2019 IEEE 35th International Conference on Data Engineering (ICDE)

# Walking with Perception: Efficient Random Walk Sampling via Common Neighbor Awareness

Yongkun Li[1], Zhiyong Wu[1], Shuai Lin[1], Hong Xie[2], Min Lv[1], Yinlong Xu[1], John C.S. Lui[3]
[1]*School of Computer Science and Technology, University of Science and Technology of China*
[2]*College of Computer Science, Chongqing University*
[3]*Department of Computer Science and Engineering, The Chinese University of Hong Kong*
ykli@ustc.edu.cn, {wzylucky, shuailin}@mail.ustc.edu.cn,
hongx87@gmail.com, {lvmin05, ylxu}@ustc.edu.cn, cslui@cse.cuhk.edu.hk

***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. Experimental results on real-world network datasets show that our approach converges remarkably faster than the state-of-the-art random walk sampling algorithms. Furthermore, to achieve the same estimation accuracy, our approach reduces the query cost (a measure of sampling budget) significantly. Lastly, we also use two case studies to demonstrate the effectiveness of our sampling framework in solving large-scale graph analysis tasks.**

## I. 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 gain 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. This can be further validated by the following two examples.

- **Investment on networking platforms.** OSNs are proven to be effective for viral marketing due to the "word-of-mouth" effect [14], [31], [7]. 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.
- **Bundling strategy in viral marketing.** Bundling sale which bundles multiple products together to sale with some discount can be witnessed everywhere in our dairy 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 an easy task to accurately estimate graph centralities or efficiently solve graph mining problems. The challenge mainly comes from two aspects. First, OSNs are usually extremely large (e.g., the number of monthly active users of Facebook has already reached two billions [27]). Second, to protect user privacy, many OSNs only allow the third-party agents to access the networking data through fixed API interfaces with rate constraint. These challenges raise a fundamental question: *How to design computationally efficient algorithms for large-scale OSNs?* Graph sampling is a promising paradigm to address the computational challenge of graph analysis tasks, since it generates representative samples of the OSNs without traversing the whole network and has received extensive attentions [4], [16], [34], [33], [32].

Among various sampling approaches, random walk based method is the mainstream one due to its scalability and simplicity of implementation. The general idea of random walk based sampling scheme is as follows. 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,

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00090

and one can start collecting samples after convergence [37], [25], [9]. The time duration to reach the stationary distribution is known as the burn-in period [25], [24]. 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 problem (i.e., slow convergence) in real-world OSNs [37], [24]. This issue can lead to a large computation overhead. As a consequence, given a sampling budget (this is usually the case in real-world sampling tasks), we may only collect a small number of representative samples. Without sufficient samples, the accuracy of graph mining tasks may get severely reduced. 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 class of algorithms aim to increase the conductance of graphs [26], [23], while these schemes often need the global information of the graph, which is usually infeasible in practical situations. The second class of algorithms modify the transition probabilities in each walking step [18], [36], these schemes usually utilize the walking history and require only the 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 sampling?

In this paper, we propose a new random walk 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. 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 approach, 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 on sampled graphs.
- 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 [36]. Furthermore, to achieve the same estimation accuracy, CNARW can also reduce the query cost by up to 35.7%.

The rest of this paper is organized as follows. In Section II, we provide necessary background on random walk and graph sampling. In Section III, we present our common neighbor aware random walk framework (CNARW) and related theoretical analysis. In Section IV, we introduce the unbiased sampling scheme via CNARW. We evaluate the performance of CNARW in Section V, and review related works in Section VI. In Section VII, we conclude this paper.

## II. Preliminaries

### A. 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 number 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

$$P_{uv} = \begin{cases} 1/deg(u), & \text{if } v \in N(u), \\ 0, & \text{otherwise.} \end{cases}$$

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

$$\pi(u) = deg(u)(2|E|)^{-1}.$$

### B. Unbiased Graph Sampling

To perform unbiased graph sampling via random walks, 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 [21], [9], [18] and independent sampling [25], [37]. 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.

In the second step of unbiased graph sampling, that 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) [18], [13], 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 $f$ respect to $\boldsymbol{\pi}$.

