A Unified Framework to Estimate Global and Local Graphlet Counts for Streaming Graphs

Xiaowei Chen, John C.S. Lui
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email: {xwchen, cslui}@cse.cuhk.edu.hk

***Abstract*—Counting small connected subgraph patterns called graphlets is emerging as a powerful tool for exploring topological structure of networks and for analysis of roles of individual nodes. Graphlets have numerous applications ranging from biology to network science. Computing graphlet counts for "dynamic graphs" is highly challenging due to the streaming nature of the input, sheer size of the graphs, and superlinear time complexity of the problem. Few practical results are known under the massive streaming graphs setting. In this work, we propose a "unified framework" to estimate the graphlet counts of the whole graph as well as the graphlet counts of individual nodes under the streaming graph setting. Our framework subsumes previous methods and provides more flexible and accurate estimation of the graphlet counts. We propose a general unbiased estimator which can be applied to any k-node graphlets. Furthermore, efficient implementation is provided for the 3, 4-node graphlets. We perform detailed empirical study on real-world graphs, and show that our framework produces estimation of graphlet count for streaming graphs with 1.7 to 170.8 times smaller error compared with other state-of-the-art methods. Our framework also achieves high accuracy on the estimation of graphlets for each individual node which previous works could not achieve.**

I. Introduction

Graphlets are defined as connected subgraph patterns in networks [1]. Counting graphlets is emerging as a powerful tool for exploring the topological structure of networks and for analysis of roles of individual nodes. Many applications rely on couting graphlets to provide insightful characterization of global and local structures of the networks. Applications of graphlets in biology include protein prediction [2], cancer gene identification [3], and network alignment [4]. In network science, graphlets have been applied for spam detection [5], anomaly detection [6], [7], graph classification [8], [9] and node classification [10].

Counting graphlets includes computing the occurance of graphlets in the whole graph, namely *global graphlet counts*, and computing the number of graphlets adjacent to each node in the graph, namely *local graphlet counts*. Despite the importance of graphlets in various application domains, exact computation of global and local graphlet counts is often infeasible due to the massive size of the input graph and the superlinear time complexity of the problem [9], [11]–[14]. It is natural in these cases to resort to *efficient-to-compute* estimation of these quantities. Moreover, many important networks are dynamic and node/edge additions are coming in as a stream, e.g., adding a new user in online social networks, paper publications in the collaboration networks, phone call history in the communication networks, etc. *Our goal is to compute accurate estimate of global and local graphlets for the online analysis of the input streaming graphs.*

Many previous contributions focus on global and local triangle counting in streaming graphs [6], [7], [15]–[18]. However, these methods are difficult to extend for general graphlet counting. Also, there are only few works on estimating the global graphlet counts [19]–[21] from a set of uniformly sampled edges (which is within the stream setting we mentioned). However, these methods are inefficient for the real time tracking of graphlet counts. More importantly, none of existing methods can be applied for computation of the local graphlet counts.

Summary of contributions. In this work, we propose a "*unified framework*" to estimate any k-node global and local graphlet counts for any streaming graphs, where $k \geq 3$. Our contributions are:

- **Unified Framework.** We propose a unified framework to estimate, at any time instant, the graphlet counts of the whole graph (i.e., global graphlet counts) as well as the graphlet counts of individual nodes (i.e., local graphlet counts) under the streaming setting. Our framework *subsumes* previous methods and also provides more accurate global graphlet count estimation by choosing appropriate sampling schemes and counting strategies. We are also the first to propose an efficient method to compute the local graphlet counts under the streaming setting, which is a more challenging problem than the computation of global graphlet counts.
- **Unbiased Estimator.** We present a general unbiased estimator whose unbiasedness is rigorously proved. The estimator can be applied for any k-node graphlets and different sampling schemes as well counting strategies.
- **Efficient Implementation.** We provide an efficient implementation for the $3, 4$-node graphlet estimation. Moreover, our implementation can be directly applied to exact counting of global and local graphlets for the streaming graphs.
- **Extensive Experiments.** We conduct extensive experiments on the real-world networks to validate the efficiency of our framework. We explore the effects of different sampling schemes and counting strategies. For the global graphlets, our framework outperforms state-of-the-art methods, e.g., the estimation produced by methods in our framework has 1.7 to 170.8 times smaller error than prior arts, and achieves

high accuracy in estimation of local graphlet counts.

Paper organization. We introduce the problem and review the related work in Sect. II. The unified framework is presented and analyzed in Sect. III. We demonstrate an efficient implementation of $3,4$-node graphlet counting in Sect. IV. Sect. V reports the results of the extensive experiments. The conclusion is drawn in Sect. VI. Additional materials can be found in the Appendix.

II. Preliminaries

A. Notations and Definitions

The Streaming Model. Let $G^{(t)} = (V^{(t)}, E^{(t)})$ be the graph observed up to time t. Without loss of generality, we assume $V^{(0)} = \emptyset$ and $E^{(0)} = \emptyset$. Let e_t be the new edge arriving from the stream at time t. At time $t+1$ ($t \geq 0$), the graph $G^{(t+1)} = (V^{(t+1)}, E^{(t+1)})$ is obtained by inserting edge $e_{t+1} = (u, v)$ into the graph $G^{(t)}$, i.e., $V^{(t+1)} = V^{(t)} \cup \{u, v\}$ and $E^{(t+1)} = E^{(t)} \cup \{e_{t+1}\}$. We assume edges are distinct and arrive from the stream in any arbitrary order.

Subgraphs. For a graph $G = (V, E)$, we say $G_k = (V_k, E_k)$ is a *subgraph* of G if $V_k \subseteq V$ and $E_k \subseteq E$, written as $G_k \subseteq G$. If $G_k \subseteq G$ and G_k contains all edges $(u, v) \in E$ where $u, v \in V_k$, then G_k is an *induced subgraph* of G. *We differentiate between subgraphs and induced subgraphs.* In general, if we do not say "*induced*", we mean the usual subgraph setting (a subset of edges).

Isomorphic. $G = (V, E)$ and $G' = (V', E')$ are isomorphic, written as $G \simeq G'$, if there exists a bijection $\varphi : V \to V'$ with $(u, v) \in E \Leftrightarrow (\varphi(u), \varphi(v)) \in E'$ for all $u, v \in V$. Intuitively, two graphs are isomorphic if they have the same appearance when one removes all node and edge labels of them.

Graphlets. Graphlets are defined as *connected subgraph patterns*. Fig. 1 depicts all 3 and 4-node graphlets. The set of all k-node graphlets is denoted as $\mathcal{G}^k$ and we denote g_i^k as the ith type of k-node graphlets. The *graphlet count* C_i^k is defined as the number of *induced subgraphs* that are isomorphic to (i.e., have the same appearance as) the graphlet g_i^k in the graph. Fig. 2 shows an example of graphlet counts $C_1^4, \cdots, C_6^4$. The C_1^4 equals to 2, meaning that there are two induced subgraphs isomorphic to g_1^4 (line of length 3) in the graph G, which are induced by nodes $\{1, 2, 3, 4\}$ and $\{1, 5, 4, 3\}$.

Orbits. Nodes in the same graphlet may have different "*positions*". For example, the node with label 8 in g_4^4 (Fig. 1) acts as a hub while the node with label 6 is a periphery node. As one can see, they play different roles in the network. Formally, nodes in a graphlet are partitioned into a set of automorphism groups called *orbits* [22]. Two nodes in the graphlet belong to the same orbit if they are mapped to each other in some projection of the graphlet onto itself. Fig. 1 illustrates that nodes in the 3-node graphlets g_1^3, g_2^3 and 4-node graphlets $g_1^4, \cdots, g_6^4$ are partitioned into 3 and 11 different orbits respectively. We denote the orbit with label j in k-node graphlets as o_j^k. The set of all orbits in the k-node graphlets is defined as $\mathcal{O}^k$. The *orbit count* $O_j^k(v)$ is defined as the number of *induced subgraphs* where the node v occupies the position of the orbit o_j^k. For example, as shown in Fig. 2, $O_1^4(1)$ equals to 2 since node 1 occupies the orbit o_1^4 in two subgraphs induced by node sets $\{1, 2, 3, 4\}$ and $\{1, 5, 4, 3\}$.

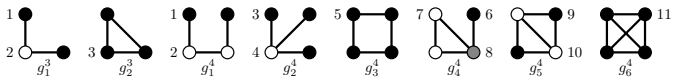


Fig. 1: 3-node and 4-node graphlets and their orbits. Nodes with the same color belong to the same orbit within that graphlet. E.g., the nodes in g_4^4 have three different orbits: 6, 7 and 8, and are marked with different colors and labels. Nodes with white color are symmetric and belong to the same orbit 7. Node with gray belongs to orbit 8, while node in black belongs to orbit 6.

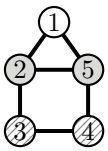
Problem Definition. Given the graphlet size k, our goal is to compute the following two structural metrics for the streaming graph $G^{(t)} = (V^{(t)}, E^{(t)})$ at any discrete time t:

1) **Global graphlet counts (GGC).** For the graph $G^{(t)}$, the global graphlet counts are defined as the set of graphlet counts C_i^k for all the graphlets in $\mathcal{G}^k$.
2) **Local graphlet counts (LGC).** For each node v in the graph $G^{(t)}$, the local graphlet counts for node v are defined as the set of orbit counts $O_j^k(v)$ for all the orbits in $\mathcal{O}^k$.

Fig. 2 gives an example of global/local graphlet counts. The GGC carries significant information about the structural properties of the graph, e.g., GGC can be used as a structural feature to classify large graphs [8], [23]. The LGC is a fine-grained description of the local topological structure for every node. It maps nodes to vectors, which can be applied for protein function prediction in biology network [2], predicting the economic attributes of a country in the world trade network [24] and node classification in social networks [10].

