Common Neighbors Matter: Fast Random Walk Sampling With Common Neighbor Awareness

Rui Wang, Yongkun Li, *Member, IEEE*, Shuai Lin, WeiJie Wu, Hong Xie, *Member, IEEE*, Yinlong Xu, and John C.S. Lui, *Fellow, IEEE*

Abstract—Random walk is widely applied to sample large-scale graphs due to its simplicity of implementation and solid theoretical foundations of bias analysis. However, its computational efficiency is heavily limited by the *slow convergence* rate (a.k.a. long burn-in period). To address this issue, we propose a common neighbor aware random walk framework called CNARW, which leverages weighted walking by differentiating the next-hop candidate nodes to speed up the convergence. Specifically, CNARW takes into consideration the common neighbors between previously visited nodes and next-hop candidate nodes in each walking step. Based on CNARW, we further develop two efficient "unbiased sampling" schemes, and we also design two variant algorithms which can reduce sampling cost and speed up the convergence. Experimental results on real-world network datasets show that our approach converges remarkably faster than the state-of-the-art random walk sampling algorithms; and to achieve the same estimation accuracy, our approach reduces the query cost significantly. Last, we use two case studies to demonstrate the effectiveness of our sampling framework in solving large-scale graph analysis tasks.

Index Terms—Random walk, online social networks, graph sampling

1 Introduction

In recent years, online social networks (OSNs) such as Facebook,Twitter and Flickr have become more and more popular, so how to take advantage of these platforms to promote commercial businesses, like viral marketing, product recommendation and advertisement promotion, has gained significant attention. This task necessitates an accurate estimation or even mining of various kinds of graph centralities. The rationale is that different kinds of graph centralities imply different attributes of users, which can be effectively used for promoting commercial businesses.

We use two application examples to further justify the usefulness of estimating graph centralities. The first example is *investment on networking platforms.* It is well known that OSNs are proven to be effective for viral marketing due to the "word-of-mouth" effect [7], [14], [30]. That is, a user who bought a product may influence her friends (neighboring nodes in OSNs) to purchase the same product. Clearly, different OSN platforms may have different potentials to do viral marketing, as both the activeness of users and the influence between users may differ significantly across different OSNs. *Therefore, one interesting problem for a product owner is: Which OSN platform should be targeted to do viral marketing so as to attract as many buyers as possible with a given advertisement budget?* This problem may be heuristically solved by estimating the average similarity of all user pairs in different OSNs, because higher similarity may imply easier influence. Another example is *bundling in viral marketing*. Note that bundling sales which bundle multiple products together to sell with some discount can be witnessed everywhere in our daily life, and it is also widely studied in network economics. In the situation of viral marketing in OSNs, we can also expect that bundling can be used to promote the sale. However, its efficiency may depend on which products to bundle, e.g., bundling two products which target young and elderly people respectively may even reduce the sale. *Thus, one interesting problem is: Which products should be bundled together so as to trigger a larger sale?* This problem can be better solved by mining the value of the OSN, e.g., we can estimate the interest distribution of users on every product, and then bundle the set of products which have similar distributions, as similar distributions may imply that users have similar interest in the set of products.

However, it is not easy to accurately estimate graph centralities or efficiently solve graph mining problems. First, OSNs can be extremely large. Second, many OSNs may only allow third-party agents to access the networking data through fixed API with rate constraints to protect user

- *Rui Wang, Shuai Lin, and Yinlong Xu are with the University of Science and Technology of China, Hefei, Anhui 230026, China. E-mail: {rwang067, shuailin}@mail.ustc.edu.cn, ylxu@ustc.edu.cn.*
- *Yongkun Li is with the University of Science and Technology of China, Hefei, Anhui 230026, China, and also with USTC Tang Scholar, Hefei, Anhui 230052, China. E-mail: ykli@ustc.edu.cn.*
- *WeiJie Wu is with Independent Researcher, California, CA 90603 USA. E-mail: wuwjpku@gmail.com.*
- *Hong Xie is with Chongqing University, Chongqing 400044, China. E-mail: hongx87@gmail.com.*
- *John C.S. Lui is with the Chinese University of Hong Kong, Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.*

Manuscript received 4 January 2021; revised 12 October 2021; accepted 29 January 2022. Date of publication 11 February 2022; date of current version 3 April 2023.
The work of Yongkun Li was supported in part by the National Key R&D Program of China under Grant 2018YFB1003204, in part by the Youth Innovation Promotion Association CAS, and in part by the USTC Research Funds of the Double First-Class Initiative under Grant YD2150002003. The work of John C.S. Lui was supported in part by GRF under Grant 14208816.
(Corresponding author: Yongkun Li.)
Recommended for acceptance by P. Cui.
Digital Object Identifier no. 10.1109/TKDE.2022.3150427

privacy. These challenges raise a fundamental question: *How to design computationally efficient algorithms for large-scale OSNs?* Graph sampling is a promising paradigm since it generates representative samples of the OSNs without traversing the whole network and has received extensive attentions [4], [16], [31], [32], [33].

Among various sampling approaches, random walk based method is the mainstream due to its scalability and simplicity of implementation. Here is the general idea. A walker starts at an arbitrary node, then repeatedly jumps to another node by choosing from the current node's neighbors uniformly at random. After many steps, the probability of a node being visited tends to reach a stationary probability distribution, and one can start collecting samples (i.e., sample nodes or sample edges), after convergence [9], [25], [36]. The time duration to reach the stationary distribution is known as the burn-in period [24], [25]. Based on the collected samples and the knowledge of the stationary probability distribution, one can generate unbiased estimations for interested graph measures.

However, random walk based sampling algorithms suffer from the long burn-in period issue in real-world OSNs [24], [36], i.e., they may cost lots of steps to reach the convergence status. This leads to a high cost, especially for the network sampling situation, in which it needs to query from the network at each step if a new node is accessed. Consequently, given a sampling budget (e.g., a limited number of nodes can be queried from the network), we may be able to collect only a small number of representative samples, which would severely reduce the accuracy of graph mining tasks. Thus, one important question is: *How to speed up the convergence of random walk over large-scale graphs?*

Currently, there are two classes of methods to accelerate the convergence of random walk. The first one aims to increase the conductance of graphs [23], [26]. These schemes often need the global information of the graph, which is usually infeasible in practice. The second class modifies the transition probabilities in each walking step [18], [35]. These schemes usually utilize the walking history and require only partial information of the graph. The key issues are what kind of partial information is needed and how to utilize the information to speed up the convergence of random walk.

In this paper, we propose a new random walk framework CNARW in which the walker optimizes the next-hop node selection by looking back previously visited nodes and also looking one step ahead with a small overhead. Our intuition behind the design is simply " *common neighbors matter.*" In particular, CNARW takes into consideration the number of common neighbors between the current node and the next-hop candidates so that it can speed up the convergence significantly. We also study another fundamental question: *How many steps should the walker look back?* Intuitively, the larger the number of steps to look back, the more historical information the walk can have, which will lead to faster convergence speed. However, this also leads to a larger computational cost. Our contributions are:

- We propose CNARW, a common neighbor aware random walk, which selects the next node to visit by taking into consideration the number of common neighbors between the currently visiting node and its neighbors. CNARW shrinks the burn-in period and speeds up the convergence of random walks.
- We also develop efficient node and edge sampling algorithms based on CNARW, and develop an efficient scheme to provide "*unbiased statistical estimation*". We also provide theoretical proofs to guarantee the unbiasedness of graph measure estimation.
- We design two variant algorithms of CNARW to further improve the performance. The first one incorporates the idea from non-backtracking random walk, which can reduce the sampling cost. The second one utilizes more visited nodes which further speeds up the convergence.
- We conduct extensive experiments on real-world datasets to evaluate the efficiency of CNARW. Results show that with CNARW the number of steps needed to converge can be reduced by up to 71.9% compared to existing schemes like SRW [22], NBRW [18] and CNRW [35]. Furthermore, to achieve the same estimation accuracy, CNARW can also reduce the query cost by up to 35.7%.

2 Preliminaries

2.1 Random Walk on Graphs

We consider undirected and connected graphs which are denoted by $G(V, E)$, where V is the set of nodes and E is the set of edges. We use $|V|$ and $|E|$ to denote the numbers of nodes and edges in G, respectively. We denote $N(v)$ for $v \in V$ as the set of neighbors of v and $deg(v)$ as the degree of v, i.e., $deg(v) = |N(v)|$.

A random walk on graph $G(V, E)$ can essentially be viewed as a finite Markov chain, in which the walker starts from a given node, say $v_0 \in V$, then randomly chooses a neighbor of v_0 and jumps to it according to some transition probability distribution defined by the random walk algorithm. The walker continues this process by repeating the above step. The transition probability distribution in one step can be represented as a $|V| \times |V|$ matrix $\mathbf{P} = (P_{uv})$, $u, v \in V$, where P_{uv} denotes the probability of moving from u to v in one step. For different algorithms, they can be mathematically represented by their transition probability matrices. Here, we introduce *simple random walk (SRW)*, which is classical and widely used as the baseline of various optimized random walks.

Simple Random Walk (SRW) [22]. Suppose that the walker is currently at node u, SRW chooses the next node v from $N(u)$ uniformly at random according to $deg(u)$, i.e., P_{uv} is $1/deg(u)$ if $v \in N(u)$ or 0 otherwise. For SRW, the stationary distribution $\boldsymbol{\pi} = \{\pi(u)\}_{u \in V}$, where $\pi(u)$ denotes the probability of node u being visited when the random walk converges, can be derived as $\pi(u) = \frac{deg(u)}{2|E|}$.

2.2 Unbiased Graph Sampling

Unbiased graph sampling aims for collecting node/edge samples with the uniform distribution, i.e., each node/edge is sampled with an equal probability, while random walks may not produce uniform samples as they visit nodes/edges with the probability of stationary distribution π after

convergence. Therefore, to realize unbiased graph sampling via random walks, a bias correction method is in need. And the whole process can be divided into two steps: (1) collect enough number of samples, and (2) perform an unbiased estimation.

In the first step, there are two ways of collecting samples: continuous sampling [9], [18], [21] and independent sampling [25], [36]. Continuous sampling initiates one walk only and keeps walking after convergence until collecting enough samples, while independent sampling initiates many random walks and collects only one sample from each walk after convergence. Note that samples can only be collected after convergence for both approaches so as to provide predictable or unbiased estimations. Thus, reducing the burn-in period is crucial to reduce the computation cost in random walk sampling, no matter which approach is used to collect samples.

The second step of unbiased graph sampling is to perform estimation on collected samples. Suppose that the graph measure to be analyzed is defined by a function $f : V \to R$, then applying f on enough number of samples for a random walk with stationary distribution $\boldsymbol{\pi}$ produces an estimation of $E_{\boldsymbol{\pi}}[f] \triangleq \sum_{u \in V} f(u)\pi(u)$. The accuracy of this estimation is guaranteed by *the Strong Law of Large Numbers* (SLLN) [13], [18], which can be stated as follows.

