Estimation of Graph Features Based on Random Walks Using Neighbors' Properties

Tsuyoshi Hasegawa[1,2(✉)], Shiori Hironaka[1,3[0000−0001−7994−2858]], and Kazuyuki Shudo[1[0000−0002−3939−9800]]

[1] Kyoto University, Yoshida-nihonmatsu-cho, Sakyo-ku, Kyoto 606-8501 Japan
[2] hasegawa.tsuyoshi.24r@st.kyoto-u.ac.jp
[3] hironaka@media.kyoto-u.ac.jp

Abstract. Using random walks for sampling has proven advantageous in assessing the characteristics of large and unknown social networks. Several algorithms based on random walks have been introduced in recent years. In the practical application of social network sampling, there is a recurrent reliance on an application programming interface (API) for obtaining adjacent nodes. However, owing to constraints related to query frequency and associated API expenses, it is preferable to minimize API calls during the feature estimation process. In this study, considering the acquisition of neighboring nodes as a cost factor, we introduce a feature estimation algorithm that outperforms existing algorithms in terms of accuracy. Through experiments that simulate sampling on known graphs, we demonstrate the superior accuracy of our proposed algorithm when compared to existing alternatives.

Keywords: Social Network · Random Walk · Graph Sampling.

1 Introduction

Examining the graph structure of nodes and edges in online social networks (OSNs) is a significant challenge, prompting active research efforts to address this issue [4, 21, 7, 15]. However, data access in conventional OSNs, like X[4], is restricted [1–3], rendering it nearly impossible to acquire and analyze the complete graph. Therefore, a pragmatic strategy involves estimating the graph's features by sampling a representative portion of the OSNs.

To estimate OSNs' features through sampling, leveraging random walks proves advantageous. Several random walk algorithms have been introduced for unbiased feature estimation [9, 18, 25, 5, 13, 19, 22, 20]. Many OSNs offer application programming interfaces (APIs) that provide access to information about a user's follower or followee lists, specifically details about adjacent nodes [1–3]. By iteratively selecting a node at random from the adjacent nodes obtained through the API and transitioning, random walk sampling on OSNs becomes feasible. Exploiting the inherent Markov property of random walks enables the computation

[4] https://twitter.com

Proc. WISE 2024 (LNCS 15437), pp.456-466, December 2024

of suitable weights for the obtained sample sequence, enabling the derivation of unbiased estimates for OSNs [18, 7, 16]. Uniform independent sampling based on node IDs is generally challenging owing to the unknown distribution of node IDs [6]. Additionally, traversal methods like breadth-first sampling [14] cannot provide unbiased features owing to unknown biases in the acquired sample sequence.

APIs from common OSNs restrict the number of queries allowed per unit of time. Moreover, certain OSNs, such as X, have introduced charges for API usage. Therefore, estimating OSN features with minimal API calls is crucial, considering time and cost factors. Iwasaki et al. [12] treated the API call count as a cost and compared it with existing random walk-based feature estimation algorithms. In our approach, we focus that many APIs allow obtaining both the list of adjacent nodes and the degree of those nodes simultaneously. We leverage this information to develop a more efficient algorithm. In the proposed algorithm, not confined to degree estimation, we can also estimate unbiased OSN features for any features obtained simultaneously when acquiring the list of adjacent nodes.

In this study, we propose an algorithm for estimating features in OSNs using random walks and properties of adjacent nodes. Through simulation experiments, we demonstrate that our proposed algorithm attains the highest accuracy in estimating OSN features compared to existing methods. Our proposed method leverages the properties of adjacent nodes, which are obtained along with the adjacent node acquisition API, for unbiased feature estimation.