B. Related Work

Exact Graphlet Counting for Static Graphs. Counting exact number of graphlets has high computation cost since the number of possible k-node graphlets grows in $O(|V|^k)$, hence, the $3, 4, 5$-node graphlet counting attracts more attention than the general k-node graphlet counting. Shervashidze et al. [8, Theorem 7] proved that counting GCC can be achieved in $O(|V| \cdot \Delta^k)$ with $k = 3, 4, 5$ where Δ is the maximum degree of the graph. Triangle, which is one of the 3-node graphlets, has been extensively studied. The most efficient algorithm is proposed by [11]. The method is based on matrix multiplication and has time complexity $O(|E|^{1.41})$. However, it also has a high space complexity $O(|V|^2)$, which restricts it applicability for large graphs. A more practical algorithm "*edge-iterator*" proposed in [12] counts triangles with time complexity $O(|E|^{1.5})$ with a reasonable space requirement. Ahmed et al. proposed an efficient parallel algorithm to compute the 4-node GCC [9]. The main idea of is to count a few graphlets for each edge in parallel and then derive the exact counts for others in constant time by leveraging



G

C_1^4	C_2^4	C_3^4	C_4^4	C_5^4	C_6^4
2	0	1	2	0	0

Global graphlet counts

v	$[O_1^4(v), \cdots, O_{11}^4(v)]$
1	$[2,0,0,0,0,0,2,0,0,0,0]$
2	$[0,1,0,0,1,0,1,1,0,0,0]$
3	$[1,1,0,0,1,1,0,0,0,0,0]$
4	$[1,1,0,0,1,1,0,0,0,0,0]$
5	$[0,1,0,0,1,0,1,1,0,0,0]$

Local graphlet counts

Fig. 2: Example of global/local graphlet counts.

the combinatorial equations between graphlets. For 5-node GCC, Pinar et al. proposed an efficient algorithmic framework by cutting graphlets into smaller ones, and using counts of smaller graphlets to obtain larger counts [14]. Methods in [9], [14] are the state-of-the-art to compute GGC. Hočevar and Demšar present a combinatorial method for counting LGC up to five nodes in [13]. The method reduces the complexity by building a system of equations that connect counts of orbits and compute all orbit counts by enumerating a single one.

Sampling Methods. Many previous contributions focus on triangle (g_2^3) counting in streaming graphs. For example, Tsourakakis et al. [15] presented a triangle estimating method by sampling a set of edges from a graph, and it requires one pass to the graph data and can be applied to the streaming graph directly. Ahmed et al. [25] proposed a more general sampling scheme "*sample and hold*" to estimate various graph properties such as triangle count and clustering coefficient. Pavan et al. [17] and Jha et al. [18] estimated the global triangle counts by sampling a set of connected path of length two. Recently, more efficient methods based on sampling a set of edges are proposed in [6], [7], [16]. However, all these works focus on triangle counting and are difficult to be extended for GGC when $k > 3$.

There are also few works on the GGC when $k > 3$. Elenberg et al. proposed a method to estimate the $3,4$-node GGC from a set of independently sampled edges in [19] and [20]. Wang et al. [21] showed how to estimate any k-node GGC from the set of independently sampled edges. The main idea of these methods is to construct an one-step Markov transition matrix to model the relationship between GGC in the sampled graphs and the original graphs. However, the results produced by these methods are not efficient enough and they can not be applied for LGC estimation.

III. A Unified Framework

We present a unified framework to estimate GGC and LGC in this section. The framework is summarized in Algo. 1. Line 4, 8, 9 in the algorithm describe the random process to maintain the sample graph G_S. The function DETERMINESAMPLE decides whether to sample the incoming edge e_t at time t and the function UPDATESAMPLE is the procedure to update G_S with the selected edge e_t. Line 5, 6, 7 describe the counting process to maintain the unbiased estimate of graphlet counts. The function DETERMINECOUNT describes our counting strategy. It decides whether to update the graphlet counts. The unbiased estimate of global and local graphlet counts is updated with the sample graph G_S and edge e_t in the function

TABLE I: Relationship between previous contributions and our framework.

Methods	Sampling Schemes	Counting Strategies	Graphlets	Global	Local
DOULION [26]	Uniform	Dependent	Triangle	✓	✗
MASCOT [6]	Uniform	Independent $q = 1$	Triangle	✓	✓
TRIÈST [16]	Reservoir	Independent $q = 1$	Triangle	✓	✓
Elenberg et al. [19]	Uniform	Dependent	$\mathcal{G}^3$	✓	✗
Elenberg et al. [20]	Uniform	Dependent	$\mathcal{G}^4$	✓	✗
MINFER [21]	Uniform	Dependent	$\mathcal{G}^k$	✓	✗
Our framework	Both	Both	$\mathcal{G}^k$	✓	✓

Algorithm 1 The unified framework

1: **Initialization:** Sample graph $G_S = (V_S, E_S)$ with $V_S = \emptyset, E_S = \emptyset$, graphlet size k, $\hat{\boldsymbol{C}}^k \leftarrow \mathbf{0}, t \leftarrow 1$
2: **for** edge e_t in the stream $\mathcal{S}$ **do**
3: $\quad count \leftarrow$ false, $sample \leftarrow$ false
4: $\quad$ DETERMINESAMPLE$(sample, t)$
5: $\quad$ DETERMINECOUNT$(count, sample)$
6: $\quad$ **if** $count$ is true **then** ▷ update counter
7: $\quad\quad$ UPDATECOUNTER$(\hat{\boldsymbol{C}}^k, G_S, e_t, t)$
8: $\quad$ **if** $sample$ is true **then** ▷ update sample graph
9: $\quad\quad$ UPDATESAMPLE(G_S, e_t)
10: $\quad t \leftarrow t + 1$

UPDATECOUNTER. We emphasize that our framework is very "*flexible*" since one can choose among different sampling schemes and counting strategies. The framework is *unified* since it subsumes a broad range of previous methods [6], [16], [19]–[21], [26]. Table I summarizes previous works and our framework. It shows that these state-of-the-art methods are simply special cases under our framework.

Notation. Let $G_S = (V_S, E_S)$ denote the sample graph in Algo. 1. $G_S \cup \{e_t\}$ denotes the graph formed by $E_S \cup \{e_t\}$.

A. Maintain Sample Graphs

In this subsection, we show two commonly used schemes to maintain the sample graphs: (1) uniform sampling and (2) reservoir sampling [27].

Uniform Sampling. For each edge (say e_t) in the stream, we sample and include it to be part of the sample graph G_S with probability p ($0 < p \leq 1$), or ignore it with probability $1 - p$. Algo. 2 shows the details of this sampling scheme.

Reservoir Sampling. The standard reservoir sampling [27] maintains the edge samples E_S with a fix sized reservoir. When edge e_t arrives at time t, if t is no larger than the reservoir size M, we insert the edge e_t into E_S deterministically. When $t > M$, we sample the edge e_t with probability M/t. If the edge e_t is chosen, we remove an edge from E_S uniformly at random to make space for the newly selected edge e_t. The detailed procedure is described in Algo. 3.

Lemma 1 (Inclusion probability)**.** *Let $\xi_{m,t}$ denote the probability that an arbitrary subgraph $B \subseteq G^{(t)}$ with m edges is*

Algorithm 2 Uniform sampling of stream edges

Parameters: sampling probability p
function DETERMINESAMPLE($sample, t$)
  if $Bernoulli(p) == 1$ **then**
    $sample \leftarrow$ true
function UPDATESAMPLE($G_S, e_t = (u, v)$)
  $V_S \leftarrow V_S \cup \{u, v\}$
  $E_S \leftarrow E_S \cup \{e_t\}$

Algorithm 3 Reservoir sampling of stream edges

Parameters: reservoir size M
function DETERMINESAMPLE($sample, t$)
  if $t \leq M$ or ($t > M$ and $Bernoulli(\frac{M}{t})$ == 1) **then**
    $sample \leftarrow$ true
function UPDATESAMPLE($G_S, e_t = (u, v)$)
  if $|E_S| == M$ **then**
    remove a random edge from E_S
    remove nodes with zero degree from V_S
  $V_S \leftarrow V_S \cup \{u, v\}$
  $E_S \leftarrow E_S \cup \{e_t\}$

included in G_S at the end of time t. If G_S is maintained by uniform sampling with sampling probability p, we have

$$\xi_{m,t} \triangleq \Pr(B \subseteq G_S) = p^m \quad \textbf{(1)}$$

and if G_S is updated by reservoir sampling with reservoir size M, we have

$$\xi_{m,t} \triangleq \Pr(B \subseteq G_S) = \begin{cases} 0 & \text{if } m > M \\ 1 & \text{if } m \leq M, \text{ and } t \leq M \\ \binom{t-m}{M-m} / \binom{t}{M} & \text{if } m \leq M, \text{ and } t > M \end{cases} \quad \textbf{(2)}$$

In practice, the reservoir size M satisfies $M \gg m$. Hence $\xi_{m,t} > 0$ in general. Eq. (1) in Lemma 1 can be proved directly by the fact that each edge on the stream is selected with probability p independently. Eq. (2) is a trivial extension of the property that any subset A of size M in $E^{(t)}$ satisfies $\Pr(A = E_S) = 1/\binom{t}{M}$ at time t if $t > M$ [27]. Note that uniform sampling only inserts edges into G_S while reservoir sampling removes an existing edge to make space for the newly sampled edge if the reservoir is already full. Consequently, reservoir sampling makes a constant consumption of the memory regardless of the length of the stream.

B. Counting Strategies

We propose the following two counting strategies for our framework. One counting strategy is to update the graphlet counts upon the edge e_t is being sampled (Line 1-2 in Algo. 4). The second counting strategy is to decouple the events of sampling and counting, i.e., we update the graphlet counts with a probability q_t ($0 < q_t \leq 1$), which is independent from whether the edge e_t is sampled or not (Line 3-6 in Algo. 4). These two strategies are referred to as the "*dependent counting strategy*" and "*independent counting strategy*" respectively.

The advantage of the dependent counting strategy is that it makes 50% less calls to the random number generator $Bernoulli$. The advantage of the independent counting strategy is that the counting probability q_t is a tunable parameter.

