparison, we take the one with the highest probability as its clustering label.

The clustering inference results of the whole graph are shown in Table 5. We can observe that the proposed *TCI* algorithm outperforms competing methods in terms of clustering quality and execution time. The overall conclusions can be drawn as follows:

- In general, the proposed *TCI* algorithm shows better clustering qualities in terms of δ-*precision*, δ-*recall* and *NMI* in these real-world graphs. Note that the entries that are marked with '–' in the table corresponds to algorithms that execution time exceeding 48 hours or returned out of memory errors.
- The proposed *TCI* algorithm is more efficient than the state-of-the-art algorithms in small graphs. Moreover, for large scale graph, we observe that only the proposed *TCI* algorithm is able to run on all these large-scale networks, and the execution time of competing algorithms are not reported in

Table 5: Clustering metrics by each inference method on real-world graphs using the *TLS-i* sampling. Each row is a specific graph with *TLS-i* sampling and each column is the inference algorithm employed. All results are the mean values over 10 repetitions on each graph. Note that for small graphs the sampling rate is set to $p = 0.5$ while for big graphs $p = 0.2$. Abbreviations are described as follows: δ-P: δ-precision; δ-R: δ-recall; *NMI*: normalized mutual information; T: the time consumption of clustering inferring methods.

		Proposed TCI				Relaxation				Gibbs				Iterative			
Dataset	Sampling	δ-P	δ-R	NMI	T (s)	δ-P	δ-R	NMI	T (s)	δ-P	δ-R	NMI	T (s)	δ-P	δ-R	NMI	T (s)
Karate	TLS-i	0.91	0.68	0.42	**0.01**	**0.99**	**0.81**	**0.67**	0.35	0.99	0.78	0.66	0.58	0.95	0.66	0.46	0.36
Football	TLS-i	**0.73**	**0.86**	**0.79**	**0.01**	0.71	0.81	0.77	0.47	0.71	0.81	0.77	1.42	0.71	0.86	0.79	0.37
Polblogs	TLS-i	**0.90**	**0.59**	0.27	**0.59**	0.87	0.45	0.20	1.27	0.87	0.44	0.20	23.86	0.89	0.58	**0.29**	1.27
Dolphins	TLS-i	0.90	0.63	0.30	**0.01**	0.90	0.63	0.31	0.35	0.90	0.63	0.31	0.67	**0.92**	**0.68**	**0.36**	0.38
Polbooks	TLS-i	0.69	**0.65**	**0.51**	**0.01**	0.68	0.46	0.52	0.49	0.68	0.46	0.52	1.11	**0.79**	0.61	0.49	0.38
email-Eu-core	TLS-i	0.57	0.60	0.47	**0.41**	**0.63**	**0.71**	**0.60**	1.29	0.61	0.69	0.59	42.17	0.51	0.63	0.48	1.10
LiveJournal	TLS-i	**0.83**	**0.98**	**0.79**	**1059.33**	-	-	-	-	-	-	-	-	-	-	-	-
DBLP	TLS-i	**0.92**	**0.62**	**0.58**	**244.23**	-	-	-	-	-	-	-	-	-	-	-	-
Amazon	TLS-i	**0.99**	**0.98**	**0.87**	**7.07**		-	-	-	-	-	-	-	-	-	-	-
Youtube	TLS-i	**0.59**	**0.89**	**0.47**	**71.74**	-	-	-	-	-	-	-	-	-	-	-	-
Friendster	TLS-i	**0.75**	**0.97**	**0.82**	**1313.98**	-	-	-	-	-	-	-	-	-	-	-	-

the table because the source codes used for its computation cannot handle the massive size of datasets[6].

Thus empirical results demonstrate that *TCI* algorithm is capable of inferring the clustering structure and performs better than state-of-the-art algorithms. Meanwhile, it runs much faster than competing algorithms for large-scale graphs.

7 CONCLUSION

Clustering of large graphs is often computationally prohibitive. Hence, (clustering-preserving) graph sampling typically is needed. Although existing approaches for clustering with graph sampling show promising results, till now there was no effective and efficient approach proposed for inferring clustering assignments for all the nodes of the whole original graph – existing clustering inference methods either have poor inferring quality or are too time-consuming.