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

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

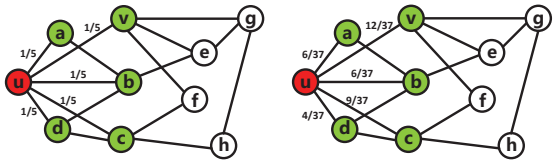
## III. CNARW: Common Neighbor Aware Random Walk

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. Main Idea of CNARW

The main idea of CNARW 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 Figure 1). Instead of choosing the next-hop node uniformly at random from all candidates as in SRW, which we call *uniform walking* (see Figure 1(a)), 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 Figure 1(b)). 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 Figure 1(b), $P_{uv} = 12/37$ 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 Figure 1(b) 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 should converge faster.




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

Figure 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.

### B. 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.

**Formulation of next node selection.** To formulate next node selection, 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*

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

*where $\bar{C} = V - C$, $E_{C,\bar{C}} = \{(u,v) \in E | u \in C, v \in \bar{C}\}$, and $Vol(C) = \sum_{u \in C} deg(u)$.*

**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 Figure 1(a), $S = \{u, a, b, v, c, d\}$. 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 $S_{-v} = S \backslash \{v\}$. For example, as in Figure 1(a), $S_{-v} = \{u, a, b, c, d\}$. 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 $S$, say $\Delta\phi_v$, can be derived as*

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

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

**Proof:** We derive $\Delta\phi_v$ as follows.

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

**Remark:** From Theorem 2, we can see that $\Delta\phi_v$ is only dependent on $deg(v)$ and $C_{uv}$. In particular, for 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 larger degree implies 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 is consistent with the intuition behind the weighted walking strategy described in Section III-A. In summary, for each $v \subset N(u)$, its contribution to the conductance of the frontier set can be taken as an effective indicator to evaluate the goodness of choosing it as the next node in each step. 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 according to Theorem (2), the intuitive strategy is to make the transition probability from $u$ to $v$ (i.e., $P_{uv}$) be proportional to $\Delta\phi_v$. For example, 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 less 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. To design a reversible random walk so as to easily derive the stationary distribution, we also guarantee the symmetric property in designing $P_{uv}$. Mathematically, we make $P_{uv}$ be proportional to $1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$ as follows.

$$P_{uv} \propto 1 - C_{uv}/\min\{deg(u), deg(v)\}. \tag{2}$$

We point out that one can also adopt other functions in Equation (2) 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, the function $\min$ is able to eliminate the dominating effect of $deg(u)$ when it is very large so that different neighbors of $u$ can be differentiated. Thus, using $\min$ usually leads to better performance, which is also validated in Section V-F via experiments.

To realize the proportional strategy in Equation (2) and limit the computation overhead, CNARW adopts a walking-with-rejection policy to determine the next-hop node in each walking step. In particular, in each step, CNARW first selects a candidate node from $N(u)$ uniformly at random, say $v \in N(u)$ is selected, but it only accepts $v$ as the next node with probability $q_{uv}$, which is defined as $q_{uv} = 1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$, where $C_{uv}$ is the number of common neighbors between $u$ and $v$. Note that if $v$ is rejected, which will happen with probability $1 - q_{uv}$, the walker repeats the selection again by checking another randomly selected node from $N(u)$. Note that with the walking-with-rejection policy, we access only node $v$ if it is accepted, but not have to access all nodes in $N(u)$, and thus reduce the computation overhead in each step.

Note that it has a chance of walking back to the currently visiting node as $\tilde{p}_{uu} = 1 - \sum_{v \in N(u)} \frac{1}{deg(u)} \times (1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}) > 0$. Since returning back to the currently visiting node also introduces overhead and slows down the convergence, to avoid backtracking, we do a normalization on

the transition probabilities. In summary, the transition matrix, $\boldsymbol{P} = [P_{uv}]_{u,v\in V}$, can be written as

$$P_{uv} = \begin{cases} \tilde{p}_{uv}/(1-\tilde{p}_{uu}), & \text{if } v \in N(u), \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

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

$$\tilde{p}_{uv} = \begin{cases} \frac{1}{deg(u)} \times (1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}), & \text{if } v \in N(u), \\ 1 - \sum_{k\in N(u)} \tilde{p}_{uk}, & \text{if } v = u, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We also take Figure 1 as an example to illustrate the walking process in one step. Note that $N(u) = \{a, b, v, c, d\}$ as shown in Figure 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 higher probability, which is consistent with the intuition that $v$ possesses as a better choice of 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 $v$;