Algorithm 4 Counting Strategies

▷The dependent counting strategy
1: **function** DETERMINECOUNT($count, sample$)
2: $count \leftarrow sample$
▷The independent counting strategy
3: **Parameters:** graphlet count updating probability q_t
4: **function** DETERMINECOUNT($count, sample$)
5: **if** $Bernoulli(q_t) == 1$ **then**
6: $count \leftarrow$ true

Algorithm 5 Update Global/Local Graphlet Counts

1: **function** UPDATECOUNTER($\hat{\boldsymbol{C}}^k, G_S, e_t, t$)
2: **if** count global graphlets **then**
3: $\boldsymbol{d}'(e_t) \leftarrow$ COUNTGLOBALSUBGRAPH(G_S, e_t)
4: **if** count local graphlets **then**
5: $\boldsymbol{d}'(e_t) \leftarrow$ COUNTLOCALSUBGRAPH(G_S, e_t)
6: $\hat{\boldsymbol{C}}^k \leftarrow \hat{\boldsymbol{C}}^k + \boldsymbol{A}^{-1}\boldsymbol{P}_t^{-1}\boldsymbol{d}'(e_t)$ ▷ update counter

For example, if G_S keeps a large fraction of edges and the edges arrive at a fast speed, one can decrease q_t to speed up the updating of graphlet counts. If we have limited memory to keep G_S, we can increase q_t to improve the accuracy of the estimation. In the extreme case where $q_t = 1$, we call the function UPDATECOUNTER deterministically to update graphlet counts whenever edge e_t arrives from the stream.

Counting probability Let p_t be the probability that we update the graphlet counts at time t ($t > 0$). p_t equals to

$$p_t = \begin{cases} p & \text{dependent counting (uniform sampling)} \\ \min\{1, \frac{M}{t}\} & \text{dependent counting (reservoir sampling)} \\ q_t & \text{independent counting} \end{cases} \quad \textbf{(3)}$$

C. Update Graphlet Counts

Let $\boldsymbol{C}^k$ denote the exact GGC/LGC matrix. For GGC, $\boldsymbol{C}^k$ is defined as a $|\mathcal{G}^k| \times 1$ matrix, i.e., $\boldsymbol{C}^k \triangleq [C_1^k, \cdots, C_{|\mathcal{G}^k|}^k]^T$. For LGC, $\boldsymbol{C}^k$ is defined as a $|\mathcal{O}^k| \times |V^{(t)}|$ matrix, i.e., $\boldsymbol{C}^k \triangleq [\boldsymbol{O}^k(v_1), \cdots, \boldsymbol{O}^k(v_{|V^{(t)}|})], v_i \in V^{(t)}$, here $\boldsymbol{O}^k(v_i) \triangleq [O_1^k(v_i), \cdots, O_{|\mathcal{O}^k|}^k(v_i)]^T$. *Our goal is to provide an unbiased estimate of $\boldsymbol{C}^k$ for $G^{(t)}$ at any time instant*. Algo. 5 shows the details of the function UPDATECOUNTER, which outputs an unbiased estimation of $\boldsymbol{C}^k$ at time t. The function first computes the number of subgraphs containing edge e_t in $G_S \cup \{e_t\}$ (Line 2-5) and then update the graphlet counts with equation in Line 6. The detailed implementation of COUNTGLOBALSUBGRAPH and COUNTLOCALSUBGRAPH in Algo. 5 will be illustrated in the next section. The unbiased estimator in Line 6 is derived based on the following key observation which relates the subgraphs in $G^{(t)}$ and sample graph G_S.

Observation 1. *Let B be a subgraph in $G^{(t)}$ formed by m edges $\{e_{t_1}, \cdots, e_{t_{m-1}}, e_t\}$, i.e., e_t is the edge in B that appears last on the stream. Let δ_B be a random variable that takes value 1 if subgraph B is counted at the end of time t, and 0 otherwise. Then we have $\Pr(\delta_B = 1) = p_t \xi_{m-1,t-1}$.*

The above observation can be proved by the fact that $\delta_B = 1$ if and only if $\{e_{t_1}, \cdots, e_{t_{m-1}}\}$ is already in E_S at time

TABLE II: The probability matrix $\boldsymbol{P}_t$ for the 3-node graphlets.

Global Graphlet Counts	Local Graphlet Counts
$\begin{array}{c} \\ g_1^3 \\ g_2^3 \end{array}\begin{array}{c} \begin{array}{cc} g_1^3 & g_2^3 \end{array} \\ \begin{bmatrix} p_t\xi_{1,t-1} & 0 \\ 0 & p_t\xi_{2,t-1} \end{bmatrix} \end{array}$	$\begin{array}{c} \\ o_1^3 \\ o_2^3 \\ o_3^3 \end{array}\begin{array}{c} \begin{array}{ccc} o_1^3 & o_2^3 & o_3^3 \end{array} \\ \begin{bmatrix} p_t\xi_{1,t-1} & 0 & 0 \\ 0 & p_t\xi_{1,t-1} & 0 \\ 0 & 0 & p_t\xi_{2,t-1} \end{bmatrix} \end{array}$

$t-1$ and we choose to update the graphlet counts at time t. It indicates a simple linear equation between the number of subgraphs in $G^{(t)}$ and G_S at time $t=T$, which is

$$\sum_{t=1}^{T} \boldsymbol{d}(e_t) = \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{P}_t^{-1}\boldsymbol{d}'(e_t)\right] \tag{4}$$

The proof of Eq. (4) is in the appendix. Intuitively, Eq. (4) is an extension of $\mathbb{E}[\delta_B] = \Pr(\delta_B = 1) = p_t\xi_{m-1,t-1}$. The notations $\boldsymbol{P}_t, \boldsymbol{d}(e_t), \boldsymbol{d}'(e_t)$ are explained as follows.

• **Probability matrix $\boldsymbol{P}_t$.** It is a *diagonal matrix*. We assume the subgraph B in Obs. 1 is isomorphic to graphlet g_i^k and a node v in B occupies the orbit o_j^k. When estimating GGC (resp. LGC), the i-th (resp. j-th) diagonal entry $P_t(i,i)$ (resp. $P_t(j,j)$) equals to $\Pr(\delta_B = 1) = p_t\xi_{m-1,t-1}$. Table II shows an example of the probability matrix for the 3-node graphlets. Note that $\boldsymbol{P}_t$ is *invertible* since $P_t(i,i) = p_t\xi_{m-1,t-1} > 0$.

• **Global (or local) subgraph count matrix $\boldsymbol{d}(e_t)$.** For the GGC, the i-th entry of the $|\mathcal{G}^k| \times 1$ matrix $\boldsymbol{d}(e_t)$ denotes the number of *subgraphs* in $G^{(t)}$ that contain edge e_t and are isomorphic to graphlet g_i^k. Let $\boldsymbol{d}_v(e_t)$ denote the local *subgraph count vector* of v in $G^{(t)}$. The i-th entry of $\boldsymbol{d}_v(e_t)$ is the number of subgraphs containing edge e_t in $G^{(t)}$ where node v occupies the orbit o_i^k. For the LGC, $\boldsymbol{d}(e_t)$ is defined as the *local subgraph count matrix*, which is written as $[\cdots, \boldsymbol{d}_{v_i}(e_t), \cdots], (v_i \in V^{(t)})$. $\boldsymbol{d}'(e_t)$ is the corresponding global (or local) subgraph count matrix in $G_S \cup \{e_t\}$.

To derive the unbiased estimator, we still need to construct the relationship between $\boldsymbol{C}^k$ and $\sum_{t=1}^{T} \boldsymbol{d}(e_t)$. We introduce a projection matrix $\boldsymbol{A}$ for this purpose.

• **Projection matrix $\boldsymbol{A}$.** With a slight abuse of notations, we define $A(g_i^k, g_j^k)$ as the number of distinct copies of the graphlet g_i^k in the graphlet g_j^k. Assume node v occupies the orbit o_j^k in the k-node graph B. Then $A(o_i^k, o_j^k)$ is defined as the number of subgraphs of B where node v occupies orbit o_i^k. When estimating GGC (resp. LGC), we let $A(i,j) \triangleq A(g_i^k, g_j^k)$ (resp. $A(i,j) \triangleq A(o_i^k, o_j^k)$). Fig. 3 gives an example of computing $\boldsymbol{A}$. Table III shows the projection matrix for 3, 4-node GGC/LGC. Computing $\boldsymbol{A}$ is not a big concern since $\boldsymbol{A}$ is independent of the input graphs and can be computed in advance. Let m_i^k be the number of edges in g_i^k. W.l.o.g, we let $m_i^k \geq m_j^k$ when $i > j$. Then $A(i,j) = 0$ when $i > j$, and $A(i,i) = 1$. Hence, $\boldsymbol{A}$ is *upper triangular*. Lemma 2 shows how matrix $\boldsymbol{A}$ connects $\boldsymbol{C}^k$ and $\sum_{t=1}^{T} \boldsymbol{d}(e_t)$. The proof of Lemma 2 is in the appendix.

TABLE III: Projection matrix $\boldsymbol{A}$ for 3, 4-node subgraphs.

$k=3$, Global

$$\boldsymbol{A} = \begin{array}{c} \\ g_1^3 \\ g_2^3 \end{array}\begin{array}{c} \begin{array}{cc} g_1^3 & g_2^3 \end{array} \\ \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} \end{array}$$

$k=3$, Local

$$\boldsymbol{A} = \begin{array}{c} \\ o_1^3 \\ o_2^3 \\ o_3^3 \end{array}\begin{array}{c} \begin{array}{ccc} o_1^3 & o_2^3 & o_3^3 \end{array} \\ \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

$k=4$, Global

$$\mathbf{A} = \begin{array}{c} \\ g_1^4 \\ g_2^4 \\ g_3^4 \\ g_4^4 \\ g_5^4 \\ g_6^4 \end{array}\begin{array}{c} \begin{array}{cccccc} g_1^4 & g_2^4 & g_3^4 & g_4^4 & g_5^4 & g_6^4 \end{array} \\ \begin{bmatrix} 1 & 0 & 4 & 2 & 6 & 12 \\ 0 & 1 & 0 & 1 & 2 & 4 \\ 0 & 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 & 4 & 12 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$k=4$, Local Graphlet Counts

$$\mathbf{A} = \begin{array}{c} \\ o_1^4 \\ o_2^4 \\ o_3^4 \\ o_4^4 \\ o_5^4 \\ o_6^4 \\ o_7^4 \\ o_8^4 \\ o_9^4 \\ o_{10}^4 \\ o_{11}^4 \end{array}\begin{array}{c} \begin{array}{ccccccccccc} o_1^4 & o_2^4 & o_3^4 & o_4^4 & o_5^4 & o_6^4 & o_7^4 & o_8^4 & o_9^4 & o_{10}^4 & o_{11}^4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & 2 & 1 & 0 & 4 & 2 & 6 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 2 & 2 & 4 & 6 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 2 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 2 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

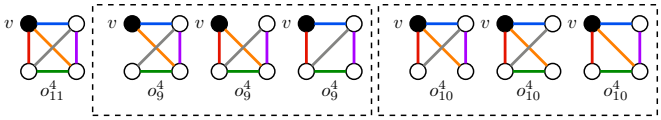


Fig. 3: Example of projection matrix $\boldsymbol{A}$. The clique (g_6^4) contains 6 distinct chordal cycles (g_5^4), hence $\boldsymbol{A}(g_5^4, g_6^4) = 6$. Node v occupies the orbit o_{11}^4 in the clique. For all the distinct chordal cycles in the clique, node v occupies orbit o_9^4 and o_{10}^4 in three of them respectively. Hence, $\boldsymbol{A}(o_9^4, o_{11}^4) = \boldsymbol{A}(o_{10}^4, o_{11}^4) = 3$.