Theorem 1. Strong Law of Large Numbers (SLLN). *Suppose that $\{X_t\}_{t \geq 0}$ is a finite, irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$, where X_t denotes the state of the Markov chain at time t. As $t \to \infty$, we have*

$$\mu_t(f) \to E_{\boldsymbol{\pi}}[f], \quad \text{almost surely (a.s.)},$$

for any function $f : V \to R$ with $E_{\boldsymbol{\pi}}[|f|] < \infty$, where

$$\mu_t(f) \triangleq \frac{1}{t}\sum_{s=0}^{t} f(X_s), \quad E_{\boldsymbol{\pi}}[f] \triangleq \sum_{u \in V} f(u)\pi(u).$$

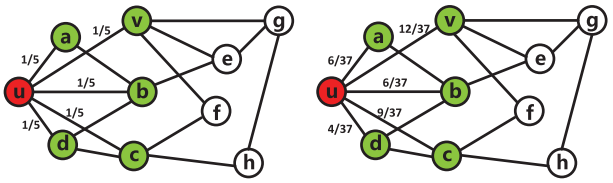
Note that in the above formula, $X_s \backsim \boldsymbol{\pi}$, $\mu_t(f)$ denotes the average of f over the samples, and $E_{\boldsymbol{\pi}}[f]$ denotes the mathematical expectation of f with respect to $\boldsymbol{\pi}$.

As random walks may not always produce uniformly distributed samples, to achieve unbiased estimation $E_U[f]$ (where U denotes the uniform distribution), we can correct the bias by using *Importance Sampling Framework* [11], [18]. That is, by setting a bias correction weight $\omega(X_s) = U(X_s)/\pi(X_s)$, we have

$$\frac{\sum_{s=1}^{t} \omega(X_s) f(X_s)}{\sum_{s=1}^{t} \omega(X_s)} \to E_U[f], \text{as } t \to \infty. \tag{1}$$

3 CNARW: Common Neighbor Aware Random Walk

In this section, we present the details of our common neighbor aware random walk (CNARW). Specifically, we first introduce the main idea of CNARW by using a simple example, then we present its algorithm design in details and provide theoretical analysis of its stationary distribution.



(a) SRW (uniform walking) (b) CNARW (weighted walking)

Fig. 1. Comparison of CNARW with SRW: SRW chooses next node uniformly at random from the current node's neighbors, while CNARW walks with higher probability to a neighbor which has larger degree and less common neighbors with the current node.

3.1 Main Idea of CNARW

The intuition of CNARW is simply "your common neighbors matter," and its main idea is to utilize the common neighbor information. To illustrate, suppose that the walker is currently at node u, so u's neighbors, i.e., nodes in $N(u)$, present as the candidates of next-hop nodes (see Fig. 1). Instead of choosing the next-hop node uniformly at random from all candidates as in SRW, which we call *uniform walking* (see Fig. 1a), CNARW differentiates the candidates by taking into consideration their degrees and the number of common neighbors between them and node u, which we call *weighted walking* (see Fig. 1b). Specifically, if a candidate node, say $v \in N(u)$, has a higher degree or less common neighbors with u, then the walker moves to v with a higher probability. That is, the weight of the transition probability from u to v is larger, e.g., in Fig. 1b, $P_{uv} = 12/37$ is the largest as v has higher degree but less common neighbors with u than other nodes in $N(u)$. In fact, we can easily verify from Fig. 1b that v should be a better choice as it is easier to explore more unvisited nodes though v.

The rationale of the above weighted walking strategy used by CNARW can be justified as follows. We observe that one key reason why simple random walk converges slowly is that it is easy to fall into local loops due to the high clustering feature, which is very common for OSNs. In other words, by moving to neighbors uniformly at random, it is very likely to walk back to previously visited nodes, and this kind of revisits clearly slow down the convergence. To avoid frequent revisits to preciously visited nodes so as to speed up the convergence, one way is to give higher priority to nodes which provide higher chance of exploring unvisited nodes in each walking step. Therefore, if a candidate node has a higher degree, then it may provide a higher chance of connecting to more unvisited nodes. However, if it has more common neighbors with previously visited nodes, then the walker may also have a high probability of walking back to those visited nodes through common neighbors. Thus, walking to a node which has higher degree but fewer common neighbors with previously visited nodes (or simply the current node) not only provides higher chance of walking to unvisited nodes, but also reduces the probability of walking back to visited nodes in the future walking steps. With the above weighted walking strategy, CNARW is expected to converge faster.

3.2 Algorithm Design of CNARW

To realize CNARW described above by using the weighted walking strategy, the key issue is to formulate the selection of the next-hop node with a mathematical model. Based on

the formulation, the transition probability matrix can be formulated, and the random walk algorithm can also be developed accordingly. In the following, we first formulate next node selection by leveraging the concept of "*set conductance*", then present the design of a transition matrix and show the random walk algorithm in details.

Importance of Common Neighbors. To evaluate the importance of common neighbors, we leverage the concept of "*set conductance*" [26]. Its definition is given below.

Definition 1. (Set Conductance) [26]. Let $G = (V, E)$ be an undirected graph and $C \subset V$ be a set of nodes, Let $\phi(C)$ be the conductance of set C and it is defined as

$$\phi(C) = \phi(C, \bar{C}) = |E_{C,\bar{C}}|/\mathrm{Vol}(C),$$

where $\bar{C} = V - C$, $E_{C,\bar{C}} = \{(u, v) \in E | u \in C, v \in \bar{C}\}$, and $\mathrm{Vol}(C) = \sum_{u \in C} deg(u)$, $deg(u)$ indicates the degree of vertex u in G.

Remark: Note that the conductance $\phi(C)$ can be considered as the ratio of the number of connections between C and $\bar{C}$ to the number of connections inside C. More importantly, the conductance of set C can be taken as an efficient indicator to reflect the difficulty of being trapped into the local community C if a walker is currently at a node in C. In particular, larger conductance $\phi(C)$ may imply a higher chance of not being trapped into the local subgraph, because larger $\phi(C)$ means more connections to nodes outside C, i.e., it provides higher chance of walking outside C.

Now we formulate the selection of the next-hop node by using the concept of set conductance described above. Suppose that the walker is currently at node u, we define a set of frontier nodes, which contains the current node and its neighbors, and call it *frontier set* denoted as S, i.e., $S = \{u\} \cup N(u)$. For example, as shown in Fig. 1a, $S = \{u, a, b, v, c, d\}$. According to previous discussions, $\phi(S)$ can be used as an indicator to characterize the difficulty of being trapped in S. Let $deg(S) = \sum_{i \in S} deg(i)$. We can derive $\phi(S)$ as

$$\phi(S) = |E_{S\bar{S}}|/deg(S).$$

Note that all candidates of the next-hop nodes are now in S, to evaluate the goodness of being selected as the next hop for each candidate, say node v, we characterize the contribution of v to the conductance of set S, which can be mathematically expressed as $\Delta\phi_v = \phi(S) - \phi(S_{-v})$ where $S_{-v} = S \setminus \{v\}$. For example, as in Fig. 1a, $S_{-v} = \{u, a, b, c, d\}$. The physical meaning is that if v contributes more to the conductance of set S, then walking through v may provide higher opportunities of exploring unvisited nodes outside S. We give the mathematical expression of $\Delta\phi_v$ in Theorem 2.

Theorem 2. *Given the current node u and it's frontier set S, the contribution of node v to the conductance of set S, denoted as $\Delta\phi_v$, can be derived as*

$$\Delta\phi_v = \frac{(1-\phi(S)) - 2(C_{uv}+1)/deg(v)}{(\sum_{i \in S} deg(i))/deg(v) - 1}, \tag{2}$$

where $deg(v)$ and C_{uv} denote the degree of v and the number of common neighbors between v and u, respectively.

Proof:

$$\begin{aligned}
\Delta\phi_v &= \phi(S) - \phi(S_{-v}) \\
&= \frac{|E_{S\bar{S}}|}{\sum_{i \in S} deg(i)} - \frac{|E_{S\bar{S}}| - [deg(v) - (C_{uv}+1)] + (C_{uv}+1)}{\sum_{i \in S} deg(i) - deg(v)} \\
&= \frac{deg(v)[\sum_{i \in S} deg(i) - |E_{S\bar{S}}|] - 2(C_{uv}+1)\sum_{i \in S} deg(v)}{\sum_{i \in S} deg(i)[\sum_{i \in S} deg(i) - deg(v)]} \\
&= \left[(1-\phi(S)) - \frac{2(C_{uv}+1)}{deg(v)}\right] \Big/ \left[\frac{\sum_{i \in S} deg(i)}{deg(v)} - 1\right].
\end{aligned}$$

Remark: We can see that $\Delta\phi_v$ is only dependent on $deg(v)$ and C_{uv}. For any fixed $deg(v)$, if v has fewer common neighbors with u, then it contributes more to the conductance (the higher $\Delta\phi_v$). On the other hand, if C_{uv} is fixed, then a larger degree implies a higher contribution to the conductance. Thus, the change of $\Delta\phi_v$ with respect to the degree and the number of common neighbors of a candidate node are consistent with the intuition behind the weighted walking strategy. In summary, for each $v \subset N(u)$, its contribution to the conductance of frontier set can be taken as an effective indicator to evaluate the goodness of being chosen as the next node. Precisely, *CNARW gives higher weights to nodes which contribute more to the conductance of the frontier set.*

Design of the Walker's Transition Matrix. To develop a weighted walking strategy, an intuitive strategy is to make the transition probability from u to v (i.e., P_{uv}) be proportional to $\Delta\phi_v$. To avoid computing $\phi(S)$, we can let P_{uv} be proportional to $1 - \frac{C_{uv}}{deg(v)}$. The rationale is that if v has a larger degree and fewer common neighbors with u, i.e., $1 - \frac{C_{uv}}{deg(v)}$ is larger, then the contribution of v to $\phi(S)$ is bigger, so the walk should select it as the next node with a higher probability. For ease of deriving the stationary distribution, we also guarantee the symmetric property when designing P_{uv}, i.e., to design a reversible random walk satisfying $\pi(u)P_{uv} = \pi(v)P_{vu}$ [29]. Mathematically, we let P_{uv} satisfy.[1]

$$P_{uv} \propto 1 - C_{uv}/\min\{deg(u), deg(v)\}. \tag{3}$$

We point out that one can also adopt other functions in Eq. (3) to develop a new transition matrix, for example, using $deg(u) + deg(v)$ or $\max\{deg(u), deg(v)\}$ can also satisfy the symmetric property. However, taking the minimum can eliminate the dominating effect of $deg(u)$ when it is very large so that different neighbors of u can be differentiated. Thus, taking the minimum usually leads to better performance, which is also validated via experiments.