2 Preliminaries

2.1 Definitions and Notations

In this study, we use the notation of a directed graph $G = (V, E)$ to represent the social graph. $V = \{v_1, v_2, ..., v_n\}$ represents the set of nodes (users), with n being the total number of nodes in the graph ($n = |V|$). E is the set of directed edges, depicting the following relationships. For every edge (v_i, v_j), we introduce a set of edges and reverse edges by adding (v_j, v_i), denoted as E'. When a directed edge (v_i, v_j) exists, we refer to node v_j as the friend of node v_i, and node v_i as the follower of node v_j. For a node $v_i \in V$, we define the set of friends as $N_{\text{out}}(v_i) = \{v_j \in V : (v_i, v_j)\}$ and the set of followers as $N_{\text{in}}(v_i) = \{v_j \in V : (v_j, v_i)\}$. Additionally, $N(v_i) = N_{\text{out}}(v_i) \cup N_{\text{in}}(v_i)$. We also define the out-degree and in-degree of each node as $d_{\text{out}}(v_i) = |N_{\text{out}}(v_i)|$ and $d_{\text{in}}(v_i) = |N_{\text{in}}(v_i)|$, respectively. Moreover, we introduce the total degree as $d_{\text{sum}}(v_i) = d_{\text{in}}(v_i) + d_{\text{out}}(v_i)$, and the mutual connections between followers and friends as $d_{\text{in-out}}(v_i) = |N_{\text{out}}(v_i) \cap N_{\text{in}}(v_i)|$.

We define the property of node v_i as $a(v_i)$. Examples of the property $a(v_i)$ include the degree of v_i, the number of posts, and binary labels such as bot labels.

Proc. WISE 2024 (LNCS 15437), pp.456-466, December 2024

2.2 Model

In this study, we focus on the APIs which enable acquiring the degree information (number of friends, number of followers) and properties to be estimated for each adjacent node when querying information about them. To clarify, when querying the list $N(v_i)$ of adjacent nodes for node v_i, we assume that the out-degree $d_{\text{out}}(v_j)$, in-degree $d_{\text{in}}(v_j)$ and property $a(v_j)$ of any node v_j within $N(v_i)$ can be obtained simultaneously. In real OSNs, X and Mastodon offer APIs that adhere to this model [3, 2].

We treats the frequency of acquiring adjacent nodes as a cost. We assume that a single instance of adjacent node acquisition allows for the simultaneous retrieval of $N_{\text{in}}(v_i)$ and $N_{\text{out}}(v_i)$. Regardless of the number of adjacent nodes, we assume that all adjacent nodes can be obtained at a fixed cost of 1. Many studies on sampling methods involving random walks have traditionally assessed accuracy based on the size of the sample sequence [7, 16, 9]. In contrast, our study adopts a different perspective by considering the acquisition of a list of neighboring nodes as a cost. This approach, aligned with the methodology of Iwasaki et al. [12], stems from the realization that obtaining the list of neighboring nodes can pose a practical bottleneck in OSN sampling.

We assume that the graph G is weakly connected and remains static during the random walk. Additionally, upon transitioning to a node, we store information like its degree and properties. This includes maintaining a list of adjacent nodes along with their respective degrees and properties.

3 Proposed Method

In this section, we present a sampling algorithm based on random walks that utilizes the properties of each acquired adjacent node during the process of obtaining adjacent nodes. We discuss the Markov chain aspect within our proposed method to elucidate the algorithms employed for estimating these features.

The features on the OSN, which our proposed method can estimate are derived from properties obtained concurrently with the acquisition of adjacent nodes. In specific terms, when retrieving the list of adjacent nodes for node v_i, if we can get the property $a(v_j)$ for any node v_j in the adjacent node list $N(v_i)$, our proposed method can estimate average and distribution of the property.

3.1 Probabilistic Addition of Adjacent Nodes to the Sample Sequence

We present an algorithm of our proposed method in Algorithm 1. The fundamental transition method in our algorithm closely resembles a standard random walk. However, after transitioning to node v_i and its addition to the sample sequence, a distinctive element is introduced. With a probability α, instead of the typical transition, we incorporate a randomly chosen node from the acquired adjacent nodes into the sample sequence. Here, α is a parameter in the range

Algorithm 1 Proposed sampling algorithm

Input: n_0 : initial node, α : parameter($0 \leq \alpha < 1$), b : number of queries
Output: *sampling_node_list* : sample sequence