Lemma 2. *Let $\boldsymbol{C}^k$ be the exact graphlet counts in $G^{(T)}$. We have*

$$\boldsymbol{A}\boldsymbol{C}^k = \sum_{t=1}^{T} \boldsymbol{d}(e_t) \tag{5}$$

Based on Eq. (4) and (5), we can easily derive the unbiased estimator for $\boldsymbol{C}^k$ in streaming graph $G^{(t)}$ at $t=T$, which is

$$\hat{\boldsymbol{C}}^k \triangleq \sum_{t=1}^{T} \boldsymbol{A}^{-1}\boldsymbol{P}_t^{-1}\boldsymbol{d}'(e_t). \tag{6}$$

Note that $\boldsymbol{A}$ is *invertible* since it is upper triangular. Thm. 1 proves the unbiasedness of Eq. (6).

Theorem 1. *The result produced by Algo. 5 is an unbiased estimate of the graphlet counts, i.e., $\boldsymbol{C}^k = \mathbb{E}[\hat{\boldsymbol{C}}^k]$.*

Proof. Since $\boldsymbol{A}\boldsymbol{C}^k = \sum_{t=1}^{T} \boldsymbol{d}(e_t)$ and $\sum_{t=1}^{T} \boldsymbol{d}(e_t) = \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{P}_t^{-1}\boldsymbol{d}'(e_t)\right]$, we have

$$\boldsymbol{C}^k = \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{A}^{-1}\boldsymbol{P}_t^{-1}\boldsymbol{d}'(e_t)\right]$$

One the other hand, $\hat{\boldsymbol{C}}^k = \sum_{t=1}^{T} \boldsymbol{A}^{-1}\boldsymbol{P}_t^{-1}\boldsymbol{d}'(e_t)$. Hence we have $\mathbb{E}[\hat{\boldsymbol{C}}^k] = \boldsymbol{C}^k$. ■

Discussion. Our framework can provide *both* global and local graphlet counts for general $\mathcal{G}^k$, for $k \geq 3$. Moreover, our

framework subsumes several state-of-the-art methods. Note that when we use the uniform sampling scheme and dependent counting strategy, the probability matrix $\boldsymbol{P}_t$ does not depend on the time t and can be written as $\boldsymbol{P}_t = \boldsymbol{P}$ where $P(i,i) = p_t\xi_{m-1,t-1} = p^m$. Besides, the i-th entry of $\sum_{t=1}^{T} \boldsymbol{d}'(e_t)$ equals to the number of subgraphs in G_S that are isomorphic to g_i^k. Here G_S is the sample graph of $G^{(T)}$. Let $\boldsymbol{C}_S^k$ denote the GGC of G_S. We have

$$\hat{\boldsymbol{C}}^k = \boldsymbol{A}^{-1}\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{C}_S^k.$$

The main idea of [19]–[21] is to compute the $\boldsymbol{C}_S^k$ and then correct the bias error using $\mathbb{E}[\boldsymbol{A}^{-1}\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{C}_S^k] = \boldsymbol{C}^k$. The matrix $\boldsymbol{A}^{-1}\boldsymbol{P}^{-1}\boldsymbol{A}$ is the so called "*one step Markov transition matrix*" in [19]–[21].

IV. Counting 3, 4-node Graphlets

In this section, we explain the implementation of functions CountGlobalSubgraph and CountLocalSubgraph in Algo. 5. To compute the number of general k-node subgraphs containing $e_t = (u,v)$ in $G_S \cup \{e_t\}$, one can apply the subgraph enumeration algorithm in [28] to the graph induced by u, v as well as nodes within distance no more than $k-2$ from u or v. Since $3,4$-node graphlets have many imporant applications [5], [9], [24], we propose a more computationally efficient global and local subgraph counting method for $k = 3, 4$. Note that our proposed counting algorithms can also be applied directly to the incremental dynamic graphs (i.e., adding new nodes/edges to graphs). In the following, $\mathcal{N}_S(u)$ denotes the set of neighbors of node u in G_S.

A. 3-node Graphlets

Algorithm for 3-node subgraph counting in presented in the Algo. 6. To count the global/local 3-node subgraphs containing edge $e_t = (u,v)$, function CountGlobalSubgraph/CountLocalSubgraph iterates through nodes in $\mathcal{N}_S(u)$ and $\mathcal{N}_S(v)$. Hence the time cost on counting subgraphs containing edge $e_t = (u,v)$ is $O(|\mathcal{N}_S(u)| + |\mathcal{N}_S(v)|)$. Note that only the estimations of local graphlet counts for nodes in $\mathcal{N}_S(u) \cup \mathcal{N}_S(v)$ are update when edge e_t arrives at time t.

Algorithm 6 Counting Global/Local 3-node Subgraphs

1: **function** CountGlobalSubgraph($G_S, e_t = (u,v)$)
2: $\quad d_1^3 \leftarrow |\mathcal{N}_S(u)| + |\mathcal{N}_S(v)|$
3: $\quad d_2^3 \leftarrow |\mathcal{N}_S(u) \cap \mathcal{N}_S(v)|$
4: $\quad$ **return** $\boldsymbol{d}' = [d_1^3, d_2^3]^T$
5: **function** CountLocalSubgraph($G_S, e_t = (u,v)$)
6: $\quad \boldsymbol{X} \leftarrow \boldsymbol{0}$, matrix $\boldsymbol{d}' \leftarrow \boldsymbol{0}$
7: $\quad$ **for each** $w \in \mathcal{N}_S(u)$ **do**
8: $\quad\quad d'(1,w) \mathrel{+}= 1, d'(2,u) \mathrel{+}= 1, d'(1,v) \mathrel{+}= 1$
9: $\quad\quad X(w) = 1$ ▷mark node w
10: $\quad$ **for each** $w \in \mathcal{N}_S(v)$ **do**
11: $\quad\quad d'(1,w) \mathrel{+}= 1, d'(1,u) \mathrel{+}= 1, d'(2,v) \mathrel{+}= 1$
12: $\quad\quad$ **if** $X(w) == 1$ **then**
13: $\quad\quad\quad d'(3,w) \mathrel{+}= 1, d'(3,u) \mathrel{+}= 1, d'(3,v) \mathrel{+}= 1$
14: $\quad$ **return** $\boldsymbol{d}'$

B. 4-node Global Subgraph Counting

Algo. 7 shows the 4-node global subgraph counting method. The 4-node local subgraph counting is presented in the Appendix B. The key of Algo. 7 is the equations in Line 17-22. Let us explain how to get these equations.

Let $\mathcal{S}(e_t)$ denote the set of subgraphs in $G_S \cup \{e_t\}$ that contain edge e_t. For the subgraphs in $\mathcal{S}(e_t)$ that are isomorphic to the same graphlet, edge e_t may occupy different positions. For example, as shown in Fig. 4, edge e_t can occupies two different positions, i.e., the center position () and the periphery position (), in the subgraphs isomorphic to g_1^4. Fig. 4 illustrates how to represent the counts of induced subgraphs that contain edge e_t with the variables $\boldsymbol{T}, \boldsymbol{S}_u, \boldsymbol{S}_v, N_{T,T}, N_{S,S}, N_{S,T}, N_2^4$. Since computing the induced subgraph counts is more intuitive than computing subgraph counts, we present the Lemma 3 which builds a relationship between the counts of induced subgraphs and normal subgraphs that containing edge e_t.

Lemma 3. *Let B be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}, e_t\}$, B' be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}\}$. Assume B is isomorphic to graphlet g_j^k. Let $d_i^k(B)$ denote the number of subgraphs in B that contain edge e_t and are isomorphic to graphlet g_i^k. We have*

$$d_i^k(B) = \begin{cases} A(i,j) - A(i,j') & \text{if } B' \simeq g_{j'}^k, \\ A(i,j) & \text{if } B' \text{ is disconnected.} \end{cases}$$

The proof of Lemma 3 is in Appendix A. Based on Lemma 3, equations in Line 17-22 can be computed systematically as shown in Eq. (7). Here $C(B)$ denotes the number of *induced subgraphs* that include e_t in the same position as B. The value $C(B)$ is presented in the right-hand side of Fig. 4.

$$\begin{bmatrix} N_1^4 \\ N_2^4 \\ N_3^4 \\ N_4^4 \\ N_5^4 \\ N_6^4 \end{bmatrix} = \boldsymbol{A} \times \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -C(\boxtimes) \\ C(\boxtimes) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -C(\boxtimes) \\ C(\boxtimes) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -C(\boxtimes) \\ 0 \\ C(\boxtimes) \\ 0 \end{bmatrix} + \begin{bmatrix} -C(\boxtimes) \\ 0 \\ C(\boxtimes) \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -C(\boxtimes) \\ 0 \\ 0 \\ C(\boxtimes) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ C(\boxtimes) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -C(\boxtimes) \\ 0 \\ C(\boxtimes) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ C(\boxtimes) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} C(\boxtimes) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} C(\boxtimes) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$
$$= \begin{bmatrix} 1 & 0 & 4 & 2 & 6 & 12 \\ 0 & 1 & 0 & 1 & 2 & 4 \\ 0 & 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 & 4 & 12 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -C(\boxtimes) - C(\boxtimes) + C(\boxtimes) + C(\boxtimes) \\ -C(\boxtimes) + C(\boxtimes) \\ -C(\boxtimes) + C(\boxtimes) \\ -C(\boxtimes) + C(\boxtimes) + C(\boxtimes) + C(\boxtimes) \\ -C(\boxtimes) + C(\boxtimes) + C(\boxtimes) \\ C(\boxtimes) \end{bmatrix} \quad (7)$$