Walking-With-Rejection Implementation. To realize the proportional strategy in Eq. (3), we need to know C_{uv} and $deg(v)$ for each $v \in N(u)$. This means we need to fetch all these neighbors from the network to conduct one step of random walk, which is expensive for high network query cost. To limit the overhead, CNARW adopts a walking-with-rejection policy: in each step, CNARW first selects a candidate node v from $N(u)$ uniformly at random with

1. In general, one can consider the common neighbors' weight in the transition matrix design as $P_{uv} \propto 1 - \frac{\alpha C_{uv}}{\min\{deg(u), deg(v)\}}$, where $0 \leq \alpha \leq 1$ to guarantee $0 \leq P_{uv} \leq 1$. When α decreases, the weight of common neighbors also decreases. We also experimentally study the impact of the common neighbors' weight in Section 6.5, and find that a larger common neighbors' weight always brings a better performance.

probability $b_{uv} = \frac{1}{deg(u)}$, if $v \in N(u)$, then fetches v's information from the network. After that, it accepts v as the next node with probability $q_{uv} = 1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$, where C_{uv} and $deg(v)$ can be computed based on the fetched information. Note that with the walking-with-rejection policy, we only need to fetch node v if it is accepted, so we do not have to access all nodes in $N(u)$, which largely reduces the network query cost.

The random walk can move from u to its neighbor v in one try with probability $\tilde{p}_{uv} = \frac{1}{deg(u)} \times (1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}})$. Note the selected node has a chance of being rejected with probability $\tilde{p}_{uu} = 1 - \sum_{v \in N(u)} \frac{1}{deg(u)} \times (1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}) > 0$. If this happens, the walker repeats the selection again by checking another randomly selected node from $N(u)$ until a selected node is accepted. We call the above process as one walking step of CNARW. We have:

Theorem 3. *The transition matrix $\boldsymbol{P} = [P_{uv}]_{u,v \in V}$ is*

$$P_{uv} = \begin{cases} \tilde{p}_{uv}/(1 - \tilde{p}_{uu}), & \text{if } v \in N(u), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $\tilde{p}_{uv}$ is defined as

$$\tilde{p}_{uv} = \begin{cases} \frac{1}{deg(u)} \times (1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}), & \text{if } v \in N(u), \\ 1 - \sum_{k \in N(u)} \tilde{p}_{uk}, & \text{if } v = u, \\ 0, & \text{otherwise}. \end{cases} \tag{5}$$

Proof: According to the walking-with-rejection and loop-until-accept policy, we have

$$\begin{aligned} P_{uv} &= \frac{q_{uv}}{deg(u)} \times \sum_{n=0}^{\infty} \left(\frac{\sum_{k \in N(u)} (1 - q_{uk})}{deg(u)} \right)^n \\ &= \tilde{p}_{uv} \times \frac{1}{1 - \tilde{p}_{uu}}. \end{aligned}$$

According to the above result, we can also to show that CNARW is a stochastic random walk in the following theorem:

Theorem 4. *Given an undirected and connected graph $G(V, E)$, CNARW on G is a stochastic random walk, i.e., $\sum_{k \in N(u)} P_{uk} = 1$.*

Proof: Based on Eqs. (4)–(5), we have

$$\begin{aligned} \sum_{k \in N(u)} P_{uk} &= \sum_{k \in N(u)} \tilde{p}_{uv} \times \frac{1}{1 - \tilde{p}_{uu}} \\ &= \frac{1}{1 - \tilde{p}_{uu}} \times \sum_{k \in N(u)} \tilde{p}_{uv} \\ &= \frac{1}{1 - \tilde{p}_{uu}} \times (1 - \tilde{p}_{uu}) = 1. \end{aligned}$$

Therefore, we can conclude that CNARW is a stochastic random walk.

Let us use Fig. 1 as an example to illustrate the walking process in one step. Note that $N(u) = \{a, b, v, c, d\}$ as shown in Fig. 1, the acceptance probabilities are $q_{ua} = 1/2$, $q_{uv} = 1$, $q_{ub} = 1/2$, $q_{uc} = 3/4$, $q_{ud} = 1/3$, and the transition probabilities are $P_{ua} = 6/37$, $P_{uv} = 12/37$, $P_{ub} = 6/37$, $P_{uc} = 9/37$, $P_{ud} = 4/37$. We can see that node v has a higher acceptance probability than other candidate nodes, and this implies that v will be selected as the next-hop node by CNARW with a higher probability, which is consistent with the intuition that v is a better choice as being the next node so as to explore more unvisited nodes. The complete random walk algorithm via CNARW is stated in Algorithm 1.

Algorithm 1. One Walking Step of CNARW

Input: current node u
Output: next-hop node v
do
  Select v uniformly at random from u's neighbors;
  Generate a random number $q \in [0, 1]$;
  Compute $q_{uv} = 1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$;
while ($q > q_{uv}$)
Return v;

3.3 Analysis of Stationary Distribution

To guarantee the effectiveness of CNARW, we provide the theoretical analysis to show that CNARW has a unique stationary distribution in Theorem 5. We also derive the probability distribution of each node and each edge being visited in Theorems 6 and 7.

Theorem 5. *Given an undirected and connected graph $G(V, E)$, CNARW on G has a unique stationary distribution.*

Proof: Note that for any node $u \in V$ and any $v \in N(u)$, the acceptance probability $q_{uv} = 1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$ is larger than 0 and the transition probability P_{uv} is also larger than 0. Thus, for any two nodes u and v in G, u and v are reachable from each other in finite steps for CNARW as G is an undirected and connected graph. Based on this, we conclude that the Markov chain constructed by CNARW is irreducible. Since any irreducible Markov chain on an undirected and connected graph has a unique stationary distribution [10], we can conclude that CNARW has a unique stationary distribution.

Theorem 6. *The stationary distribution $\boldsymbol{\pi}$ of CNARW satisfies the following condition: $\forall u, v \in V$, we have $\pi(u)/\pi(v) = [deg(u)(1 - \tilde{p}_{uu})]/[deg(v)(1 - \tilde{p}_{vv})]$. Further, we have $\pi(u) = Z \times deg(u)(1 - \tilde{p}_{uu})$, where Z is a normalization constant.*

Proof: To derive the stationary distribution $\boldsymbol{\pi}$, we first show the time reversibility of the Markov chain constructed by CNARW. According to Proposition 1.1 in [29], we only need to show that $\pi(u) \times P_{uv} = \pi(v) \times P_{vu}$ has a unique solution. Based on Eqs. (4)–(5), we have

$$\frac{\pi(u)}{\pi(v)} = \frac{P_{vu}}{P_{uv}} = \frac{deg(u)(1 - \tilde{p}_{uu})}{deg(v)(1 - \tilde{p}_{vv})}. \tag{6}$$

Note that $1 - \tilde{p}_{vv}$ and $1 - \tilde{p}_{uu}$ are fixed for a given graph, so Eq. (6) has a unique solution as $\sum_{u \in V} \pi(u) = 1$. So the stationary probability for any u can be derived as

$$\pi(u) = Z \times deg(u) \times (1 - \tilde{p}_{uu}), \tag{7}$$

where Z is the normalization constant.

Theorem 7. *After CNARW algorithm converges, for any edge $e_{uv} \in E$, the stationary probability of e_{uv} being visited $\pi(e_{uv})$ is $Z^{-1} \times (1 - C_{uv}/\min\{d_u, d_v\})$, and it satisfies $\pi(e_{uv}) = \pi(u) \times P_{uv}$.*

Proof: Let $X_t \in V$ $(t = 0, 1, 2, ...)$ denote the location at time t. We construct an expanded Markov chain $\{Z_t = (X_{t-1}, X_t)\}_{t\geq 1}$ with its transition matrix $EP = \{EP_{e_{ij},e_{lk}}\}_{e_{ij},e_{lk}\in E}$ given by

$$EP_{e_{ij},e_{lk}} = \begin{cases} P_{lk}, & \text{j=l,} \\ 0, & \text{j}\neq\text{l.} \end{cases} \tag{8}$$

One can easily find that the static probability of edge (u, v) being visited by CNARW is equal to the static probability of state e_{uv} being visited by the expanded Markov chain. Then, we can utilize the definition of static distribution [15]. We first prove that $\sum_{i\in V}\sum_{j\in N(i)} \pi(e_{ij}) = 1$:

$$\begin{aligned} &\sum_{i\in V}\sum_{j\in N(i)} \pi(e_{ij}) = \sum_{i\in V}\sum_{j\in N(i)} \frac{1}{Z} \times \left(1 - \frac{C_{ij}}{\min\{d_i, d_j\}}\right) \\ &= \sum_{i\in V}\sum_{j\in N(i)} \frac{d_i(1 - P_{ii})}{Z} \times \frac{\frac{1}{d_i}(1 - \frac{C_{ij}}{\min\{d_i,d_j\}})}{1 - p_{ii}} \\ &= \sum_{i\in V}\sum_{j\in N(i)} \pi(i)P_{ij} = \sum_{i\in V}\pi(i)\sum_{j\in N(i)} P_{ij} = \sum_{i\in V}\pi(i) = 1. \end{aligned}$$

We then prove $\pi(e_{ij}) = \sum_{k\in V}\pi(e_{ki})P^{'}(e_{ki}, e_{ij})$:

$$\begin{aligned} &\sum_{k\in V}\pi(e_{ki})P^{'}(e_{ki}, e_{ij}) \\ &= \sum_{k\in N(i)} \frac{1}{Z}\left(1 - \frac{C_{ki}}{\min\{d_k, d_i\}}\right) \times \frac{\frac{1}{d_i} \times (1 - \frac{C_{ij}}{\min\{d_i,d_j\}})}{1 - P_{ii}} \\ &= \frac{1}{Z}\left(1 - \frac{C_{ij}}{\min\{d_i, d_j\}}\right)\left(\frac{1}{1 - P_{ii}}\sum_{k\in N(i)}\frac{1}{d_i}(1 - \frac{C_{ki}}{\min\{d_k, d_i\}})\right) \\ &= \frac{1}{Z}\left(1 - \frac{C_{ij}}{min\{d_i, d_j\}}\right) = \pi(e_{ij}). \end{aligned}$$

Up till now, we have stated our novel design of CNARW framework, its corresponding random walk algorithm, and provided theoretical proof on its stationary distribution. In the next two sections, we will discuss (a) how to perform unbiased graph sampling based on our new algorithm, and (b) two variants of the algorithm based on CNARW framework which might have better performance.