Our paper bridges this gap – we proposed a two-stage clustering inference algorithm to infer clustering affiliations for all the nodes of the original graph based on sampled clustered graph. The proposed *TCI* method takes advantages of the *label propagation* algorithm which makes the sample's label efficiently propagated to unlabelled nodes. Empirical results demonstrated that with the sample size of only 20% of the original graph, *TCI* algorithm in conjunction with any cluster-preserving sampling strategy is capable of inferring the clustering affiliation of the population commendably and performs better than the competing methods in most cases. Additionally, *TCI* has no parameters which makes it more stable in practice without putting more effort in tuning parameters.

REFERENCES

[1] Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. 2014. Network sampling: From static to streaming graphs. *TKDD* 8, 2 (2014), 7.

[2] Carlos Alzate and Johan AK Suykens. 2010. Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 2 (2010), 335–347.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 10 (2008), 008.

[4] Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The American Statistician* 49, 4 (1995), 327–335.

[5] Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659 (2016), 1–44.

[6] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12 (2002), 7821–7826.

[7] Christian Hübler, Hans-Peter Kriegel, Karsten Borgwardt, and Zoubin Ghahramani. 2008. Metropolis algorithms for representative subgraph sampling. In *ICDM*. IEEE, 283–292.

[8] Myunghwan Kim and Jure Leskovec. 2011. The network completion problem: Inferring missing nodes and edges in networks. In *ICDM*. SIAM, 47–58.

[9] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E* 78, 4 (2008), 046110.

[10] Rocco Langone, Carlos Alzate, and Johan AK Suykens. 2012. Kernel spectral clustering for community detection in complex networks. In *IJCNN*. IEEE, 1–8.

[11] Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *ICDM*. ACM, 631–636.

[12] László Lovász. 1993. Random walks on graphs. *Combinatorics, Paul erdos is eighty* 2 (1993), 1–46.

[13] Sofus A Macskassy and Foster Provost. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research* 8, May (2007), 935–983.

[14] Arun S Maiya. 2011. *Sampling and inference in complex networks*. University of Illinois at Chicago.

[15] Arun S Maiya and Tanya Y Berger-Wolf. 2010. Sampling community structure. In *WWW, 2010*. ACM, 701–710.

[16] Arun S Maiya and Tanya Y Berger-Wolf. 2011. Benefits of bias: Towards better characterization of network sampling. In *KDD*. ACM, 105–113.

[17] Raghvendra Mall, Rocco Langone, and Johan AK Suykens. 2013. FURS: Fast and Unique Representative Subset selection retaining large-scale community structure. *Social Network Analysis and Mining* 3, 4 (2013), 1075–1095.

[18] Raghvendra Mall, Rocco Langone, and Johan AK Suykens. 2013. Kernel spectral clustering for big data networks. *Entropy* 15, 5 (2013), 1567–1586.

[19] Andrew Mehler and Steven Skiena. 2009. Expanding network communities from representative examples. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3, 2 (2009), 7.

[20] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21, 6 (1953), 1087–1092.

[21] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* 74, 3 (2006), 036104.

[22] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 76, 3 (2007), 036106.

[23] Mostafa Salehi, Hamid R Rabiee, and Arezo Rajabi. 2012. Sampling from complex networks with high community structures. *Chaos: an Interdisciplinary Journal of Nonlinear Science* 22, 2 (2012), 023126.

[24] Yanhong Wu, Nan Cao, Daniel Archambault, Qiaomu Shen, Huamin Qu, and Weiwei Cui. 2017. Evaluation of graph sampling: A visualization perspective. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 401–410.

[25] Jaewon Yang and Jure Leskovec. 2012. Community-affiliation graph model for overlapping network community detection. In *ICDM*. IEEE, 1170–1175.

[26] Jaewon Yang and Jure Leskovec. 2014. Structure and overlaps of ground-truth communities in networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 2 (2014), 26.

[27] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[28] Jianpeng Zhang. 2018. *On Graph Sample Clustering*. Eindhoven University of Technology (TU/e) [https://drive.google.com/open?id=1D0twRFVSkUDBnWtTKsDs3HJwDGo1-OaE].

[29] Jianpeng Zhang, Yulong Pei, George HL Fletcher, and Mykola Pechenizkiy. 2016. Structural measures of clustering quality on graph samples. In *ASONAM, 2016*. IEEE, 345–348.

[30] Jianpeng Zhang, Kaijie Zhu, Yulong Pei, George Fletcher, and Mykola Pechenizkiy. 2017. Clustering-structure representative sampling from graph streams. In *Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017*. Springer, 265–277.

[6] The open source code and parameter specification of NetKit can be found at http://netkit-srl.sourceforge.net.