Example. According to Eq. (7), N_5^4 in Algo. 7 (i.e., the count of subgraphs in $\mathcal{S}(e_t)$ that are isomorphic to g_5^4), equals to $5C(\boxtimes)+C(\boxtimes)+C(\boxtimes)$, where $C(\boxtimes)$, $C(\boxtimes)$, and $C(\boxtimes)$ denote the number of induced subgraphs that include e_t in the same positions as $\boxtimes$, $\boxtimes$, and $\boxtimes$ respectively. More specifically,

$$\begin{aligned} N_5^4 &= (A(5,6) - A(5,5)) \times C(\boxtimes) + C(\boxtimes) + C(\boxtimes) \\ &= 5 \times (T[2]) + (T[1] + T[3]) + (N_{T,T} - T[2]) \\ &= T[1] + 4T[2] + T[3] + N_{T,T}. \end{aligned}$$

Time Complexity. Let $\Gamma_l(u,v)$ represent the set of vertices in the graph $G_S \cup \{e_t\}$ with distance no more than l from node

Algorithm 7 Counting Global 4-node Subgraphs

function CountGlobalSubgraph($G_S, e_t = (u, v)$)
  $\boldsymbol{X} \leftarrow \boldsymbol{0}, \text{Star}_u = \emptyset, \text{Star}_v = \emptyset, \text{Tri}_e = \emptyset$
  for each $w \in \mathcal{N}_S(u)$ **do**
    Add w to Star_u and set $X(w) = 1$
  for each $w \in \mathcal{N}_S(v)$ **do**
    if $X(w) = 1$ **then**
      Add w to Tri_e and set $X(w) = 2$
      Remove w from Star_u
    else
      Add w to Star_v and set $X(w) = 3$
  $\boldsymbol{T} = \text{TriCount}(\boldsymbol{X}, \text{Tri}_e)$
  $\boldsymbol{S}_u = \text{StarCount}(\boldsymbol{X}, \text{Star}_u)$
  $\boldsymbol{S}_v = \text{StarCount}(\boldsymbol{X}, \text{Star}_v)$
  $T[2] = \frac{T[2]}{2}, S_u[1] = \frac{S_u[1]}{2}, S_v[3] = \frac{S_v[3]}{2}$
  $N_{T,T} = |\text{Tri}_e|(|\text{Tri}_e| - 1)/2,\ N_{S,S} = |\text{Star}_u||\text{Star}_v|$
  $N_{S,T} = (|\text{Star}_u| + |\text{Star}_v|)|\text{Tri}_e|$
  ▷ Get counts of 4-node subgraphs containing edge e_t
  $N_6^4 = T[2]$
  $N_5^4 = T[1] + 4T[2] + T[3] + N_{T,T}$
  $N_4^4 = T[0] + 2T[1] + 4T[2] + 2T[3] + S_u[1] + S_v[3] + 4N_{T,T} + N_{S,T}$
  $N_3^4 = T[1] + 2T[2] + T[3] + S_u[3]$
  $N_2^4 = \binom{|\mathcal{N}_S(u)|}{2} + \binom{|\mathcal{N}_S(v)|}{2}$
  $N_1^4 = 2T[0] + 3T[1] + 3T[3] + 4T[2] + S_u[0] + S_v[0] + 2S_u[1] + 2S_v[3] + 2S_u[3] + 2N_{T,T} + N_{S,S} + N_{S,T}$
  return $\boldsymbol{d}' = [N_1^4, \cdots, N_6^4]^T$

function TriCount($\boldsymbol{X}, \text{Tri}_e$)
  $\boldsymbol{T} = [0, 0, 0, 0]^T$
  for each $w \in \text{Tri}_e$ **do**
    for each $r \in \mathcal{N}_S(w) \backslash \{u, v\}$ **do**
      $T[X(r)]$ += 1
  return $\boldsymbol{T}$
function StarCount($\boldsymbol{X}, \text{Star}$)
  $\boldsymbol{S} = [0, 0, 0, 0]^T$
  for each $w \in \text{Star}$ **do**
    for each $r \in \mathcal{N}_S(w) \backslash \{u, v\}$ **do**
      $\boldsymbol{S}[X(r)]$ += 1
  return $\boldsymbol{S}$

TABLE IV: Datasets used in experiments.

Graph	Nodes	Edges	Description
Facebook [29]	63731	817035	A small subset of the total Facebook friendship graph
Epinion [30]	75879	405740	Who-trusts-whom network of Epinions.com
Slashdot [30]	77360	469180	The technology-related news website Slashdot from Nov. 2008
Gowalla [30]	196591	950327	A social network where users share the locations by checking-in
Wikitalk [30]	2394385	4659565	The network contains all the users and discussion from the inception of Wikipedia till Jan. 2008

u or v. To compute $\boldsymbol{d}'(e_t)$ where $e_t = (u, v)$, Algo. 7 iterates through all nodes in $\Gamma_2(u, v)$. The time cost on computing $\boldsymbol{d}'(e_t)$ is $O(|\Gamma_2(u, v)|)$.

Remarks. If we keep all the edges from the stream, our algorithms can be applied to compute the "*exact*" global/local graphlet counts of the streaming graph directly.

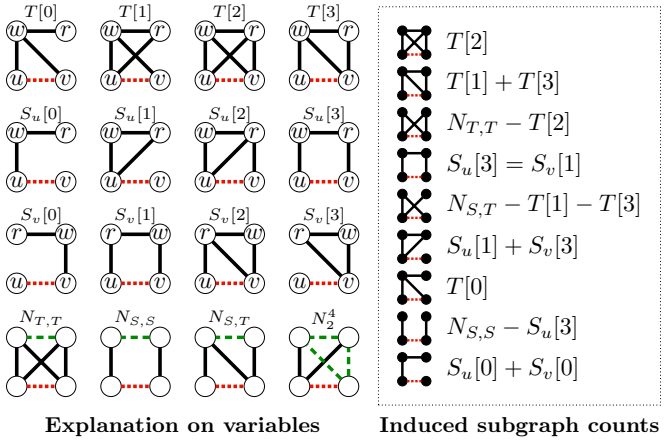


Fig. 4: Illustration of Algo. 7. The red dotted line represents the edge $e_t = (u, v)$ which arrives from the stream at time t. The green dashed lines represent edges that may exists. The left-hand side of the figure illustrates the meaning of variables $\boldsymbol{T}, \boldsymbol{S}_u, \boldsymbol{S}_v, N_{T,T}, N_{S,S}, N_{S,T}, N_2^4$. The first three rows demonstrate the computation process of TriCount and StarCount. The variables above the small figures denote the number of induced subgraphs in the sampled graph which include edge e_t in the same positions as the small figures. The right-hand side of the figure shows the count of induced subgraphs which include edge e_t in the specific positions. E.g., the number of induced subgraphs that include e_t as is $S_u[0] + S_v[0]$.

V. Experiments

A. Experimental Setup

We evaluate our framework on several real-world networks. For all graphs under study, we remove the directions, self-loops and duplicate edges. The streams are generated by permuting the edges in a random order. Table IV summarizes our datasets. We obtain the exact 3, 4-node GGC with the parallel algorithm in [9]. The exact 3, 4-node LGC are computed with the method in [13]. All the algorithms are implemented in C++ and ran on high performance computing servers equipped with 1.9GHz Intel Xeon CPUs.

Error Metrics. We use the following two metrics to measure the performance of our framework.

- *Relative Error* (RE): This measures how close the estimation of the GGC is to the ground truth. For the graphlet count C_i^k, the RE of the estimation $\hat{C}_i^k$ is defined as $\frac{|\hat{C}_i^k - C_i^k|}{C_i^k + 1}$. Here we add one to both C_i^k and $\hat{C}_i^k$ in case of $C_i^k = 0$.
- *Mean of Relative Error* (MRE): This measures how accurate is the estimation of LCC. For the orbit o_i^k, the MRE is defined as $\frac{1}{|V|}\sum_{v \in V} \frac{|\hat{O}_i^k(v) - O_i^k(v)|}{O_i^k(v) + 1}$, where $\hat{O}_i^k(v)$ is the unbiased estimation of $O_i^k(v)$. Similar to RE, we add one to both $\hat{O}_i^k(v)$ and $O_i^k(v)$ in case of $O_i^k(v) = 0$.

We report the average RE and MRE by running 100 independent simulations since our algorithm is randomized. In

the following, **US** and **RS** are short notations for uniform sampling and reservoir sampling respectively. **IC** and **DC** refer to independent counting and dependent counting respectively.

B. Estimation of Global Graphlet Counts

1) Effect of Sampling Schemes and Counting Strategies: In the following, we present how the sampling schemes and counting strategies may affect the performance of our framework with the 4-node graphlets. Due to the space limitation, in Fig. 5 and 6, we only report the accuracy of the estimation of C_2^4 and C_6^4 for graph Facebook. We choose C_2^4 and C_6^4 since  (g_2^4) and  (g_6^4) usually appears the most and the least frequently among all 4-node graphlets in the graphs.

Sampling Schemes. From Fig. 5 and 7, we observe that RS is more accurate than US with the same memory usage and counting strategies (or counting probability q_t). However, RS has longer processing time for each edge.

Counting Strategies. Compared with the dependent counting strategy, the independent counting strategy is *more flexible*. In Fig. 6, we compare different counting strategies with the same sampling scheme US. For a fair comparison, we set the counting probability of the IC strategy as $q_t = p$ for each simulation. By observing Fig. 6, we conclude that the IC strategy, which decouples the counting and edge sampling, *improves the accuracy* of US. Besides, the IC strategy has the same time cost as the DC strategy.

Counting Probability q_t. Fig. 7 illustrates the effect of q_t of the IC strategy. One can see that the counting probability q_t is a tunable parameter to control the trade-off between accuracy and update time. For the case where we want to keep a large fraction of edges, we can decrease q_t so to speed up the processing time for each edge.

2) Accuracy of the Framework: Based on above discussion, we know that when given the memory usage, RS with IC strategy ($q_t = 1$) has the highest accuracy in estimating GGC. Fig. 8 presents the relative error for all the datasets and all 3, 4-node graphlets, with a reservoir of size $0.1|E|$ and counting probability 1.0. The errors are below 8% in all instances. We conclude that our framework is capable of producing accurate estimation of GGC with a small amount of sampled edges.