4 Unbiased Graph Sampling

In this section, we introduce how to use CNARW to develop an asymptotically unbiased graph sampling. We focus on two sampling schemes, unbiased node sampling and unbiased edge sampling, which can be used to sample a sequence of nodes and a sequence of edges, respectively. Note that, nodes and edges may have label and property information in social networks, and such information can also help with the graph sampling, such as filtering the mismatching information. For example, suppose we want to estimate the average number of friends of female users, we could only sample the nodes labeled by ' *Person*' with the '*gender*' property valued by '*Female*'. Our proposed algorithms can also be applied to execute this kind of sampling tasks, and incorporate with such kind of label/property filtering mechanisms.

Algorithm 2. Unbiased Node Sampling via CNARW

Input: Graph $G(V, E)$, initial node x, sample size k
Output: estimated result $\mu(f)$

```
Run CNARW until converges; /* Burn-in Period */
Let u denote the first node being visited after convergence;
sum_f ← 0; sum ← 0;
for i = 1 to k do
    /* Sampling Phase after Convergence*/
    Sample node v based on node u via random walk defined
    by Algorithm 1, and get the value of γ(v);
    w = γ(v) / deg(v);
    sum_f += w × f(v); /* Aggregate Function f */
    sum += w; u ← v;
end
Return μ(f) = sum_f / sum;
```

Unbiased Node Sampling. Since the stationary probability distribution of a node being visited via CNARW is not uniform, bias correction is necessary to achieve asymptotically unbiased estimation. Based on *important sampling framework* (see Section 2.2), we set the bias correction weight $w(u)$ as:

$$w(u) = \gamma(u)/deg(u), \quad \text{where} \quad \gamma(u) = 1/(1 - \tilde{p}_{uu}). \tag{9}$$

With the above weight factors, unbiased estimation can be derived based on the following theorem.

Theorem 8. *For a function of interest f, which is related to node properties, and a set of samples R collected by CNARW, when $|R| \to \infty$, the* unbiased estimation with bias correction weight *of f over the samples, which we denote as $\mu(f)$, can be derived as follows:*

$$\mu(f) = \frac{\sum_{u\in R}\frac{\gamma(u)}{deg(u)}f(u)}{\sum_{u\in R}\frac{\gamma(u)}{deg(u)}} \to E_U(f), \ a.s.$$

Proof: Based on Theorem 1 and $U(i) = 1/n$, we have

$$\begin{aligned} \mu_t(f) &= \frac{\sum_{u\in R}\frac{\gamma(u)}{deg(u)}f(u)}{\sum_{u\in R}\frac{\gamma(u)}{deg(u)}} = \frac{\sum_{u\in R}\frac{1/|V|}{Z\times deg(u)\times(1-\tilde{p}_{uu})}f(u)}{\sum_{u\in R}\frac{1/|V|}{Z\times deg(u)\times(1-\tilde{p}_{uu})}} \\ &= \frac{\sum_{u\in R}\frac{U(u)}{\pi(u)}f(u)}{\sum_{u\in R}\frac{U(u)}{\pi(u)}} \to \frac{E_\pi(\frac{U(X)}{\pi(X)}f(X))}{E_\pi(\frac{U(X)}{\pi(X)})} \\ &= E_{\boldsymbol{\pi}}\left[\frac{U(u)}{\pi(u)}f\right] = E_U(f), \quad a.s. \end{aligned}$$

Remark: From Theorem 8, we can see that given a set of samples R, if we know the weight functions $w(u)$ $(u \in R)$, we can achieve an asymptotically unbiased estimation for $E_U(f)$. However, computing $w(u)$ requires us to compute $\gamma(u)$, which not only requires the degree of sampled nodes, but also requires the information of the neighbors of the sampled nodes. This may introduce a high network query cost. To address this efficiency issue, we propose an optimization technique to approximate $\gamma(u)$ as follows. Observe that $1 - \tilde{p}_{uu}$ is the probability of jumping out of node u, since the number of self-loop transitions to node u is

geometrically distributed, then $\gamma(u)$ corresponds to the average number of attempts required to jump out of node u. Based on this understanding, we can simply take the number of self-loops in node u as an approximation of $\gamma(u)$. We formally present the unbiased node sampling algorithm with this optimization in Algorithm 2.

Unbiased Edge Sampling. With CNARW, we can also perform an unbiased edge sampling. The whole sampling framework is similar to that of unbiased node sampling except for two things. First, the function f should be an aggregate function of an attribute defined on edges, and not an attribute defined on nodes as in node sampling. In particulary, f has a form of $f(e_{uv})$ where $e_{uv} \in E$. Second, the weight function ω should also be defined on edges, and we set the weight function $\omega(e_{uv})$ on edge e_{uv} as $w(e_{uv}) = 1/[1 - C_{uv}/\min\{d_u, d_v\}]$..

Based on the bias correction weight $w(e_{uv})$, we can also achieve an asymptotical unbiased edge sampling.

Theorem 9. *For a function of interest f, which is related to edge properties, and a set of samples R collected by CNARW, when $|R| \to \infty$, the* unbiased estimation*of f over the samples, which we denote as $\mu(f)$, can be derived as $\mu(f) = \frac{\sum_{e_{uv}\in R} w(e_{uv}) f(e_{uv})}{\sum_{e_{uv}\in R} w(e_{uv})} \to E_U(f)$, almost surely.*

Proof: Based on SLLN theorem and $U(e_{uv}) = \frac{1}{2|E|}$, we have

$$\begin{aligned}
\mu_t(f) &= \frac{\sum_{e_{uv}\in R} w(e_{uv}) f(e_{uv})}{\sum_{e_{uv}\in R} w(e_{uv})} \\
&= \frac{\sum_{e_{uv}\in R} 1/(1 - C_{uv}/\min\{d_u, d_v\}) \times f(e_{uv})}{\sum_{e_{uv}\in R} 1/(1 - C_{uv}/\min\{d_u, d_v\})} \\
&= \frac{\sum_{e_{uv}\in R} (\frac{1}{2|E|}/[\frac{1}{Z} \times (1 - \frac{C_{uv}}{\min\{d_u, d_v\}})] \times f(e_{uv})}{\sum_{e_{uv}\in R} (\frac{1}{2|E|}/[\frac{1}{Z} \times (1 - \frac{C_{uv}}{\min\{d_u, d_v\}})]} \\
&= \frac{\sum_{e_{uv}\in R} U(e_{uv})/\pi(e_{uv}) \times f(e_{uv})}{\sum_{e_{uv}\in R} U(e_{uv})/\pi(e_{uv})} \\
&\to \frac{E_\pi(\frac{U(X)}{\pi(X)} f(X))}{E_\pi(\frac{U(X)}{\pi(X)})} = E_{\boldsymbol{\pi}}\left[\frac{U(u)}{\pi(u)} f\right] = E_U(f), a.s.
\end{aligned}$$

We formally present the unbiased edge sampling algorithm in Algorithm 3.

Algorithm 3. Unbiased edge sampling via CNARW

Input: Graph $G(V, E)$, initial node x, sample size k
Output: estimated result $\mu(f)$

Run CNARW until converges; /* Burn-in Period */
Let u denote the first node being visited after convergence;
$sum_f \leftarrow 0$; $sum \leftarrow 0$;
for $i = 1$ *to* k **do**
  /* Sampling Phase after Convergence*/
  Sampling an edge e_{uv} via random walk defined by Algorithm 1;
  $w = 1/(1 - C_{uv}/\min\{d_u, d_v\})$;
  $sum_f += w \times f(u)$; /* Aggregate Function f */
  $sum += w$;
  $u \leftarrow v$;
end
Return $\mu(f) = sum_f / sum$;

5 Extensions: Algorithm Variants

We would like to emphasize that CNARW is really a general framework and one can develop numerous algorithms based on it; the one we have presented earlier in Section 3 is nothing but a typical and effective one of them. In this section, we will discuss two variant algorithms and their potential improvements over the basic one.

5.1 Incorporating NBRW With CNARW

In this subsection we introduce a variant of CNARW algorithm that may reduce the sampling cost, by incorporating the idea from non-backtracking random walk (NBRW) [18]. It is a typical random walk sampling algorithm, which was the first one to utilize the walking history to speed up sampling by avoiding backtracking to the node of previous random walk step. Authors in [18] show that NBRW is theoretically guaranteed to achieve unbiased graph sampling with higher efficiency than SRW.

Note that our CNARW approach is orthogonal to NBRW. To further improve the efficiency of CNARW, we can consider incorporating the idea of NBRW in CNARW by just avoiding backtracking to the node of previous step in CNARW. In particular, we modify the first step in Algorithm 1, i.e., if random walk visits node x in last step, we change the select probability b_{uv} as follows:

$$b'_{uv} = \begin{cases} \frac{1}{deg(u)-1}, & \text{if } v \in N(u), v \neq x, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Then, we accept the node selection and move random walk to node v with probability q_{uv}, or reject it and re-select a node otherwise. We loop the process until the random walk is successfully forwarded. We rewrite the transition probability $\boldsymbol{P'} = [P'_{uv}]_{u,v\in V}$ as

$$P'_{uv} = \begin{cases} \tilde{p}'_{uv}/(1 - \tilde{p}'_{uu}), & \text{if } v \in N(u), \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

where $\tilde{p}'_{uv}$ is defined as

$$\tilde{p}'_{uv} = \begin{cases} \frac{1}{deg(u)-1} \times (1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}), & \text{if } v \in N(u), v \neq x \\ 1 - \sum_{k\in N(u)} \tilde{p}_{uk}, & \text{if } v = u, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

We name this variant algorithm as NB-CNARW, which is stated in Algorithm 4.

Algorithm 4. One walking step of NB-CNARW

Input: current node u, last node x
Output: next-hop node v

do
  Select v uniformly at random from u's neighbors except x;
  Generate a random number $q \in [0, 1]$;
  Compute $q_{uv} = 1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$;
while $(q > q_{uv})$
Return v;

Theoretical Analysis of NB-CNARW. According to [18], by avoiding backtracking to the node of the previous walk step, the new random walk still has a unique stationary

distribution, so we will be able to realize unbiased graph sampling. When we use the new random walk to conduct unbiased graph sampling, we can get a smaller asymptotic variance than using the origin random walk. It is proven to guarantee that we can use the new random walk to realize the same sampling accuracy with fewer samples. We restate the authors' finding as follows.

Theorem 10. [18] [Theorem3] *Suppose that $\{X_t\}$ is an irreducible Markov chain on the state space N with transition matrix $P = \{P(u,v)\}$ and stationary distribution π. Construct a Markov chain $\{Z'_t\}$ on the state space Ω with transition matrix $P' = \{P'(e_{uv}, e_{xy})\}$ in which the transition probabilities $P'(e_{uv}, e_{xy})$ satisfy the following two conditions:*

for all $e_{uv}, e_{vu}, e_{vy}, e_{yv} \in \Omega$ with $u \neq y$,

$$P(v,u)P'(e_{uv}, e_{vy}) = P(v,y)P'(e_{yv}, e_{vu}), \tag{13}$$

$$P'(e_{uv}, e_{vy}) \geq P(v,y). \tag{14}$$

Then, the Markov chain $\{Z'_t\}$ is irreducible and non-reversible with a unique stationary distribution π' in which $\pi'(e_{uv}) = \pi(u)P(u,v), e_{uv} \in \Omega$. Also, for any function f, the asymptotic variance of $\tilde{\mu}'_t(f)$ is not larger than of $\tilde{\mu}_t(f)$, i.e., $\sigma'^2(f) \leq \sigma^2(f)$.