$v_i \leftarrow n_0$
sampling_node_list $\leftarrow \{v_i\}$
staying_node_list $\leftarrow \{v_i\}$
query_count $\leftarrow 1$
while *query_count* $< b$ **do**
  $N(v_i) \leftarrow$ A list obtained by concatenating $N_{\text{out}}(v_i)$ and $N_{\text{in}}(v_i)$
  $p \leftarrow$ Random number generated between 0 and 1
  while $p < \alpha$ **do**
    $v_j \leftarrow$ A node uniformly selected at random from *neighbors*
    Append v_j to *sampling_node_list*
    $p \leftarrow$ Random number generated between 0 and 1
  end while
  $v_i \leftarrow$ A node uniformly selected at random from *neighbors*
  if $v_i \notin$ *staying_node_list* **then**
    query_count $\leftarrow$ *query_count* $+ 1$
  end if
  Append v_i to *sampling_node_list*
  Append v_i to *staying_node_list*
end while
return *sampling_node_list*

$0 \leq \alpha < 1$, and the transition from node v_i to node v_j occurs with a probability of $1 - \alpha$. In this paper, we refer to the node reached after the transition as the *staying node*, the node added to the sample sequence as the *sampling node*, the operation executed with a probability α as *adjacent node sampling*, and the operation performed with a probability $1 - \alpha$ as *transition sampling*. In adjacent node sampling, the staying node remains unchanged, and the sampling node is randomly selected from the adjacent nodes of the staying node. In transition sampling, the staying node is updated, and the sampling node becomes identical to the staying node.

In this study, we treat OSNs as directed graphs, allowing transitions and sampling to occur even on reverse edges. This occurs because, in real OSNs when transitioning between users, it is possible to choose users who transition from both followers and friends. Consequently, the list of adjacent nodes for a node v_i is a combination of its in-neighbors $N_{\text{in}}(v_i)$ and out-neighbors $N_{\text{out}}(v_i)$. In this context, the list of adjacent nodes for node v_i is an array that allows duplicates. Specifically, for a node v_j, if $v_j \in N_{\text{out}}(v_i) \cap N_{\text{in}}(v_i)$, then node v_i has two occurrences of node v_j in its list of adjacent nodes. Because the node to transition or sample is uniformly and randomly chosen from the list of adjacent

nodes, a node appearing twice in the list has double the probability of being chosen compared to a node that appears only once.

In our study, sampling adjacent nodes is a cost-free process because we define the acquisition frequency of adjacent nodes as the cost. This operation randomly adds nodes to the sample sequence from the existing list of adjacent nodes, incurring no new expenses. In contrast, transition sampling involves obtaining the adjacent nodes of the transition destination node, incurring a cost. However, we assume that the information about once-acquired adjacent nodes is stored (Section 2.2). Consequently, if a node previously sampled through transition sampling is sampled again, no additional cost is incurred. Therefore, the frequency of acquiring adjacent nodes corresponds to the count of new transition sampling events.

We use the term *query count* to represent the number of times the list of adjacent nodes is obtained through the new transition sampling. The proposed sampling algorithm stops when the number of acquired adjacent nodes reaches a specified query limit b. This limit can be established based on the OSN's API specifications, ensuring a reasonable number of queries within a given time frame. Notably, a higher query limit b generally correlates with improved accuracy in estimation accuracy. The relationship between the query limit and estimation accuracy are elucidated in Section 4.

3.2 Feature Estimation

We introduce an algorithm for feature estimation in the context of the proposed sampling algorithm. In this section, we present the algorithm for estimating features using in the proposed sampling method. The primary objective is to apply weighting to the sample sequence generated by the proposed transition sampling algorithm, creating a process for estimating the expected value of features on the OSN. The proof of this process can be found in [10].

Let f be any function $f : V \to \mathbb{R}$, and consider the uniform distribution $\mathbf{u} \stackrel{\text{def}}{=} [u(1), u(2), ..., u(n)] = [1/n, 1/n, ..., 1/n]$. The expected value $\mathbb{E}_u(f)$ of the feature to be estimated on the OSN is given as follows.