3) Comparison with Previous Works: Fig. 9 shows the comparison results for all graphs in the datasets. For simplicity, we compare our framework with the previous methods [20], [21] (i.e., US scheme with DC strategy) by giving the same memory usage. We choose the method which can achieve the highest accuracy in our framework, i.e., RS and IC strategy ($q_t = 1$). We focus on 4-node graphlets since our framework subsumes and has the same performance as previous methods in estimating 3-node graphlet triangle. Fig. 9 shows that the method RS-1.0 has a much smaller error, i.e., $1.7\times \sim 170.8\times$ smaller relative error. In conclusion, our framework not only subsumes previous methods, but also outperforms these state-of-the-art methods with appropriate sampling schemes and counting strategies. More specifically, we propose a new state-of-the-art algorithm to compute the general k-node GGC.

C. Estimation of Local Graphlet Counts

In the following, we evaluate the performance of our framework in estimating the LGC. Due to space limitation, we only choose the uniform sampling. Reservoir sampling has similar performance. Also, we let the counting probability $q_t = 1.0$. In Fig. 11 and 12, we only choose orbits $o_2^3, o_3^3, o_{10}^4, o_{11}^4$ in graph Facebook for demonstration.

Convergence. Fig. 10 shows the MRE vs the sampling probability p in graph Facebook and Epinion. As we increase p, the MRE of all orbit counts decreases quickly, which means our proposed estimator converges to the actual LGC as the sampling probability (i.e., memory usage) increases.

Unbiasedness. We plot the scatter plot of the orbit counts vs degrees in Fig. 11. The figure shows that different orbits have different relations with the degrees, which can be further applied for anomaly detection [6], [7]. The average of the simulation results overlaps heavily with the ground truth value, which shows the unbiasedness of our proposed estimator.

Accuracy. Fig. 12 shows the scatter plot between the true LGC and its estimations for all nodes in Facebook in one simulation. Our algorithm is especially accurate for nodes with higher degrees. For these high degree nodes, the points are nearly along the line $y = x$. For low degree nodes, our points are balanced over $y = x$.

VI. Conclusion

We present a unified framework to estimate the local and global graphlet counts of the streaming graphs at any time instant. Our unified framework subsumes previous methods and provides more accurate estimation of global graphlet counts. It is also the first framework to provide an efficient estimation of the local graphlet counts. Both theoretical analysis and experimental evaluation validate the unbiasedness and efficiency of our proposed estimator. We show the superb performance of our framework in estimating $3, 4$-node local and global graphlets with extensive experiments on real-world graphs. For future work, we aim to develop and investigate other sampling schemes and counting strategies.

References

[1] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Science*, 2002.

[2] T. Milenkoviæ and N. Pržulj, "Uncovering biological network function via graphlet degree signatures," *Cancer Informatics*, 2008.

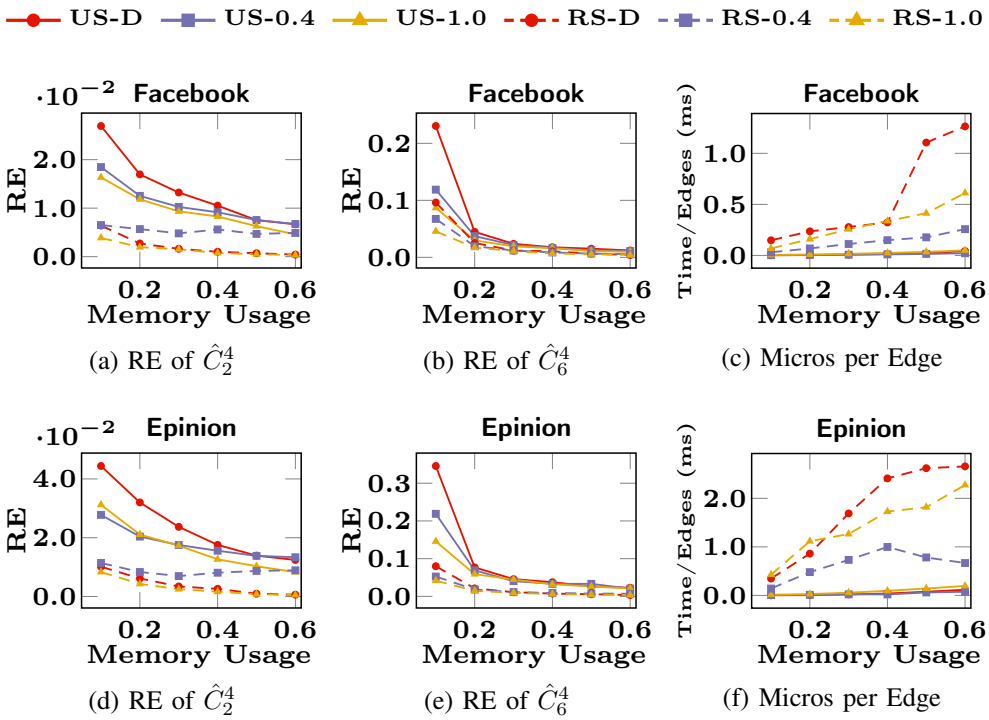
[3] T. Milenković, V. Memišević, A. K. Ganesan, and N. Pržulj, "Systems-level cancer gene identification from protein interaction network topology applied to melanogenesis-related functional genomics data," *J. R. Soc.*, 2010.

[4] F. E. Faisal, H. Zhao, and T. Milenković, "Global network alignment in the context of aging," *TCBB*, 2015.

[5] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *KDD'08*.

[6] Y. Lim and U. Kang, "MASCOT: Memory-efficient and accurate sampling for counting local triangles in graph streams," in *KDD'15*.

[7] M. Jung, S. Lee, Y. Lim, and U. Kang, "FURL: Fixed-memory and uncertainty reducing local triangle counting for graph streams," *to appear in WWW'17*.



(a) RE of $\hat{C}_2^4$ (b) RE of $\hat{C}_6^4$ (c) Micros per Edge

(d) RE of $\hat{C}_2^4$ (e) RE of $\hat{C}_6^4$ (f) Micros per Edge

Fig. 5: Trade-off between the memory usage vs the accuracy and running time. The ticks on the x-axis denote the ratio of sampled edges kept in the memory. The character "D"/number appended after the sampling schemes RS and US refers to the dependent counting scheme or parameter q_t of the independent counting.

[8] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *AISTATS*, 2009.

[9] N. K. Ahmed, J. Neville, R. Rossi, N. G. Duffield, and T. L. Willke, "Fast parallel graphlet counting for large networks," in *ICDM'15*.

[10] M. Fang, J. Yin, X. Zhu, and C. Zhang, "Trgraph: Cross-network transfer learning via common signature subgraphs," *TKDE*, 2015.

[11] N. Alon, R. Yuster, and U. Zwick, "Finding and counting given length cycles," *Algorithmica*, 1997.

[12] T. Schank and D. Wagner, "Finding, counting and listing all triangles in large graphs, an experimental study," in *WEA*, 2005.

[13] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, 2014.

[14] A. Pinar, C. Seshadhri, and V. Vishal, "ESCAPE: Efficiently counting all 5-vertex subgraphs," *to appear in WWW'17*.

[15] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller, "Triangle sparsifiers," *JGAA*, 2011.

[16] L. De Stefani, A. Epasto, M. Riondato, and E. Upfal, "TRIÈST: Counting local and global triangles in fully-dynamic streams with fixed memory size," in *KDD'16*.

[17] A. Pavan, K. Tangwongsan, S. Tirthapura, and K.-L. Wu, "Counting and sampling triangles from a graph stream," *PVLDB'13*.

[18] M. Jha, C. Seshadhri, and A. Pinar, "A space efficient streaming algorithm for triangle counting using the birthday paradox," in *KDD'13*.

[19] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis, "Beyond triangles: A distributed framework for estimating 3-profiles of large graphs," in *KDD'15*.

[20] ——, "Distributed estimation of graph 4-profiles," in *WWW'16*.

[21] P. Wang, J. Lui, and D. Towsley, "Minfer: Inferring motif statistics from sampled edges," in *ICDE'16*.

[22] N. Pržulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, 2007.

[23] N. K. Ahmed, J. Neville, R. A. Rossi, N. G. Duffield, and T. L. Willke, "Graphlet decomposition: Framework, algorithms, and applications," *Knowledge and Information Systems*, 2016.

[24] A. Sarajlić, N. Malod-Dognin, Ö. N. Yaveroğlu, and N. Pržulj, "Graphlet-based characterization of directed networks," *Scientific reports*, vol. 6, 2016.

[25] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *KDD'14*.

[26] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "DOULION: Counting triangles in massive graphs with a coin," in *KDD'09*.

[27] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, 1985.

[28] S. Wernicke, "Efficient detection of network motifs," *TCBB*, 2006.

[29] "KONECT Datasets: The koblenz network collection," http://konect.uni-koblenz.de, 2015.

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

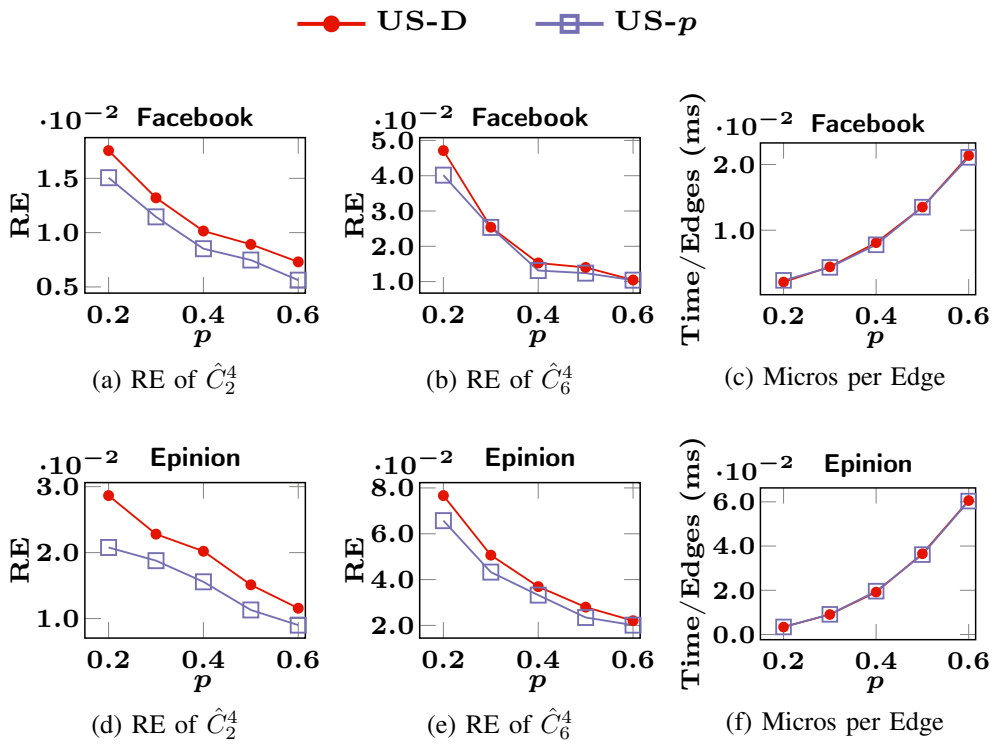
APPENDIX

A. Supplementary Proofs

Proof of Eq. (4).