Therefore, if we can prove CNARW and NB-CNARW satisfy the two conditions in (13) and (14), then we can obtain NB-CNARW algorithm's stationary distribution π' and the asymptotic variance $\tilde{\mu}'_t(f)$. We state them in the following theorem.

Theorem 11. *After NB-CNARW converges, for any edge $e_{uv} \in E$, the stationary probability of e_{uv} being visited satisfies $\pi'(e_{uv}) = \pi(u) \times P(u,v) = \pi(e_{uv})$, i.e., it equals the stationary probability of e_{uv} being visited in CNARW. Besides, for any function f, the asymptotic variance of NB-CNARW $\tilde{\mu}'_t(f)$ is not larger than of CNARW $\tilde{\mu}_t(f)$, i.e., $\sigma'^2(f) \leq \sigma^2(f)$.*

Proof: Let $X_t \in V$ ($t = 0, 1, 2, ...$) denote the location of an CNARW. We construct an expanded Markov chain based on Theorem 10, and then prove Eqs. (13)–(14) respectively.

First, we have:

$$P(v,u)P'(e_{uv}, e_{vy}) = \frac{1}{deg(v)} \times \frac{1}{deg(v)-1} = P(v,y)P'(e_{yv}, e_{vu})$$

Note that we also have:

$$P'(e_{uv}, e_{vy}) = \frac{1}{deg(v)-1} \geq \frac{1}{deg(v)} = P(v,y).$$

From Theorem 11, we can derive the stationary distribution π' of NB-CNARW. When we use NB-CNARW to conduct unbiased graph sampling, we can get a smaller asymptotic variance and realize the same sampling accuracy with fewer samples than using CNARW. In other words, we can further decrease the sampling cost by NB-CNARW.

5.2 Extension of CNARW to Utilize More Visited Nodes

In this subsection, we state the second variant algorithm, by utilizing more visited nodes. Note that the algorithm CNARW in Algorithm 1 only uses the information of current node, or it only looks back one step. Considering that the larger the number of steps to look back, the more historical information the walk can have, which will lead to faster convergence speed. Thus, it is interesting to study how much gain can be further obtained by considering more visited nodes. To answer this problem, we consider an extension of CNARW by utilizing multiple previously visited nodes. We first extend the definition of frontier set, and denote H as the number of previously visited nodes being utilized, In particular, $H = 0$ corresponds to SRW and $H = 1$ corresponds to CNARW. For $H \geq 2$, we redefine the frontier set $S = N(x_H) \cup N(x_{H-1}) \cup \ldots \cup N(x_2) \cup N(u)$, where u is the current node. For a candidate node $v \subset N(u)$, we characterize the contribution of v to the conductance of S, which can be mathematically expressed as $\Delta\phi_v^H = \phi(S) - \phi(S_{-v})$ where $S_{-v} = S \setminus \{v\}$. Through similar derivation as in Eq. (2), we can get the following result:

$$\Delta\phi_v^H = \left[(1-\phi(S)) - \frac{2(C_{Sv}+1)}{deg(v)}\right] \Bigg/ \left[\frac{\sum_{i\in S} deg(i)}{deg(v)} - 1\right].$$

One can easily see that the $\Delta\phi_v^H$ is only dependent on $deg(v)$ and C_{Sv}, which denote the degree of v and the number of neighbors of v in the frontier set S respectively. Following the design of CNARW, we can define the transition probability from node u to node v as follows:

$$P_{uv}^H = \begin{cases} \frac{\tilde{p}_{uv}^H}{1-\tilde{p}_{uu}^H}, & \text{if } v \in N(u), \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

where $\tilde{p}_{uv}^H$ is defined as

$$\tilde{p}_{uv}^H = \begin{cases} \frac{1}{deg(u)} \times (1 - \frac{C_{Sv}}{\min\{|S|, deg(v)\}}), & \text{if } v \in N(u), \\ 1 - \sum_{k\in N(u)} \tilde{p}_{uk}^H, & \text{if } v = u, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Now the complete random walk algorithm by taking use of H previously visited nodes is stated in Algorithm 5.

Algorithm 5. One Step of the Extended CNARW

Input: current node u, a queue Q_H
/* Q_H stores the H most recently visited nodes*/
Output: next-hop node v
do
  Select v uniformly at random from u's neighbors;
  Compute C_{Sv} and $|S|$ by using the queue Q_H;
  Generate a random number $q \in [0,1]$;
  Compute $q_{Sv} = 1 - \frac{C_{Sv}}{\min\{|S|, deg(v)\}}$;
while ($q > q_{Sv}$)
Pop the tail node of Q_H and push node v into Q_H;
Return v;

To answer how much history information is good enough (i.e., to determine the best value of H), we study the impact of H on the convergence rate through experiments

TABLE 1
Summary of Datasets

Name	# of Nodes	# of Edges	Avg. Clustering Coefficient
Facebook	775	28012	0.4714
Ca-GaQc	2879	18474	0.4416
Phy1	4158	26844	0.5486
Google Plus	64517	2867802	0.3428
Flickr	80513	11799764	0.1652
DBLP	226413	1432920	0.6353
LiveJournal	1500000	29425194	0.2552

in Section 6.4. In fact, our experiments show that using only the current node is adequate and the benefit is twofold. First, it is much easier and more efficient to implement the algorithm when using only the current node compared to using multiple previously visited nodes, and this is usually one of the key factors when considering to deploy an algorithm in practical applications. Second, as shown by our experiment results in Section 6.4, leveraging only the current node already takes most of the benefit of speeding up the convergence. Thus, the marginal benefit of considering more historical information might be marginal.

Summary. We stated two typical variants of random walk algorithms based on the CNARW framework, but there are unlimited potentials to explore new variants. We would also like to mention that for each of the variant random walk algorithms, one can use the same methodology to design unbiased graph sampling algorithms so as to realize unbiased node/edge sampling.

6 Evaluation

In this section, we conduct extensive experiments on real-world network datasets to evaluate the effectiveness and efficiency of CNARW. Experiment results show that CNARW reduces the query cost significantly over the state-of-the-art sampling algorithms with the same estimation accuracy. We also reveal fundamental understandings on why CNARW has such a significant improvement.

6.1 Datasets & Experiment Setup

We conduct experiments on the datasets released by Leskovec *et al.* [20] and Rossi *et al.* [28]. We present some simple statistics of these datasets in Table 1. For datasets that are directed graphs, we convert them into undirected graphs by selecting the largest connected component after removing edges which appear in one direction only. This method has been used in prior works [3], [24], [25], [35], [36]. We categorize the datasets into two groups: (1) large-scale graphs, i.e., Google Plus, Flickr, DBLP and LiveJournal, which are used to study the performance measures like convergence rate and query cost; and (2) small-scale graphs, i.e., Facebook, Ca-GaQc, and Phy1, which are used to study mixing rate which is computationally expensive for large-scale datasets.

We compare our algorithm with three typical random walk sampling algorithms (RWSAs): (1) simple (or naive) random walk (*SRW*) [19], which serves as our comparison baseline; (2) non-backtracking random walk (*NBRW*) [18], which was the first one to utilize the walking history to speed up sampling; (3) circulated neighbors random walk (*CNRW*) [35], which is the state-of-the-art walking history aware sampling algorithm. All algorithms are implemented in C++, and we conducted experiments on a computer with two Intel Xeon E5-2650 2.60GHz CPUs and 64GB RAM.

TABLE 2
The SLEM for SRW and CNARW: Smaller SLEM Means Faster Convergence Speed

Algorithms	Facebook	Ca-GaQc	Phy1
SRW	0.9923	0.9981	0.9981
CNARW	0.9852	0.9820	0.9847

6.2 Convergence Speed

Mixing Time. We measure how fast a random walk sampling algorithm converges to its stationary distribution by using the concept of mixing rate [22]. One key indicator is the second largest eigenvalue modulus (SLEM) of the transition matrix [22]. The smaller the SLEM is, the faster the random walk converges. Since it is very expensive to compute SLEM, we consider three small-scale social networks listed in Table 1. Besides, since the closed-form transition matrices of NBRW and CNRW are hard to derive, we only compare our CNARW with SRW. Table 2 shows the results of SLEM for SRW and CNARW. We see that our CNARW indeed has a smaller SLEM than SRW. This means that CNARW should converge faster to the stationary distribution than SRW.

Convergence to Mean. To further evaluate how much improvement CNARW can achieve on large graphs, and also study its performance compared to NBRW and CNRW, we show the convergence speed by evaluating another concept called *converge to mean*, which characterizes the convergence to the mean of a graphs statistics, e.g., average node degree, instead of using the convergence to the stationary distribution. Note that the notion of convergence to mean only applies to sampling tasks that estimate the mean of some functions defined on the sampled variables, e.g., average node degree and average local clustering coefficient, and this metric depends on the sampling task. To quantify the speed of convergence to mean, we define

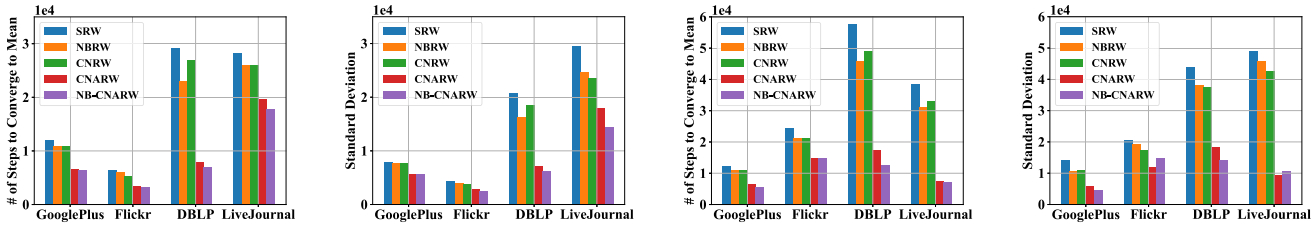
$$T_{\mathrm{cm}} \triangleq \mathbb{E}[\text{min. \# of steps needed to converge to the mean}].$$

It is also computationally expensive to compute the exact value of T_{cm}. Thus, we estimate T_{cm} by simulating the RWSA, and use the Geweke convergence monitor to detect whether a RWSA converges to the mean. Note that this method is computationally efficient and scalable to large scale networks. The Geweke convergence monitor has been widely used in prior works[6], [8], and we set its key parameter, i.e., the threshold $Z \leq 0.1$ by default.