Definition 1. $\mathbb{E}_u(f) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{v \in V} f(v)$

To obtain the expected value of the desired feature on the OSN, this can be accomplished by appropriately defining the function f. For instance, if you wish to estimate the out-degree distribution $\mathbb{P}\{D_G = d\}, (d = 1, 2, ..., n-1)$ of a graph G, selecting the function $f(v) = \mathbb{1}_{d_{\text{out}}(v)=d}$ would be suitable.

To define a re-weighting process for obtaining $\mathbb{E}_u(f)$ from the sample sequence $\{Z_s\}_{s=1}^t$ generated by the proposed sampling algorithm, a new function $g : \Omega \to \mathbb{R}$ is introduced for the function f as follows.

Definition 2. $g(e_{ij}) \stackrel{\text{def}}{=} f(v_j)$

The function g applies the function f to the sampling nodes in the sample Z_t.

Next, we define a weighting function as follows.

Table 1. Dataset Overview

Network	Type	Nodes	Edges
ego-Twitter	Social Network	81,306	1,768,149
soc-Slashdot	Social Network	82,168	948,464
Amazon	Product Network	262,111	1,234,877
DBA model	Generation Network	100,000	1,000,000

Definition 3. $w(e_{ij}) \stackrel{\text{def}}{=} \frac{1}{d_{\text{sum}}(v_j)}$

In this study, we assume that the number of friends $d_{\text{out}}(v_j)$ and followers $d_{\text{in}}(v_j)$ of sampling nodes are accessible. Consequently, we can obtain $d_{\text{sum}}(v_j)$ for the sampling node v_j.

Here, the following theorem holds.

Theorem 1. *For the sample sequence* $\{Z_s\}_{s=1}^t$ *obtained from the proposed transition algorithm, as* $t \to \infty$*,*

$$\frac{\sum_{s=1}^t w(Z_s)g(Z_s)}{\sum_{s=1}^t w(Z_s)} \to \mathbb{E}_u(f) \ \ a.s.$$

Proof. Refer to [10].

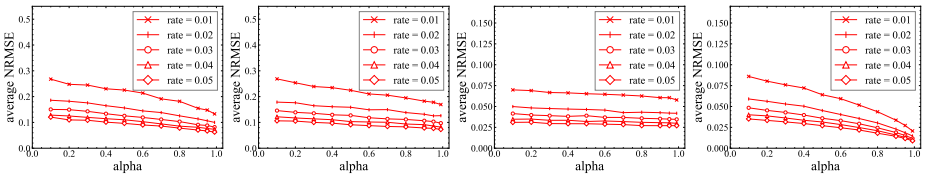
As stated in Theorem 1, for the sample sequence $\{Z_s\}_{s=1}^t$ obtained by the proposed sampling algorithm, the estimator $\sum_{s=1}^t w(Z_s)g(Z_s)/\sum_{s=1}^t w(Z_s)$ converges to the expected value of the desired feature on the OSN. Though the sample Z_t contains information about both sampling node X'_t and the staying node X_t, retaining information solely about sampling node X'_t is sufficient for feature estimation. Therefore, in Algorithm 1, information about the staying node X_t is not returned.

The proposed method can estimate the unbiased features regarding any properties that can be obtained when acquiring adjacent nodes. This is because the estimable feature $g(e_{ij})$ for the sample e_{ij} added through adjacent node sampling is derived from the property $a(v_j)$ obtained at the same time during the acquisition of adjacent nodes. The specific content of this properties varies based on the OSN's API specifications. For instance, in X, which provides degree information [3], one can estimate the average degree and degree distribution. Similarly, in Mastodon, which provides bot rates and post counts [2], these parameters can also be estimated.

Thus, we have successfully developed a weighting process to estimate the expected value of features on the OSN for the sample sequence acquired through the proposed sampling algorithm.

4 Experiment

We assess the accuracy of the proposed method across various networks. In real-world OSN sampling, the target graph is often unknown. However, for these



(a) ego-Twitter (b) Slashdot (c) Amazon (d) DBA model

Fig. 1. Average NRMSE for each feature categorized by query rate at each α.