Proof. W.l.o.g., we discuss the global graphlet counts. Let B be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}, e_t\}$. Assume B is isomorphic to g_i^k. Let δ_B be a random variable that takes value 1 if B is counted at the end of time t and 0 otherwise. The expected value of δ_B is $\mathbb{E}[\delta_B] = P_t(i,i)$. Let $\mathcal{S}(e_t)$ denote the set of subgraphs whose last edge on the stream is e_t. We have

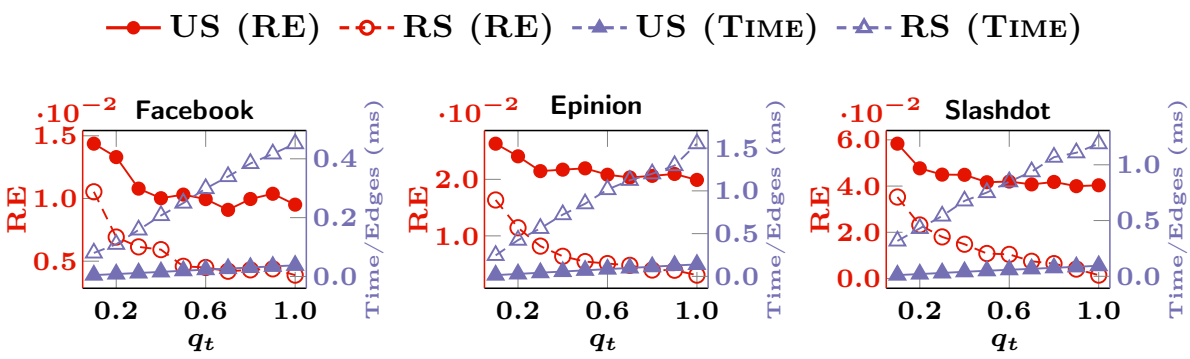
$$\mathbb{E}[\{\boldsymbol{d}'(e_t)\}_i] = \mathbb{E}\left[\sum_{\substack{B \in \mathcal{S}(e_t) \\ B \simeq g_i^k}} \delta_B\right] = P_t(i,i)\{\boldsymbol{d}(e_t)\}_i,$$



(a) RE of $\hat{C}_2^4$ (b) RE of $\hat{C}_6^4$ (c) Micros per Edge

(d) RE of $\hat{C}_2^4$ (e) RE of $\hat{C}_6^4$ (f) Micros per Edge

Fig. 6: Comparison between different counting strategies. The legend "US-D" refers to as the uniform sampling with dependent counting strategy, while "US-p" refers to as the uniform sampling with the independent counting strategy and $q_t = p$.



(a) Clique ⊠ (b) Clique ⊠ (c) Clique ⊠

Fig. 7: Effect of the counting probability q_t when we choose the IC strategy. For the US, we set $p = 0.6$. For the RS, we set the reservoir size $M = 0.6|E|$.

i.e., $\mathbb{E}[\boldsymbol{d}'(e_t)] = \boldsymbol{P}_t \boldsymbol{d}(e_t)$. Here $\{\boldsymbol{d}(e_t)\}_i$ is the i-th entry of $\boldsymbol{d}(e_t)$. Since $\boldsymbol{P}_t$ is a diagonal matrix and all the diagonal entries are positive, $\boldsymbol{P}$ must be invertible. Hence, we have

$$\sum_{t=1}^{T} \boldsymbol{d}(e_t) = \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{P}_t^{-1} \boldsymbol{d}'(e_t)\right]$$

■

Lemma 2. *Let $\boldsymbol{C}^k$ be the exact graphlet counts in $G^{(T)}$. We have*

$$\boldsymbol{A}\boldsymbol{C}^k = \sum_{t=1}^{T} \boldsymbol{d}(e_t) \tag{5}$$

Proof. We prove the lemma by discussing the following two cases.

1) *Global graphlet counts.* Let N_i^k denote the number of (normal) subgraphs in the graph $G^{(T)}$ that are isomorphic to graphlet g_i^k. Recall that $A(i,j)$ is the number of distinct copies of the graphlet g_i^k in the graphlet g_j^k. The relationship between C_i^k and N_i^k is as follows.

$$N_i^k = \sum_{j=1}^{|\mathcal{G}^k|} A(i,j) C_j^k.$$

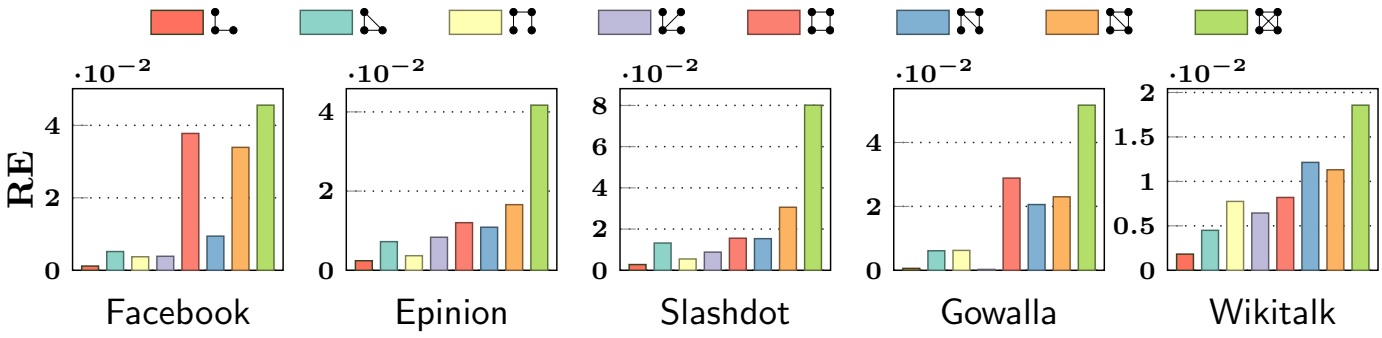


Fig. 8: The relative error of our framework when we choose RS and IC strategy. The reservoir size is $0.1|E|$ and the counting probability $q_t = 1$.

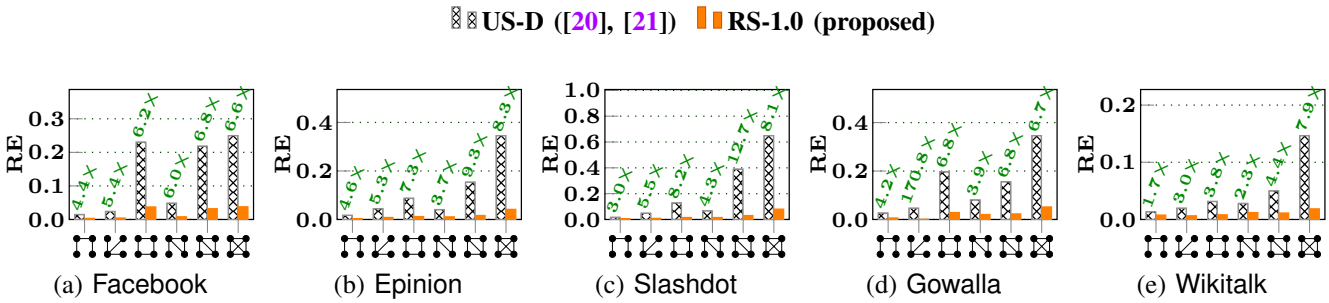


Fig. 9: Comparison between the method in our framework and the state-of-the-art methods [20], [21], which are also subsumed in our framework. For US, we set the edge sampling probability $p = 0.1$. For RS, we let the reservoir size $M = 0.1|E|$.

Notice that the sum $\sum_{t=1}^{T} \boldsymbol{d}(e_t)$ ensures each subgraph B with m edges $\{e_{t_1}, \cdots, e_{t_m}\}$ is counted exactly once at time t_m. Hence, the i-th entry of $\sum_{t=1}^{T} \boldsymbol{d}(e_t)$ is the number of subgraphs that are isomorphic to g_i^k, i.e., $\{\sum_{t=1}^{T} \boldsymbol{d}(e_t)\}_i = N_i^k = \sum_{j=1}^{|\mathcal{G}^k|} A(i,j) C_j^k$. Therefore, Eq. (5) is indeed the global graphlet counts.

2) *Local graphlet counts.* Let $N_i^k(v)$ denote the number of normal subgraphs in $G^{(T)}$ where node v occupies the orbit o_i^k. Recall that $A(i,j)$ denotes the number of ways to subtract a subgraph where node v occupies orbit o_i from another graph where node v occupies orbit o_j. Hence,

$$N_i^k(v) = \sum_{j=1}^{|\mathcal{O}^k|} A(i,j) O_j^k(v).$$

Using the fact that $\sum_{t=1}^{T} \boldsymbol{d}_v(e_t)$ ensures each subgraph where node v occupies orbit o_i^k is counted exactly once, we have $\{\sum_{t=1}^{T} \boldsymbol{d}_v(e_t)\}_i = N_i^k(v) = \sum_{j=1}^{|\mathcal{O}^k|} A(i,j) O_j^k(v)$, i.e., $\sum_{t=1}^{T} \boldsymbol{d}_v(e_t) = \boldsymbol{A}\boldsymbol{O}^k(v)$. According to the definition of $\boldsymbol{C^k}$ and $\boldsymbol{d}(e_t)$, we have $\boldsymbol{A}\boldsymbol{C}^k = \sum_{t=1}^{T} \boldsymbol{d}(e_t)$.