To evaluate the rate of convergence to mean, i.e., T_{cm}, we repeat the simulation for n times to obtain n samples $T_{\mathrm{cm}}^1, \ldots, T_{\mathrm{cm}}^n$, and study both the mean and standard deviation. Mathematically, we evaluate the average convergence rate by the following metric:

$$\bar{T}_{\mathrm{cm}} \triangleq \textstyle\sum_{i=1}^{n} T_{\mathrm{cm}}^i / n.$$

We apply standard deviation (SD) to quantify the variation of convergence rate, and we use the following estimator:



(a) $\bar{T}_{\mathrm{cm}}$ (Degree as Indicator) (b) $\sigma(T_{\mathrm{cm}})$ (Degree as Indicator) (c) $\bar{T}_{\mathrm{cm}}$ (Similarity as Indicator) (d) $\sigma(T_{\mathrm{cm}})$ (Similarity as Indicator)

Fig. 2. Comparison of convergence speed, which is measured by the average and standard deviation (SD) of the convergence to mean (i.e., $\bar{T}_{\mathrm{cm}}$ and $\sigma(T_{\mathrm{cm}})$) when using node degree and node pair similarity as the indicator.

$$\sigma(T_{\mathrm{cm}}) \triangleq \sqrt{\sum_{i=1}^{n}(T_{\mathrm{cm}}^{i} - \bar{T}_{\mathrm{cm}})^2/(n-1)}.$$

In particular, we take the average node degree and average node pair similarity as indicators. Figs. 2a and 2b presents the minimum number of steps needed to converge to mean $\bar{T}_{\mathrm{cm}}$ and its corresponding standard deviation $\sigma(T_{\mathrm{cm}})$ respectively by taking node degree as the indicator of convergence. Figs. 2c and 2d show the results by taking similarity of node pairs which is computed by using Jaccard index [5] as the indicator of convergence. Each value of $\bar{T}_{\mathrm{cm}}$ and $\sigma(T_{\mathrm{cm}})$ is estimated by using 300 runs of an algorithm. From Fig. 2 we observe that our algorithm CNARW has a smaller $\bar{T}_{\mathrm{cm}}$ than SRW, NBRW and CNRW, which means that CNARW converges faster. In particular, CNARW requires fewer steps to converge to mean, e.g., the reduction is up to 71.9%. We also observe that $\bar{T}_{\mathrm{cm}}$ varies across datasets, and this means that graph structure has an significant impact on convergence speed. For example, the number of steps required to converge to mean increases significantly from Flickr to LiveJournal. Thus, we need more steps to converge to mean when we sample a graph with larger number of nodes, and this also implies that speeding up the convergence of a sampling process is really meaningful, especially for large graphs. Besides, Fig. 2 also shows that our CNARW has a smaller standard deviation on $\bar{T}_{\mathrm{cm}}$ than SRW, NBRW and CNRW. This means that the variation of the convergence speed when using our CNARW is smaller. This property is also very important in practical systems, e.g., it can make our CNARW more suitable for parallel sampling. Besides, it shows that NB-CNARW always has a smaller $\bar{T}_{\mathrm{cm}}$ than CNARW from the results, and the reduction is up to 29.2%. This means by using the variant algorithm NB-CNARW, we can further speed up the convergence speed and decrease the burn-in time. We also observe a smaller $\sigma(T_{\mathrm{cm}})$ in NB-CNARW, which means a smaller variance of the convergence speed of NB-CNARW.

6.3 Sampling Cost

Estimation Error versus Query Cost. A fundamental tradeoff of sampling algorithms is: *estimation error* versus *query cost*. The estimation error of a sampling algorithm decreases as the query budget (or query cost) increases. In this paper, we adopt relative error to quantify the estimation accuracy:

$$\text{relative error} \triangleq |\hat{X} - X|/X,$$

where $\hat{X}$ and X denote the estimated value and the ground truth of a specific measure, e.g., average degree.

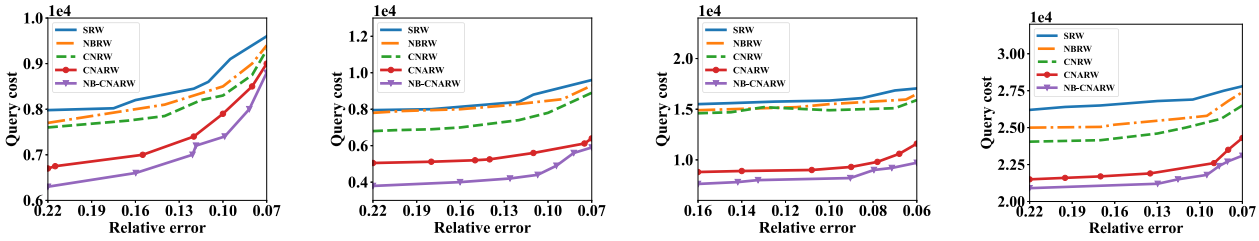
We define the query cost as the total number of unique queries (i.e., the number of sampled unique nodes queried from the network) in a sampling task, including the queries in both burn-in period and sampling phase:

$$\text{query cost} \triangleq \#\{\text{unique queries in a sampling task}\}.$$

For example, suppose that a sampling task visits a sequence of nodes (a, b, c, d, a, c, d), then the query cost is 4. This metric is widely used to evaluate the efficiency of sampling algorithms [25], [35], [36]. The reason why we consider only unique queries is that once a node is visited, we can store its associated information in local, and thus when it is visited again, we do not need to query from the network again. Note that in CNARW, each step may incur additional queries to find a better next hop, we also include this part when evaluating the query cost of CNARW.

We now study the tradeoff between estimation error and query cost. We only show the results of one typical sampling task, i.e., average degree estimation. We observe similar results for other sampling tasks like average clustering coefficient estimation. We run five sampling algorithms (i.e., SRW, NBRW, CNRW, and our algorithm CNARW, NB-CNARW) on four large-scale datasets. Each algorithm is repeated for 200 times to estimate the average node degree and we also take an average to measure query cost. Note that all these algorithms can realize the unbiased node sampling through bias correction with theoretical guarantees. Fig. 3 shows the tradeoff between estimation error and query cost, where the horizontal axis represents the estimation error and the vertical axis represents the corresponding query cost. From Fig. 3, we observe that the query cost increases as the relative error decreases, which implies that we need more queries to increase the estimation accuracy. We also observe that CNARW requires a smaller query cost to achieve the same estimation accuracy. Furthermore, the reduction in query cost (or improvement in estimation accuracy) is significant for CNARW (e.g., by up to 35.7%). Besides, NB-CNARW always requires a smaller query cost to achieve the same estimation accuracy compared to CNARW, and the reduction of query cost is up to 25.4%.

Query Cost and Estimation Error Comparison. We also study the query cost and estimation error comparison under different sample size settings, and the results are shown in Fig. 4. Note that the sample size denotes the number of samples collected after the random walk converges. The query cost is defined as the total number of unique sampling nodes queried from the network in a sampling task, including the queries in both the burn-in period and the sampling



(a) Google Plus (b) Flickr (c) DBLP (d) LiveJournal

Fig. 3. Tradeoff between estimation error and query cost, where the query cost is defined as the total number of unique queries, i.e., #(sampled unique nodes), in a sampling task, including the queries in both burn-in period and sampling phase.

phase. We observe that as the sample size increases, the query cost also increases, and the relative error decreases accordingly, which implies that we need more samples and queries to increase the estimation accuracy. We also observe that the five algorithms achieve similar estimation accuracy with the same sample size setting, but our algorithms CNARW and NB-CNARW always require the lowest query cost.

Time Cost Comparison. We further evaluate the actual time cost of the five algorithms under the network sampling situations. The total time cost includes two parts: (1) Network query time cost and (2) Compute time cost. For the network query time cost, it can be simulated through the *sleep()* function. In our experiment, we set the latency as $(106.603 + \frac{4 \times deg(u)}{956.410})$ms for querying a node u with $deg(u)$ neighbors in G, by deriving from the *ping and wget test results for the SNAP datasets*. The compute time cost includes the local computing for C_{uv} and $\gamma(u)$ in each step. We show the total time cost and the compute time cost under different sample size settings in Fig. 5. We can see that CNARW and NB-CNARW bring much higher local compute time cost than the baselines due to the extra computation of C_{uv} and $\gamma(u)$. However, in the network sampling situations, the compute time cost only accounts for a small proportion, and the total time cost is dominated by the network query cost. Therefore, CNARW and NB-CNARW cost much less total time cost because of the largely reduced network query cost.

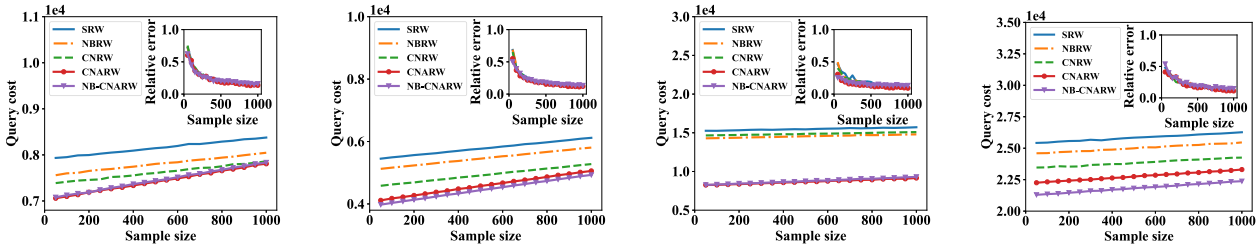
6.4 Tradeoff of Using More History Information

Our experimental results thus far consider one historical neighbor for our CNARW and NB-CNARW. We also run experiments to further show the impact of H on the convergence speed. Fig. 6 presents the speed of convergence to mean (i.e., $\bar{T}_{\text{cm}}$) and the rejection rate when H varies from 0 to 5. We observe that $\bar{T}_{\text{cm}}$ decreases significantly when H increases from 0 to 1. This implies that we can speed up the convergence by exploiting the current node. However, when utilizing more history information by further increasing H, i.e., considering more previously visited nodes, we can only have a diminish return. In particular, the reduction of the number of steps required to converge is not significant any more when we consider more than one visited node.

Therefore, we conclude that utilizing more previously visited nodes brings quite limited benefits for the following reasons: (1) Larger H increases the rejection rate in each walking step as shown in Fig. 6, which increases the average number of nodes queried from the network, i.e., larger H introduces higher query cost, which severely increases the time cost. (2) Larger H decreases the difference between C_{Sv} and $min\{|S|, deg(v)\}$, so we have no choice but to choose the next node v from $N(u)$ uniformly at random just like SRW does. Thus, it is good enough to consider one visited node only as CNARW and NB-CNARW.