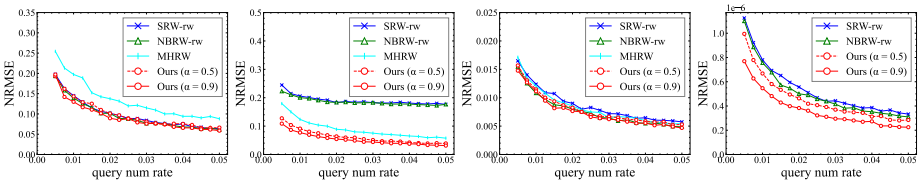
experiments, we conduct sampling simulations on known graphs to facilitate evaluation.

4.1 Experimental Setup

Dataset: Our experiments utilized three datasets from the Stanford Large Network Dataset Collection [17], and we also employed the Directed-Barabasi-Albert model (DBA model) [23], a generative model for complex networks. The target graphs are directed, and we focus on the maximum weakly connected component. The DBA model extends the Barabasi-Albert model [23] to directed graphs.
Simulation: For the proposed sampling algorithm, the initial nodes are randomly selected from the graph, this process is independent for each experiment. All sampling simulations are independently conducted 1000 times. Query count b is chosen by the proportion of the total number of nodes in the graph.
Evaluation metrics: The evaluation metric for each feature estimation is the Normalized Root Mean Square Error (NRMSE). NRMSE is widely employed in related studies to assess the accuracy of estimated values [16, 9, 13, 12]; lower values indicate superior performance. It is computed as follows. NRMSE $= \frac{1}{x}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x-\hat{x_i})^2}$,where x represents the true value of the feature, $\hat{x_i}$ is the estimated value of the feature in the i-th sampling simulation, and N is the number of simulations.
Features: The features under investigation in our experiments include average out-degree and label rate. labels are binary synthetically introduced to the dataset, simulating scenarios akin to bot labels in real OSNs. Labels are assigned randomly and repetitively assigned to selected nodes based on the corresponding probability until the labeled node proportion reaches 10% of the total. The introduction of labeled nodes enables simulations to estimate the proportion of nodes with certain characteristics.

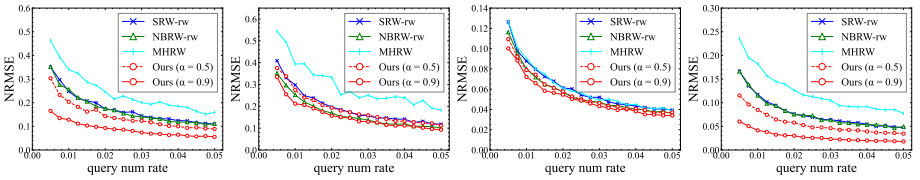
4.2 Relationship between α and Estimation Accuracy

In the proposed method, we conducted experiments to investigate the relationship between the probability α of performing neighboring node sampling and



(a) ego-Twitter (b) Slashdot (c) Amazon (d) DBA model

Fig. 2. NRMSE for out-degree estimation.



(a) ego-Twitter (b) Slashdot (c) Amazon (d) DBA model

Fig. 3. NRMSE for random label estimation.

estimation accuracy. Across each dataset, we varied α from 0.1 to 0.9 in increments of 0.1. Additionally, we tested α values of 0.95 and 0.99. The query count was chosen 0.1% to 0.5% of the total number of nodes in the graph, adjusting in increments of 0.1%. Figure 1 illustrates the average NRMSE for each feature discussed in Section 4.1. Results are presented for query counts representing $1\%, 2\%, 3\%, 4\%$, and 5% of all node counts. Figure 1 shows that as α approaches 1, the average NRMSE decreases across all graphs and query numbers. Therefore, the parameter α should be set to the largest possible value that is still less than 1.