**Remark**: We point out that our CNARW stated in Algorithm 1 only utilizes the information of the current node. Clearly, we can extend CNARW to utilize more historical information by taking into consideration more previously visited nodes. Specifically, we extend the algorithm in Section III-D and evaluate its performance via experiments in Section V-E

### C. Analysis of Stationary Distribution

To guarantee the effectiveness of CNARW, we provide theoretical analysis to show that CNARW has a unique stationary distribution in Theorem 3, and we also derive the probability distribution of each node and each edge being visited in Theorem 4 and Theorem 5.

**Theorem** *3: 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** *4: The stationary distribution $\boldsymbol{\pi}$ of CNARW satisfies the following condition: for any $u, v \in V$, $\pi(u)/\pi(v) = [deg(u)(1 - \tilde{p}_{uu})]/[deg(v)(1 - \tilde{p}_{vv})]$, so 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 [30], we only need to show that the following equation has a unique solution.

$$\pi(u) \times P_{uv} = \pi(v) \times P_{vu}. \tag{5}$$

Based on Equation (3)-(4), 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 Equation (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** *5: After CNARW 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 of an CNARW. 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

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

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 use the definition of static distribution [15] to prove that the static probability of the expanded Markov chain for edge $(i, j)$ is $\frac{1}{Z} \times (1 - \frac{C_{ij}}{\min\{d_i, d_j\}})$ through two steps.
**Step 1.** Prove $\sum_{i\in V} \sum_{j\in N(i)} \pi(e_{ij}) = 1$.

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

**Step 2.** Prove $\pi(e_{ij}) = \sum_{k\subset V} \pi(e_{ki})P'(e_{ki}, e_{ij})$.

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

Summarize the above two steps, we finish the proof. ∎

### *D.* Extension of CNARW to Utilize More Visited Nodes

Note that the algorithm CNARW in Algorithm 1 only uses the information of current node, that is, 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 $S=N(x_H)\cup N(x_{H-1})\cup\cdots\cup N(x_2)\cup N(u)$, 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 Equation (1), we can get the following result:

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

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:

$$P_{uv}^H=\begin{cases}\frac{\tilde{p}_{uv}^H}{1-\tilde{p}_{uu}^H}, & \text{if } v\in N(u),\\ 0, & \text{otherwise,}\end{cases}\tag{9}$$

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

$$\tilde{p}_{uv}^H=\begin{cases}\frac{1}{deg(u)}\times(1-\frac{C_{Sv}}{\min\{|S|,deg(v)\}}), & \text{if } v\in N(u),\\ 1-\sum_{k\in N(u)}\tilde{p}_{uk}^H, & \text{if } v=u,\\ 0, & \text{otherwise.}\end{cases}\tag{10}$$

Now the complete random walk algorithm by taking use of $H$ previously visited nodes is stated in Algorithm 2.

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 in Section V-E. 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 V-E, leveraging only the current node already takes most of the benefit of speeding up the convergence. That is, the marginal benefit of considering more historical information becomes very small.

**Algorithm 2:** 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 $v$;

## IV. 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.

**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 II-B), we set the weight $w(u)$ as follows:

$$w(u)=\gamma(u)/deg(u),\quad \text{where}\quad \gamma(u)=1/(1-\tilde{p}_{uu}).\tag{11}$$

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

**Theorem** *6: 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 *of $f$ over the samples, which we denote as $\mu(f)$, can be derived as follows:*

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

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

$$\frac{\sum_{u\in R}\frac{\gamma(u)}{deg(u)}f(u)}{\sum_{u\in R}\frac{\gamma(u)}{deg(u)}}=\frac{\sum_{u\in R}\frac{1/|V|}{Z\times deg(u)\times(1-\tilde{p}_{uu})}f(u)}{\sum_{u\in R}\frac{1/|V|}{Z\times deg(u)\times(1-\tilde{p}_{uu})}}$$
$$=\frac{\sum_{u\in R}\frac{U(u)}{\pi(u)}f(u)}{\sum_{u\in R}\frac{U(u)}{\pi(u)}}\to\frac{E_\pi(\frac{U(X)}{\pi(X)}f(X))}{E_\pi(\frac{U(X)}{\pi(X)})}=E_U(f),\quad a.s.$$ ∎

**Remark:** From Theorem 6, 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 needs the information of $u$'s neighbors (see Equation (4)). That is, directly computing $w(u)$ 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 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 3.