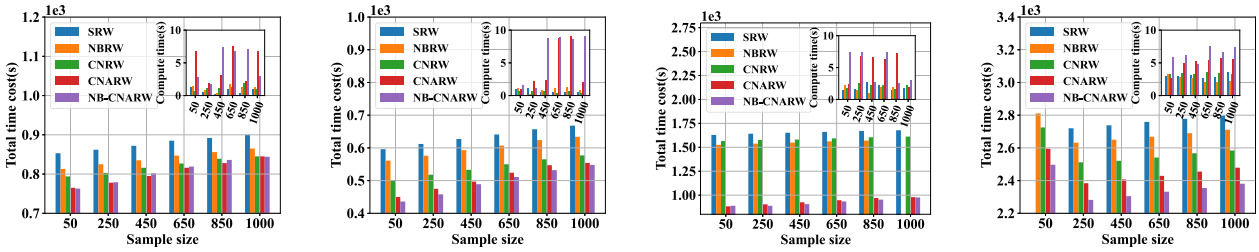
6.5 Impact of Common Neighbors' Weight

We further consider the common neighbors' weight in the transition matrix design as $P_{uv} \propto 1 - \frac{\alpha C_{uv}}{\min\{deg(u), deg(v)\}}$, where $0 \leq \alpha \leq 1$. When α decreases, the weight of common neighbors also decreases. In particular, $\alpha = 0$ and $\alpha = 1$ represents the SRW and CNARW, respectively. Fig. 7 shows the convergence speed with respect to various common neighbors' weight, and the results show that a larger common neighbors' weight always brings a better performance, i.e., fewer steps to converge to the mean. This further validates the positive impact of taking common neighbors into transition matrix designs.



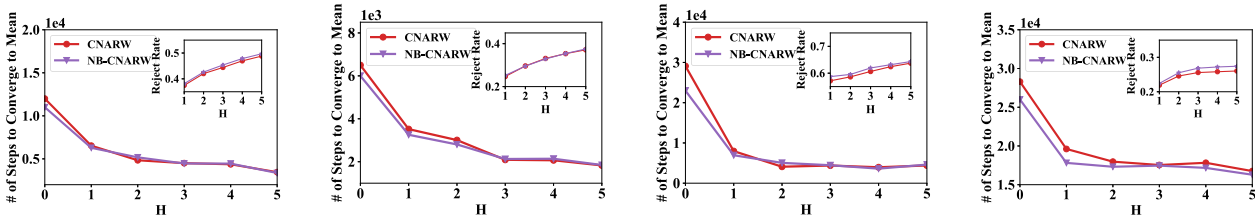
(a) Google Plus (b) Flickr (c) DBLP (d) LiveJournal

Fig. 4. Query cost and estimation error comparison under different sample size settings, where the query cost is defined as the total number of unique queries, i.e., #(unique sampled nodes) here, in a sampling task, including the queries in both burn-in period and sampling phase.



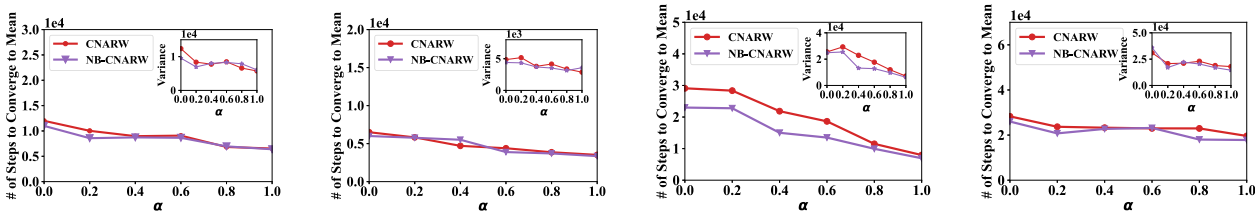
(a) Google Plus (b) Flickr (c) DBLP (d) LiveJournal

Fig. 5. Time cost comparison under different sample size settings.



(a) Google Plus (b) Flickr (c) DBLP (d) LiveJournal

Fig. 6. Impact of utilizing H previously visited nodes ($H = 1$ for CNARW and $H = 0$ for SRW).



(a) Google Plus (b) Flickr (c) DBLP (d) LiveJournal

Fig. 7. Impact of utilizing α as common neighbors' weight ($\alpha = 1$ for CNARW and $\alpha = 0$ for SRW).

6.6 Applications of CNARW

In this subsection, we investigate two applications, which we mentioned in Section 1, to further study the accuracy and efficiency of CNARW.

Application 1: Investment on Networking Platforms. A fundamental problem for this application is to estimate the average similarity of node pairs. We use the Jaccard index [5] to calculate the similarity of a node pair, and focus on the average similarity over all node pairs. We apply CNARW to sample edges and make unbiased estimation based on Algorithm 3. According to the convergence rate of CNARW in Fig. 2c, we run CNARW until the total number of sampling edges reach 10^5 for each sampling process. Then we use the sampled edges to estimate the average similarity. To verify the accuracy of CNARW, we repeat the sampling process three times (each time with a random start), and the corresponding experimental results are shown in Figs. 8a and 8b. One can observe that, as the number of samples increases, the relative error drops fast and it is close to zero when the number of samples is more than 10^4. This implies CNARW can accurately estimate the average edge similarity with a small number of samples. Note that we only show the experiment results of Flickr and Google Plus, and similar results can also be found for other two datasets. We further show the efficiency of CNARW in Figs. 8c and 8d, where the horizontal axis represents the estimation error and the vertical axis represents the corresponding query cost. Note that CNRW is not included here, because CNRW is not suitable for edge sampling. From Figs. 8c and 8d, we observe that CNARW requires a smaller query cost to achieve the same estimation accuracy, which implies a higher efficiency.

Application 2: Bundling Strategy in Viral Marketing. A fundamental problem for this application is to estimate the distribution of user interests for each product. In particular, we aim to estimate the distribution of user interests in different age groups. Note that our datasets only contain the topology of OSNs, we thus synthesize the distribution of user interests, which are shown in Table 3. Specifically, we consider four different age groups and three products. The percentage of populations of each age group is shown in the first column, and the percentage of users in each age group being interested in each product are shown in the right three columns. Based on this dataset, we assign to each user with her interested products, and Fig. 9a shows the ground truth for the ratio of populations in different age groups who are interested in each product. From Fig. 9a, one can observe that, product A and B should be sold as a bundle, since they

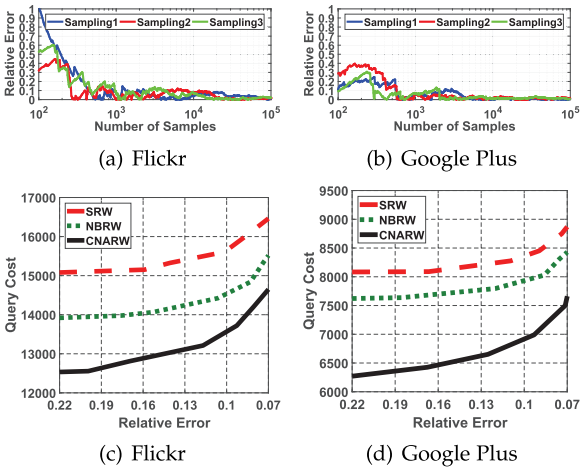


Fig. 8. Performance of CNARW in Application 1.

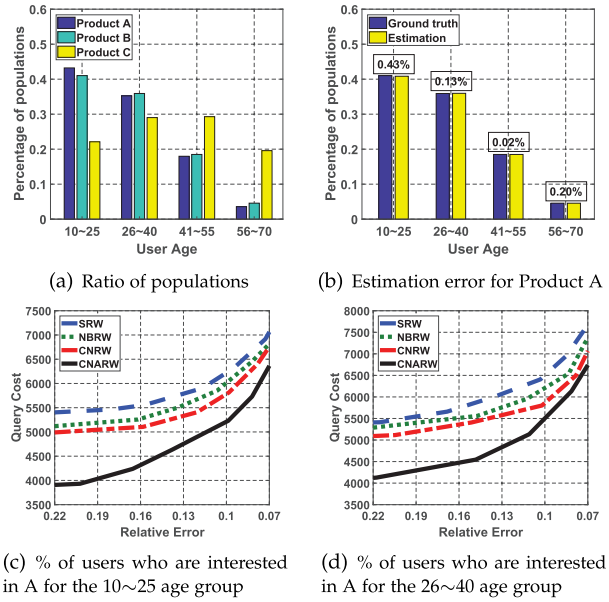


Fig. 9. Performance of CNARW in Application 2.

have similar distribution of user interests. To estimate the distribution of user interests, we apply CNARW to collect 10^5 node samples and make an unbiased estimation based on Algorithm 2. Fig. 9b shows that the estimated distribution of user interests using our CNARW's is very close to the ground truth, implying high accuracy. Figs. 9c and 9d further show that our CNARW is more efficient than the stat-of-the-art sampling algorithms.

Lessons Learned. Our CNARW is highly accurate and efficient in estimating graph measures defined on both nodes and edges. For example, a product owner can apply our CNARW to estimate the average similarity of node pairs with edge samples, and can also accurately estimate the distribution of user interests with node samples. More importantly, our CNARW requires much less query cost than the stat-of-the-art sampling algorithms.

7 Related Works

Traditional sampling methods, e.g., random node sampling, random edge sampling, and random subgraph sampling, all need the knowledge of the global graph topology [19], which is infeasible in many applications like network sampling situations. Instead, graph sampling via crawling, e.g., BFS, DFS, and random walk, have been widely used for that purpose. Even though BFS and DFS are simple to implement, they are hard to derive the sampling probability [17], thus they can not be used for unbiased estimation. Random walk sampling has become the mainstream for its simplicity of implementation and solid theoretical foundations of bias analysis [9], [18], [21], [35], [36].

TABLE 3
Summary of Datasets

Age groups	Product A	Product B	Product C
10∼25 (30%)	0.8	0.9	0.2
26∼40 (40%)	0.5	0.6	0.2
41∼35 (20%)	0.5	0.6	0.6
56∼70 (10%)	0.2	0.3	0.8

This table shows the percentage of users in each age group being interested in each product. Note that a user may be interested in multiple products.

To improve the efficiency and effectiveness of random walk sampling, a variety of methods have been proposed. Jin *et al.* [12] and Xu *et al.* [32] considered random jumping to increase the estimation accuracy. Lee *et al.* [18] proposed non-backtracking and delayed acceptance to reduce the asymptotic variance of estimators. Li *et al.* [21] combined the idea of delayed acceptance with Metropolis-Hastings algorithm to further reduce the asymptotic variance. Ribeiro *et al.* [27] proposed a multidimensional random walk and Zhao *et al.* [34] proposed a multi-graph random walk to address the limitation that a walk can easily get trapped by local communities.