In practice, α should be determined by considering both the computational complexity and the API rate limit. The value of α affects the number of adjacent node samplings, which on average is $1/(1-\alpha)$ per API call (lines 8 of Algorithm 1). The sampling operation itself is efficient, taking $O(1)$ time (line 9-11 of Algorithm 1). Therefore, the average computational complexity of a adjacent node sampling for each API call is $O(1/(1-\alpha))$. As α gets closer to 1, the complexity of sampling increases dramatically. Thus, α should be set based on the API's rate limit. For instance, if Mastodon's API allows one call per second [2], α should be chosen to be as large as possible while ensuring that the sampling process (line 6-18 of Algorithm 1) finishes within one second, depending on the system's computational power.

4.3 Comparison with Existing Methods

We compared the estimation accuracy of each feature between the proposed and existing methods. Existing methods for comparison include well-known random walk-based feature estimation techniques: Simple Random Walk with Reweighting (SRW-rw) [7, 24], Non-backtracking Random Walk with Reweighting (NBRW-rw) [16], and Metropolis-Hastings Random Walk (MHRW) [7, 24]. These existing methods align with those compared by Iwasaki et al. [12] under the same cost setting. Iwasaki et al. noted a tendency for NBRW-rw to achieve better accuracy when considering the acquisition frequency of adjacent nodes as the cost. However, they also highlighted the possibility of accuracy reversal based on the specific graph and features.

Figures 2 and 3 illustrate a comparison of the NRMSE values for each feature using the proposed method with $\alpha = 0.5, 0.9$ and existing methods. The horizontal axis represents the query count as a ratio to the total number of nodes. We vary the query count ratio from 0.25% to 5%, adjusting in increments of 0.25%. MHRW has been omitted owing to significant deviations, particularly in the average out-degree estimation of the DBA model. Across all graphs and features, the proposed method with $\alpha = 0.9$ consistently matches or outperforms existing methods. In the proposed method, we achieved higher accuracy at $\alpha = 0.9$ compared to at $\alpha = 0.5$, but even at $\alpha = 0.5$, it surpasses the existing methods for many graphs and features. Additionally, the results indicate that higher query counts b lead to improved estimation accuracy.

5 Related Work

We discuss relevant research on graph sampling. Gjoka et al. [7] compared RW-rw and MHRW and illustrated that SRW-rw achieved superior accuracy by reweighting from the steady-state distribution of random walks to obtain unbiased estimates. Lee et al. [16] introduced NBRW, a non-backtracking random walk that avoids revisiting the previous node. Through theoretical and experimental investigations, they established that NBRW-rw provided unbiased estimates and outperformed SRW-rw. Iwasaki et al. [12] proposed a method for comparing sampling algorithms based on query count and demonstrated the potential for accuracy evaluations of SRW-rw and NBRW-rw to reverse depending on the feature.

Next, we explore related research focusing on sampling techniques involving information from neighboring nodes. Han et al. [8] introduced a method in which, during random walks, adjacent nodes are sampled based on the motif under estimation. The decision to acquire information from adjacent nodes depends on the motif, resulting in variations in the probability and depth of obtaining such information, distinguishing it from our study. Additionally, they consider the sampled node count as a cost, which differs from our study using API query count as the cost and estimating various features from adjacent nodes. Illenberger et al. [11] proposed a technique for estimating features by adjusting the sample sequence of snowball sampling and collecting information from adjacent nodes

Proc. WISE 2024 (LNCS 15437), pp.456-466, December 2024

of accessed nodes. However, their approach assumes knowledge of the overall node count, distinguishing it from our study, which employs random walks to estimate features in unknown OSNs.

6 Conclusion

In this study, we introduced a random walk that stochastically utilized information from adjacent nodes, considering the number of queries needed to obtain this information on OSNs as the cost. Through experiments, we demonstrated that our proposed method provided more accurate estimations for average degree, the proportion of randomly assigned binary labels, the proportion of labels biased toward high-degree nodes, and the proportion of labels biased toward low-degree nodes compared to existing methods. Furthermore, experiments conducted across various datasets have revealed that accuracy improves as the parameter α, representing the probability of adjacent node sampling, approaches 1.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Number JP21H04872.