**Algorithm 3:** 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;
$sum_f \leftarrow 0; sum \leftarrow 0;$
**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 $\gamma(v)$;
  $w = \gamma(v)$ / $deg(v)$;
  $sum_f += w \times f(v)$; /* Aggregate Function $f$ */
  $sum += w$; $u \leftarrow v$;
**end**
Return $\mu(f) = sum_f$ / $sum$;

**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 $\omega$ should also be defined on edges, and we set the weight function $\omega(e_{uv})$ on edge $e_{uv}$ as follows.

$$w(e_{uv}) = 1/[1 - C_{uv}/\min\{d_u, d_v\}]. \tag{12}$$

Based on $w(e_{uv})$, we can also achieve an asymptotical unbiased edge sampling, which can be derived as follows.

**Theorem** *7: 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 follows:*

$$\mu(f) = \frac{\sum_{e_{uv} \in R} w(e_{uv}) f(e_{uv})}{\sum_{e_{uv} \in R} w(e_{uv})} \to E_U(f), \ a.s.$$

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

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

∎

In the following, we formally present the unbiased edge sampling algorithm in Algorithm 4.

**Algorithm 4:** 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 $\mu(f) = sum_f$ / $sum$;

## V. 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.

### A. Datasets & Experiment Setup

We conduct experiments on the datasets released by Leskovec *et al.* [20] and Rossi *et al.* [29]. We present some simple statistics of these datasets in Table I. 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], [37], [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, CaGaQc, and Phy1, which are used to study mixing rate which is computationally expensive for large-scale 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 |

Table I
SUMMARY OF DATASETS

We compare our algorithm with three typical random walk sampling algorithms: (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**) [36], 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.

### *B.* Performance Metrics

**Estimation error and query cost.** A fundamental tradeoff of sampling algorithms is: *estimation error* v.s. *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 in a sampling task, including the queries in both burn-in period and sampling phase:

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

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 is a reasonable cost metric, and it is also widely used to evaluate the efficiency of sampling algorithms [36], [37], [25]. The reason why we consider only unique queries is that once a node is visited, we can store its associated local information, and thus when it is visited again, we do not need to query the graph 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.

**Mixing time.** Note that query cost or estimation error is heavily dependent on the convergence speed of the random walk. In particular, if a random walk sampling algorithm (RWSA) converges fast, then with the same query budget, it can generate more representative samples, which leads to a smaller estimation error. 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. Computing the SLEM is computational expensive and not scalable to large-scale graphs. Thus, we study SLEM on small-scale datasets so as to gain some fundamental understandings on the convergence speed of our algorithm.

**Convergence to mean.** Besides, we further evaluate the convergence speed on large-scale graphs. We adopt 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

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

It is also computationally expensive to compute the exact value of $T_{\text{cm}}$. Thus, we estimate $T_{\text{cm}}$ 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., $T_{\text{cm}}$, we repeat the simulation for $n$ times to obtain $n$ samples $T_{\text{cm}}^1, \ldots, T_{\text{cm}}^n$, and study both the mean and standard deviation. Mathematically, we evaluate the average convergence rate by the following metric:

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

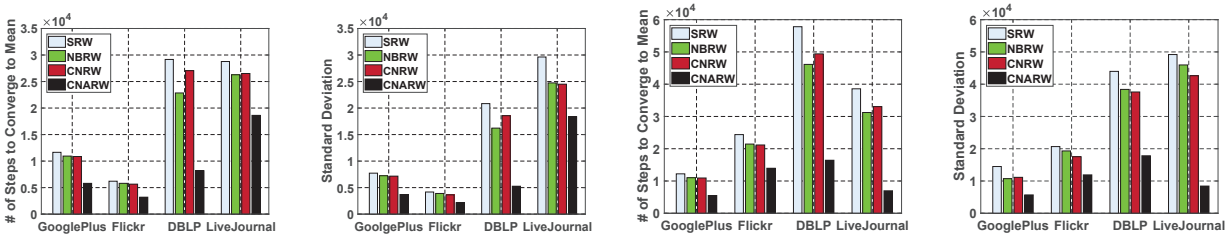
We apply standard deviation (SD) to quantify the variation of convergence rate, and we use the following estimator of SD:

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

To summarize, we will evaluate the sampling algorithms in five aspects, i.e., relative error, query cost, average and standard deviation of convergence to mean (i.e., $\bar{T}_{\text{cm}}$ and $\sigma(T_{\text{cm}})$), and convergence to stationary distribution (i.e., SLEM).

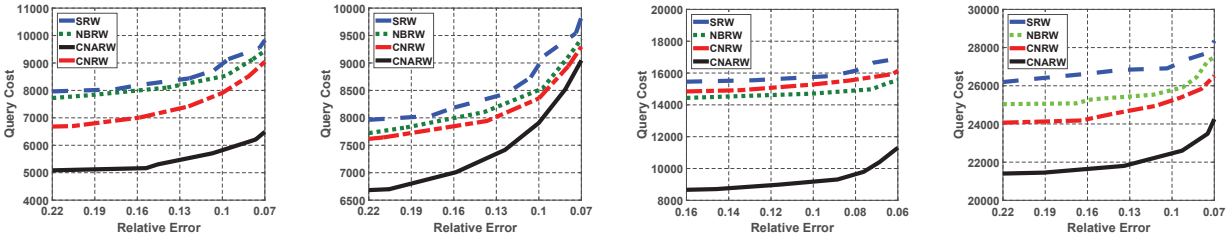
### *C.* Convergence Speed

We first evaluate the convergence speed for general sampling tasks by studying the convergence to stationary distribution, which is characterized by the second largest eigenvalue modulus (SLEM) of the transition matrix. Since it is very expensive to compute SLEM, we consider three small-scale social networks listed in Table I. Besides, since the closed-form transition matrices of NBRW and CNRW are hard to derive, we only compare our CNARW with SRW. Table II 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.




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

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




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

Figure 3. Tradeoff between estimation error and query cost.

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

Table II
THE SLEM FOR SRW AND CNARW: SMALLER SLEM MEANS FASTER CONVERGENCE SPEED.

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 the convergence to mean, which is related to a specific estimation task. In particular, we take the average node degree and average node pair similarity as indicators. Figure 5(a) and 5(b) presents the minimum number of steps needed to converge to mean $\bar{T}_{\rm cm}$ and its corresponding standard deviation $\sigma(T_{\rm cm})$ respectively by taking node degree as the indicator of convergence. Figure 5(c) and 5(d) 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 $\bar{T}_{\rm cm}$ and $\sigma(T_{\rm cm})$ is estimated by using 300 runs of an algorithm. From Figure 2 we observe that our algorithm CNARW has a smaller $\bar{T}_{\rm cm}$ 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 $\bar{T}_{\rm cm}$ 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, Figure 2 also shows that our CNARW has a smaller standard deviation on $\bar{T}_{\rm cm}$ 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.

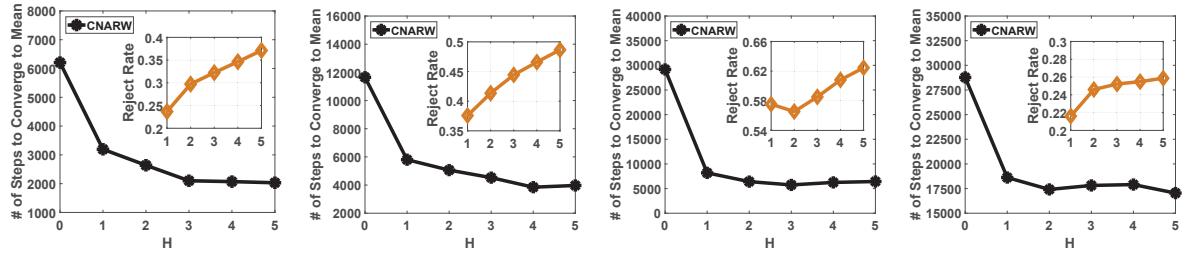
### D. Estimation Error v.s. Query Cost

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 four sampling algorithms (i.e., SRW, NBRW, CNRW, and our algorithm 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.