In the aspect of speeding up random walk sampling, Boyd *et al.* [2] applied optimization techniques to optimize the mixing rate, but requiring full information of the graph. Avrachenkov *et al.* [1] combined uniform sampling with random walk sampling to speed up the convergence. Zhou *et al.* [35], [36] proposed to speed up the convergence by utilizing the walking history, which was first utilized in non-backtracking random walk (NBRW) [18]. They constructed a "virtual" overlay network from the walking history to guide the walker in [36], and later, they proposed the Circulated Neighbors random walk (CNRW) by constructing a higher-ordered Markov chain [35].

The difference of CNARW from existing approaches is that CNARW speeds up the convergence by utilizing the walking history and next-hop candidates, i.e., the number of common neighbors between visited nodes and next-hop candidates. Querying the neighbors of a next-hop candidate only incurs a small overhead. Besides, the unbiased sampling scheme via CNARW only requires the information of visited nodes only, but not requires the information about their neighbors, so CNARW provides fast and efficient unbiased graph sampling.

8 Conclusion

In this paper, we propose a fast and efficient random walk based sampling approach. Specifically, we first design a

common neighbor aware random walk to speed up the convergence, which takes into account both measures of degree and the number of common neighbors between next-hop candidate nodes and the current node, and then develop an efficient unbiased sampling scheme with theoretical guarantee on the unbiasedness by using the tailored random walk. We also conduct extensive experiments with real-world graph datasets, and results show that our sampling approach not only speeds up the convergence, but also reduces the query cost with the same estimation accuracy.

REFERENCES

[1] K. Avrachenkov, B. Ribeiro, and D. Towsley, "Improving random walk estimation accuracy with uniform restarts," in *Proc. Int. Workshop Algorithms Models Web-Graph*, 2010, pp. 98–109.

[2] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, 2004.

[3] X. Chen, Y. Li, P. Wang, and J. Lui, "A general framework for estimating graphlet statistics via random walk," *Very Large Data Bases*, vol. 10, no. 3, pp. 253–264, 2016.

[4] F. Chiericetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós, "On sampling nodes in a network," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 471–481.

[5] G. G. Chowdhury, *Introduction to Modern Information Retrieval*. London, U.K.: Facet, 2010.

[6] M. K. Cowles and B. P. Carlin, "Markov chain monte carlo convergence diagnostics: A comparative review," *J. Amer. Statist. Assoc.*, vol. 91, no. 434, pp. 883–904, 1996.

[7] A. De Bruyn and G. L. Lilien, "A multi-stage model of word-of-mouth influence through viral marketing," *Int. J. Res. Marketing*, vol. 25, no. 3, pp. 151–163, 2008.

[8] J. Geweke *et al.*, *Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments*, vol. 196. Federal Reserve Bank of Minneapolis, Research Dept. Minneapolis, Minneapolis, MN, USA, 1991.

[9] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: A case study of unbiased sampling of OSNs," in *Proc. IEEE Infocom*, 2010, pp. 1–9.

[10] O. Häggström, *Finite Markov Chains and Algorithmic Applications*, vol. 52. New York, NY, USA: Cambridge Univ. Press, 2002.

[11] M. H. Hansen and W. N. Hurwitz, "On the theory of sampling from finite populations," *Ann. Math. Statist.*, vol. 14, no. 4, pp. 333–362, 1943.

[12] L. Jin *et al.*, "Albatross sampling: Robust and effective hybrid vertex sampling for social graphs," in *Proc. 3rd ACM Int. Workshop MobiArch*, 2011, pp. 11–16.

[13] G. L. Jones *et al.*, "On the Markov chain central limit theorem," *Probability Surv.*, vol. 1, no. 299,320, pp. 5–1, 2004.

[14] R. A. King, P. Racherla, and V. D. Bush, "What we know and don't know about online word-of-mouth: A review and synthesis of the literature," *J. Interactive Marketing*, vol. 28, no. 3, pp. 167–183, 2014.

[15] T. Konstantopoulos, "Introductory lecture notes on Markov chains and random walks," *Dept. Math. Uppsala Univ.*, vol. 200, no. 9, pp. 1–128, 2009.

[16] M. Kurant, M. Gjoka, C. T. Butts, and A. Markopoulou, "Walking on a graph with a magnifying glass: Stratified sampling via weighted random walks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, pp. 241–252, 2011.

[17] M. Kurant, A. Markopoulou, and P. Thiran, "Towards unbiased BFS sampling," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1799–1809, 2011.

[18] C.-H. Lee, X. Xu, and D. Y. Eun, "Beyond random walk and metropolis-hastings samplers: Why you should not backtrack for unbiased graph sampling," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, pp. 319–330, 2012.

[19] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2006, pp. 631–636.

[20] J. Leskovec and A. Krevl, SNAP Datasets: Stanford large network dataset collection, Jun. 2014. [Online]. Available: http://snap.stanford.edu/data

[21] R.-H. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin, "On random walk based graph sampling," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 927–938.

[22] L. Lovsz, "Random walks on graphs: A survey," *Combinatorics, Paul Erdos Eighty*, vol. 2, pp. 1–46, 1993.

[23] A. Mohaisen and S. Hollenbeck, "Improving social network-based sybil defenses by rewiring and augmenting social graphs," in *Proc. Int. Workshop Informat. Secur. Appl.*, 2013, pp. 65–80.

[24] A. Mohaisen, P. Luo, Y. Li, Y. Kim, and Z.-L. Zhang, "Measuring bias in the mixing time of social graphs due to graph sampling," in *Proc. IEEE Military Commun. Conf.*, 2012, pp. 1–6.

[25] A. Nazi, Z. Zhou, S. Thirumuruganathan, N. Zhang, and G. Das, "Walk, not wait: Faster sampling over online social networks," *Proc. VLDB Endowment*, vol. 8, pp. 678–689, 2015.

[26] M. Papagelis, "Refining social graph connectivity via shortcut edge addition," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 2, pp. 1–35, 2015.

[27] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multidimensional random walks," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 390–403.

[28] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 4292–4293.

[29] K. Sigman, "Time-reversible markov chains," *Lecture Notes Stochastic Model. I*, pp. 1–4, 2009. [Online]. Available: http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-Time-Reversibility.pdf

[30] M. Trusov, R. E. Bucklin, and K. Pauwels, "Effects of word-of-mouth versus traditional marketing: Findings from an internet social networking site," *J. Marketing*, vol. 73, no. 5, pp. 90–102, 2009.

[31] X. Wang, R. T. Ma, Y. Xu, and Z. Li, "Sampling online social networks via heterogeneous statistics," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2587–2595.

[32] X. Xu, C. H. Lee, and D. Y. Eun, "A general framework of hybrid graph sampling for complex network analysis," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 2795–2803.

[33] X. Xin and C.-H. Lee, "Challenging the limits: Sampling online social networks with cost constraints," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[34] J. Zhao, J. Lui, D. Towsley, P. Wang, and X. Guan, "A tale of three graphs: Sampling design on hybrid social-affiliation networks," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 939–950.

[35] Z. Zhou, N. Zhang, and G. Das, "Leveraging history for faster sampling of online social networks," *Very Large Data Bases*, vol. 8, no. 10, pp. 1034–1045, 2015.

[36] Z. Zhou, N. Zhang, Z. Gong, and G. Das, "Faster random walks by rewiring online social networks on-the-fly," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 769–780.

Rui Wang received the BEng degree in computer science from Donghua University and the PhD degree in computer science from the University of Science and Technology of China under the supervision of Prof. Yinlong Xu and Associate Prof. Yongkun Li. She is currently a postdoctor with the College of Computer Science, Zhejiang University. Her research interests include graph computing systems and storage systems.

Yongkun Li (Member, IEEE) received the BEng degree in computer science from the University of Science and Technology of China in 2008, and the PhD degree in computer science and engineering from The Chinese University of Hong Kong in 2012. He is currently an associate professor with the School of Computer Science and Technology, University of Science and Technology of China. After that, he worked as a postdoctoral fellow in Institute of Network Coding with The Chinese University of Hong Kong. His research mainly focuses on memory and file systems support for data computing, including key-value systems, distributed file systems, graph computing systems, as well as memory and I/O optimization for virtualized systems, and cloud computing. He is a member of the CCF/ACM, and he is USTC tang scholar.

Shuai Lin received the BEng degree from the University of Electronic Science and Technology of China. He is currently working toward the PhD degree with the School of Computer Science and Technology, University of Science and Technology of China. His main research interests are graph systems.

Weijie Wu received the BSc degree in electronics from Peking University, Beijing, and the PhD degree in computer science from The Chinese University of Hong Kong. In this research, he worked as an independent researcher. He is also a principal software engineer with Roblox Corporation, CA, USA. Before that, he held research or faculty positions with ZeroEx Inc., Huawei Technologies, Shanghai Jiao Tong University, and National University of Singapore. His current research interests include modeling and performance evaluation of computer networks, resource allocation in networked systems, and network economics.

Hong Xie (Member, IEEE) received the BEng degree in computer science from the University of Science and Technology of China and the PhD degree in computer science from The Chinese University of Hong Kong proudly under the supervision of Prof. John. C.S. Lui. He is currently a researcher with the College of Computer Science, Chongqing University. His research interests include online learning and data science. He is a member of the ACM.

Yinlong Xu received the BS degree in mathematics from Peking University in 1983, and the MS and PhD degrees in computer science from the University of Science and Technology of China (USTC) in 1989 and 2004, respectively. He is currently a professor with the School of Computer Science and Technology, USTC, and is leading a research group in doing some networking and high performance computing research. His research interests include network coding, storage systems, combinatorial optimization, design and analysis of parallel algorithms, parallel programming tools, etc. He received the Excellent PhD Advisor Award of Chinese Academy of Sciences in 2006.

John C. S. Lui (Fellow, IEEE) received the PhD degree in computer science from UCLA, 1992. He is currently the Choh-Ming Li chair professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong (CUHK), Hong Kong. He was the chairman of the Department from 2005 to 2011. His current research interests include communication networks, network/system security (e.g., cloud security, mobile security, etc.), network economics, network sciences, machine learning theories, large-scale distributed systems, and performance evaluation theory. He serves on the editorial board of the *IEEE/ACM Transactions on Networking*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, the *Journal of Performance Evaluation* and the *International Journal of Network Security*. He received various departmental teaching awards and the CUHK vice-chancellor's Exemplary Teaching Award. He is also a co-recipient of the IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 Best Student Paper Awards. He is a fellow of the Association for Computing Machinery (ACM), a croucher senior research fellow, a fellow of the Hong Kong Academy of Science Engineering (HKAES) and an elected member of the IFIP WG 7.3.