References

1. Facebook API documents. https://developers.facebook.com/docs/graph-api, (Accessed on 19/11/2023)
2. Mastodon API documents. https://docs.joinmastodon.org/api, (Accessed on 19/11/2023)
3. Twitter API documents. https://developer.twitter.com/en/docs/twitter-api, (Accessed on 19/11/2023)
4. Ahn, Y.Y., Han, S., Kwak, H., Moon, S., Jeong, H.: Analysis of topological characteristics of huge online social networking services. In: Proceedings of the 16th International Conference on World Wide Web. pp. 835–844 (2007)
5. Chen, X., Li, Y., Wang, P., Lui, J.C.S.: A general framework for estimating graphlet statistics via random walk. Proceedings of the VLDB Endowment **10**(3), 253–264 (2016)
6. Chiericetti, F., Dasgupta, A., Kumar, R., Lattanzi, S., Sarlós, T.: On sampling nodes in a network. In: Proceedings of the 25th International Conference on World Wide Web. pp. 471–481 (2016)
7. Gjoka, M., Kurant, M., Butts, C.T., Markopoulou, A.: Practical recommendations on crawling online social networks. IEEE Journal on Selected Areas in Communications **29**(9), 1872–1892 (2011)
8. Han, G., Sethu, H.: Waddling random walk: Fast and accurate mining of motif statistics in large graphs. In: Proceedings of the 2016 IEEE 16th International Conference on Data Mining. pp. 181–190 (2016)
9. Hardiman, S.J., Katzir, L.: Estimating clustering coefficients and size of social networks via random walk. In: Proceedings of the 22nd International Conference on World Wide Web. pp. 539–550 (2013)

10. Hasegawa, T., Hironaka, S., Shudo, K.: Estimation of graph features based on random walks using neighbors' properties (2024), https://arxiv.org/abs/2409.08599
11. Illenberger, J., Flötteröd, G.: Estimating network properties from snowball sampled data. Social Networks **34**(4), 701–711 (2012)
12. Iwasaki, K., Shudo, K.: Comparing graph sampling methods based on the number of queries. In: Proceedings of the 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications. pp. 1136–1143 (2018)
13. Iwasaki, K., Shudo, K.: Estimating the clustering coefficient of a social network by a non-backtracking random walk. In: Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing. pp. 114–118 (2018)
14. Kurant, M., Markopoulou, A., Thiran, P.: Towards unbiased bfs sampling. IEEE Journal on Selected Areas in Communications **29**(9), 1799–1809 (2011)
15. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web. pp. 591–600 (2010)
16. Lee, C.H., Xu, X., Eun, D.Y.: Beyond random walk and metropolis-hastings samplers: Why you should not backtrack for unbiased graph sampling. In: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems. pp. 319–330 (2012)
17. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (2014)
18. Li, R.H., Yu, J.X., Qin, L., Mao, R., Jin, T.: On random walk based graph sampling. In: Proceedings of the 2015 IEEE 31st International Conference on Data Engineering. pp. 927–938 (2015)
19. Matsumura, T., Iwasaki, K., Shudo, K.: Average Path Length Estimation of Social Networks by Random Walk. In: Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing. pp. 611–614 (2018)
20. Matsumura, T., Shudo, K.: Metropolis-hastings random walk with a reduced number of self-loops. In: Proceedings of the 2019 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking. pp. 1556–1563 (2019)
21. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement. pp. 29–42 (2007)
22. Nakajima, K., Iwasaki, K., Matsumura, T., Shudo, K.: Estimating top-k betweenness centrality nodes in online social networks. In: Proceedings of the 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications. pp. 1128–1135 (2018)
23. Pósfai, M., Barabasi, A.L.: Network Science. Cambridge University Press (2016)
24. Rasti, A.H., Torkjazi, M., Rejaie, R., Duffield, N., Willinger, W., Stutzbach, D.: Respondent-driven sampling for characterizing unstructured overlays. In: Proceedings of the 28th International Conference on Computer Communications. pp. 2701–2705 (2009)
25. Ribeiro, B., Towsley, D.: On the estimation accuracy of degree distributions from graph sampling. In: Proceedings of the 51st IEEE Conference on Decision and Control. pp. 5240–5247 (2012)

Proc. WISE 2024 (LNCS 15437), pp.456-466, December 2024