Figure 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 Figure 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%).

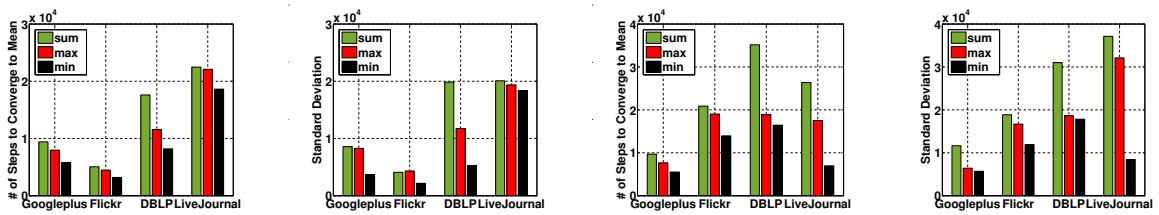
### E. Tradeoff of Using More History Information

Our experimental results thus far consider one historical neighbor for our CNARW. We also run experiments to further show the impact of $H$ on the convergence speed. Figure 4 presents the speed of convergence to mean (i.e., $\bar{T}_{\rm cm}$) and the rejection rate when $H$ varies from 0 to 5. We observe that $\bar{T}_{\rm cm}$




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

Figure 4. Impact of utilizing $H$ previously visited nodes.




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

Figure 5. Convergence speed under various design choices of the transition matrix. Note that sum, max, and min denote the cases of using $deg(u) + deg(v)$, $\max\{deg(u), deg(v)\}$, and $\min\{deg(u), deg(v)\}$ in Equation (2), respectively, and CNARW uses min which has the best performance.

decreases significantly when $H$ increases from 0 to 1. This implies that we can speed up the convergence with CNARW 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 (or the acceleration of the convergence) is not significant any more when we consider more than one visited node. Besides, the rejection rate in each walking step may also increase for large $H$, which may introduce larger query cost as each walking step may access more nodes. Therefore, we conclude that it is good enough to consider one visited node only, just like CNARW.

### *F.* Impact of Transition Matrix Design

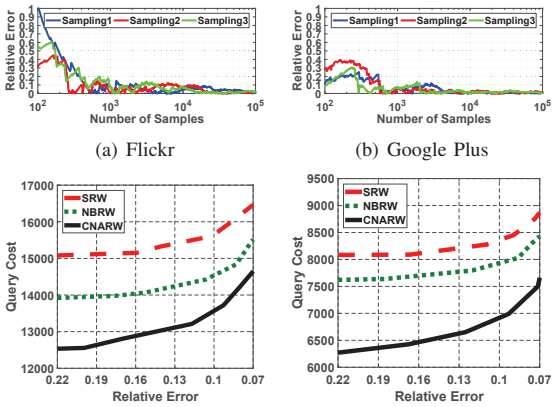
As analyzed in Section III-B, the rationale behind the design of the transition matrix in CNARW is trying to maximize the conductance of the frontier set, so we design the transition probability $P_{uv}$ by making it proportional to $1 - \frac{C_{uv}}{\min\{deg(u), deg(v)\}}$ as stated in Equation (2). Clearly, it is also flexible to consider other function forms in the transition matrix design. In this subsection, we evaluate the performance of using other two function forms to demonstrate the efficiency of CNARW. Specifically, we study the cases of using $deg(u) + deg(v)$ and $\max\{deg(u), deg(v)\}$ instead of $\min\{deg(u), deg(v)\}$ as in CNARW. Note that all these function forms satisfy the symmetric property, so they can be adapted to the same analysis framework. Figure 5 shows the convergence speed with different transition matrix designs. The convergence speed is measured by the average and standard deviation (SD) of the convergence to mean (i.e., $\bar{T}_{\text{cm}}$ and $\sigma(T_{\text{cm}})$). We also consider to use both node degree and node pair similarity as the indicator. From the results, we can see that using the function $\min$ as in CNARW always has the best performance. For example, using the sum function needs 50%-2.8$\times$ more steps to converge on average when using node pair similarity as indicator. The reason why $\min$ performs better is that it can eliminate the dominating effect of $deg(u)$ when it is very large. That is, when $deg(u)$ is very large, using $\min$ can still differentiate different neighbors of $u$ very well, so CNARW can always choose a better node to walk.

### *G.* Applications of CNARW

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

**Application 1: Investment on networking platforms.** As discussed in Section I, 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 4. According to the convergence rate of CNARW in Figure 5(c), 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 Figure 6(a) and Figure 6(b). 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 Figure 6(c) and Figure 6(d), 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 Figure 6(c) and Figure 6(d), we observe that CNARW requires a smaller query cost to achieve the same estimation accuracy, which implies a higher efficiency.




(a) Flickr (b) Google Plus

(c) Flickr (d) Google Plus

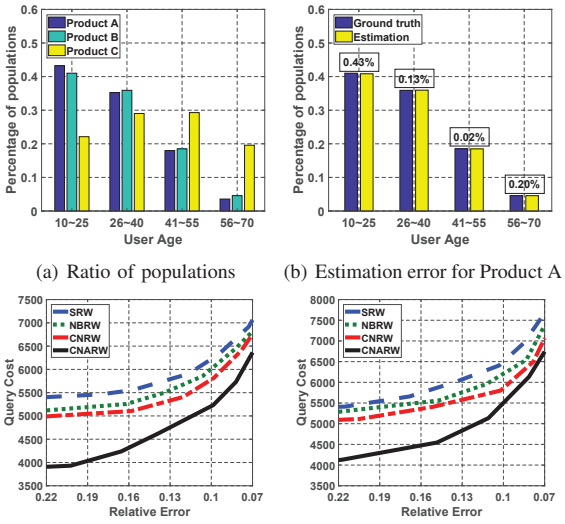
Figure 6. Performance of CNARW in Application 1.

**Application 2: Bundling strategy in viral marketing.** As discussed in section I, 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 III. 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 Figure 7(a) shows the ground truth for the ratio of populations in different age groups who are interested in each product. From Figure 7(a), one can observe that, product A and B should be sold as a bundle, since they 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 3. Figure 7(b) shows that the estimated distribution of user interests using our CNARW's is very close to the ground truth, implying high accuracy. Figure 7(c) and Figure 7(d) 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

| 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 |

Table III
SUMMARY OF DATASETS. 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.




(a) Ratio of populations (b) Estimation error for Product A

(c) % of users who are interested in A for the 10∼25 age group (d) % of users who are interested in A for the 26∼40 age group

Figure 7. Performance of CNARW in Application 2.

CNARW requires much less query cost than the stat-of-the-art sampling algorithms.

## VI. Related Works

Traditional sampling methods, e.g., random node sampling, random edge sampling and random subgraph sampling all need the knowledge of global graph topology [19]. To relax this constraint, graph sampling schemes via crawling techniques have become popular. These approaches include breadth-first search (BFS) and depth-first search (DFS), as well as random walk based approach. Even though BFS and DFS are simple, they are hard to derive the sampling probability [17]. Thus, random walk sampling has become the mainstream and also been widely used, e.g., [9], [21], [37], [18], [36].

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.* [33] 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.* [28] proposed a multidimensional random walk and Zhao *et al.* [35] 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. However, uniform samples are computationally expensive to obtain in OSNs. Recently, Zhou *et al.* [37], [36] proposed to speed up the convergence by utilizing the walking history, which was first utilized in non-backtracking random walk (NBRW) [18]. In particular, they constructed a "virtual" overlay network from the walking history to guide the walker in [37], and later, they proposed the Circulated Neighbors random walk (CNRW) by constructing a higher-ordered Markov chain [36].

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.

## VII. 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.

## Acknowledgments

The work is supported in part by National Key R&D Program of China under Grant No. 2018YFB1003204, and National Nature Science Foundation of China under Grant No. 61772484, 61772486, 61672486. The work by John C.S. Lui is supported in part by GRF 14208816.

## References

[1] K. Avrachenkov, B. Ribeiro, and D. Towsley. Improving random walk estimation accuracy with uniform restarts. In *WAW*, pages 98–109, 2010.
[2] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 46(4):667–689, 2004.
[3] X. Chen, Y. Li, P. Wang, and J. Lui. A general framework for estimating graphlet statistics via random walk. *VLDB*, 10(3):253–264, 2016.
[4] F. Chiericetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós. On sampling nodes in a network. In *WWW*, 2016.
[5] G. G. Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010.
[6] M. K. Cowles and B. P. Carlin. Markov chain monte carlo convergence diagnostics: a comparative review. *JASA*, 91(434):883–904, 1996.
[7] A. De Bruyn and G. L. Lilien. A multi-stage model of word-of-mouth influence through viral marketing. *International Journal of Research in Marketing*, 25(3):151–163, 2008.
[8] J. Geweke et al. *Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments*, volume 196. Federal Reserve Bank of Minneapolis, Research Department Minneapolis, 1991.
[9] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Infocom*, 2010.
[10] O. Häggström. *Finite Markov chains and algorithmic applications*, volume 52. Cambridge University Press, 2002.
[11] M. H. Hansen and W. N. Hurwitz. On the theory of sampling from finite populations. *The Ann. of Math. Stat.*, 14(4):333–362, 1943.
[12] L. Jin, Y. Chen, P. Hui, C. Ding, T. Wang, A. V. Vasilakos, B. Deng, and X. Li. Albatross sampling: robust and effective hybrid vertex sampling for social graphs. In *MobiArch, 2011*, 2011.
[13] G. L. Jones et al. On the markov chain central limit theorem. *Probability surveys*, 1(299-320):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. *Journal of Interactive Marketing*, 28(3):167–183, 2014.
[15] T. Konstantopoulos. Introductory lecture notes on markov chains and random walks. *Department of Mathematics, Uppsala University*, 200(9), 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. In *SIGMETRICS*, 2011.
[17] M. Kurant, A. Markopoulou, and P. Thiran. Towards unbiased bfs sampling. *J-SAC*, 29(9):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. In *SIGMETRICS*, volume 40, pages 319–330, 2012.
[19] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *SIGKDD, 2006*, 2006.
[20] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.
[21] R.-H. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin. On random walk based graph sampling. In *ICDE, 2015*, 2015.
[22] L. Lovász. Random Walks on Graphs: A Survey, 1993.
[23] A. Mohaisen and S. Hollenbeck. Improving social network-based sybil defenses by rewiring and augmenting social graphs. In *WISA, 2013*, 2013.
[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 *MILCOM, 2012*, 2012.
[25] A. Nazi, Z. Zhou, S. Thirumuruganathan, N. Zhang, and G. Das. Walk, not wait: Faster sampling over online social networks. *VLDB*, 2015.
[26] M. Papagelis. Refining social graph connectivity via shortcut edge addition. *TKDD*, 10(2):12, 2015.
[27] Recode. *Facebook now has two billion monthly users. https://www.recode.net/2017/6/27/15880734/facebook-mark-zuckerberg-two-billion-monthly-users*. 2017.
[28] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *IMC, 2010*, 2010.
[29] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.
[30] K. Sigman. Time-reversible Markov Chains. http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-Time-Reversibility.pdf, 2009.
[31] M. Trusov, R. E. Bucklin, and K. Pauwels. Effects of word-of-mouth versus traditional marketing: findings from an internet social networking site. *Journal of marketing*, 73(5):90–102, 2009.
[32] X. Wang, R. T. Ma, Y. Xu, and Z. Li. Sampling online social networks via heterogeneous statistics. In *IEEE INFOCOM*, 2015.
[33] X. Xu, C. H. Lee, and D. Y. Eun. A general framework of hybrid graph sampling for complex network analysis. In *IEEE INFOCOM*, 2014.
[34] X. Xu, C.-H. Lee, and Y. Eun. Challenging the limits: Sampling online social networks with cost constraints. 2017.
[35] J. Zhao, J. Lui, D. Towsley, P. Wang, and X. Guan. A tale of three graphs: Sampling design on hybrid social-affiliation networks. In *ICDE, 2015*, 2015.
[36] Z. Zhou, N. Zhang, and G. Das. Leveraging history for faster sampling of online social networks. *VLDB*, 8(10):1034–1045, 2015.
[37] Z. Zhou, N. Zhang, Z. Gong, and G. Das. Faster random walks by rewiring online social networks on-the-fly. In *ICDE, 2013*, 2013.