■

Lemma 3. *Let B be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}, e_t\}$, B' be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}\}$. Assume B is isomorphic to graphlet g_j^k. Let $d_i^k(B)$ denote the number of subgraphs in B that contain edge e_t and are isomorphic to graphlet g_i^k. We have*

$$d_i^k(B) = \begin{cases} A(i,j) - A(i,j') & \text{if } B' \simeq g_{j'}^k, \\ A(i,j) & \text{if } B' \text{ is disconnected.} \end{cases}$$

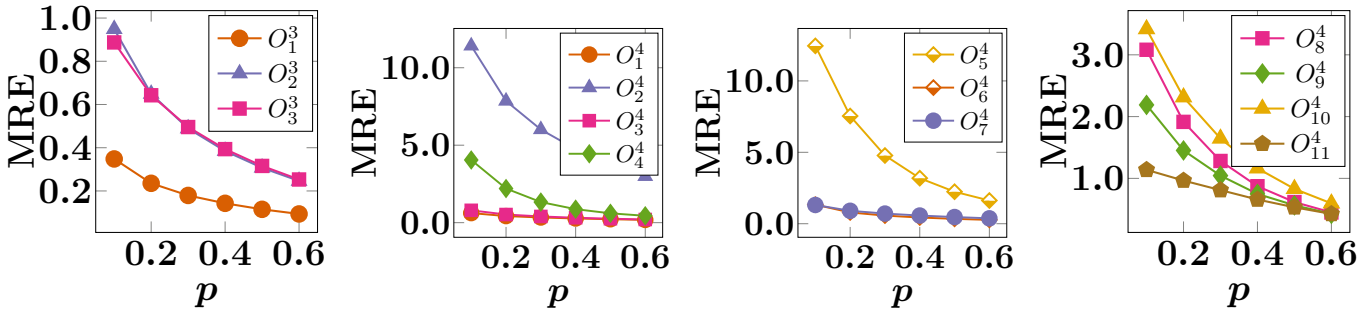
Proof. For a subgraph H of B, assume H does not contain edge e_t. Then H is also a subgraph of B'. On the other hand, if H contains edge e_t, H must not be a subgraph of B'. Recall that $A(i,j)$ is defined as the number of distinct copies of g_i^k in g_j^k. Hence we have $A(i,j) = d_i^k(B) + A(i,j')$ if $B' \simeq g_{j'}^k$ and $A(i,j) = d_i^k(B)$ if B' is disconnected. ■

B. Pseudo Code of 4-node Local Subgraph Counting

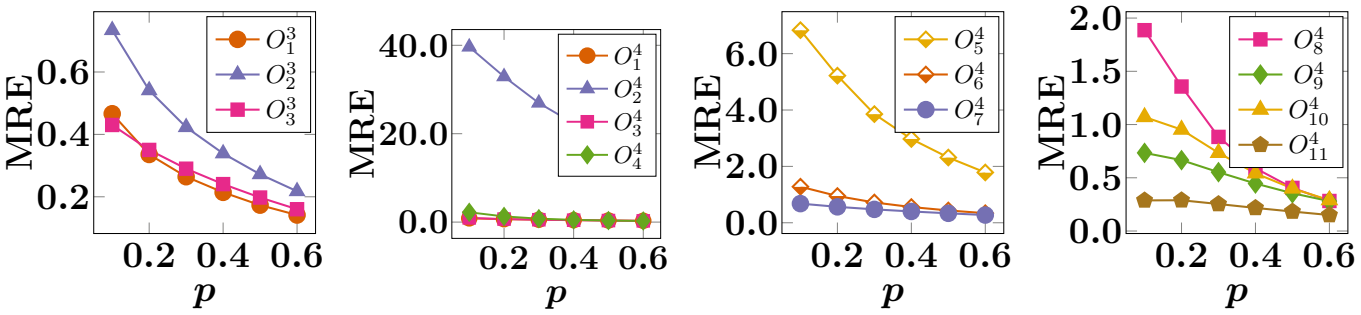
Algorithm 8 computes the 4-node local subgraph counts. The $\boldsymbol{e}_i$ in the pseudo code is defined as a vector whose only non-zero entry is the i-th entry with value 1. The matrix $\boldsymbol{A}$ is defined as the projection matrix for 4-node local graphlet counts in Table III. The equations in the pseudo code are derived based the following lemma.

Lemma 4. *Let B be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}, e_t\}$, B' be a subgraph formed by edges $\{e_{t_1}, \cdots, e_{t_{m-1}}\}$. Assume node v occupies orbit o_j^k in B. Let $d_i^k(B)$ denote the number of subgraphs of B that contain edge e_t and node v occupies orbit o_i^k in them. We have*

$$d_i^k(B) = \begin{cases} A(i,j) - A(i,j') & \text{if } v \text{ occupies } o_{j'}^k \text{ in } B' \\ A(i,j) & \text{if } B' \text{ is disconnected} \end{cases}$$

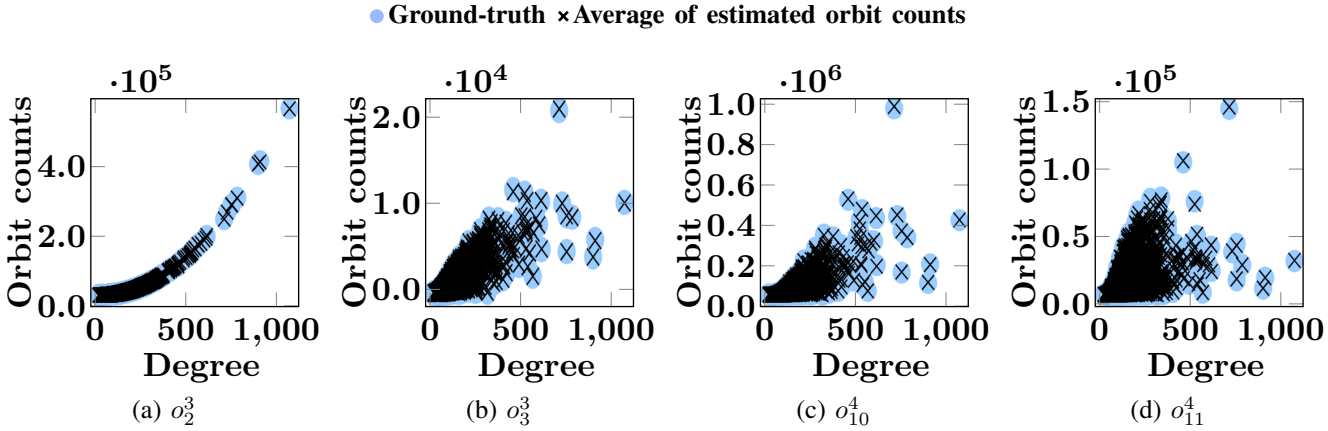


(a) Facebook



(b) Epinion

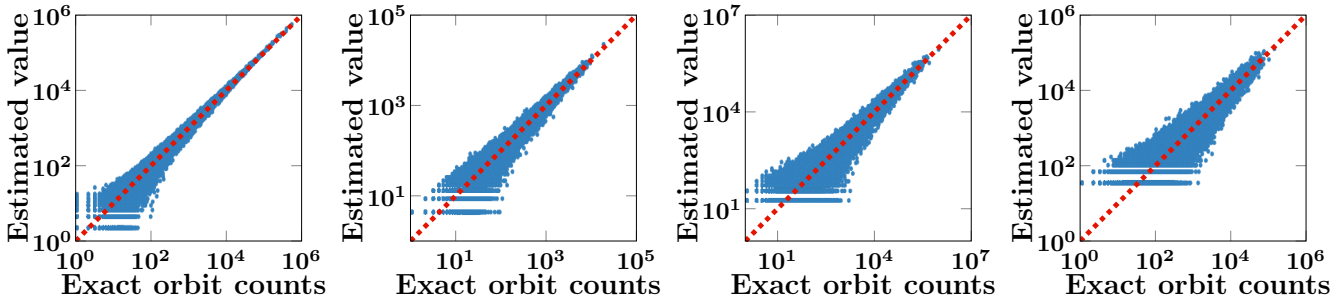
Fig. 10: Mean of relative error vs the sampling probability p of the uniform sampling. We let the counting probability $q_t = 1.0$.



(a) o_2^3 (b) o_3^3 (c) o_{10}^4 (d) o_{11}^4

Fig. 11: Scatter plot of the orbit counts vs degrees for all nodes in Facebook. The blue dots represent the ground truth. The crosses represent the average of estimated orbit counts from 100 simulations. We use the US scheme and IC strategy. The sampling probability $p = 0.5$ and the counting probability $q_t = 1.0$.

We omit the proof for Lemma 4 since it has similar proof to Lemma 3.



(a) o_2^3 (b) o_3^3 (c) o_{10}^4 (d) o_{11}^4

Fig. 12: Scatter plot between the true LGC and its estimation in Facebook in one simulation. We use the US scheme and IC strategy. The sampling probability $p = 0.5$ and the counting probability $q_t = 1.0$.

Algorithm 8 Counting 4-node local subgraph counts

function CountLocalSubgraph(G_S, e)
  $\boldsymbol{X} \leftarrow \mathbf{0}, \text{Star}_u = \emptyset, \text{Star}_v = \emptyset, \text{Tri}_e = \emptyset, \boldsymbol{N} = \mathbf{0}$
  for each $w \in \mathcal{N}_S(u)$ **do**
    Add w to Star_u and set $X(w) = 1$
  for each $w \in \mathcal{N}_S(v)$ **do**
    if $X(w) = 1$ **then**
      Add w to Tri_e and set $X(w) = 2$
      Remove w from Star_u
    else
      Add w to Star_v and set $X(w) = 3$
  TriOrbit($G_S, \text{Tri}_e, \boldsymbol{N}, u, v$)
  StarOrbit($G_S, \text{Star}_u, \boldsymbol{N}, u, v, 0$)
  StarOrbit($G_S, \text{Star}_v, \boldsymbol{N}, v, u, 4$)
  NbrNbrOrbit($G_S, \boldsymbol{N}, u, v$)
  NbrOrbit($G_S, \boldsymbol{N}, \text{Star}_u, u, v$)
  NbrOrbit($G_S, \boldsymbol{N}, \text{Star}_v, v, u$)
  return $\boldsymbol{d}' = \boldsymbol{A}\boldsymbol{N}$
function TriOrbit($G_s, \text{Tri}_e, \boldsymbol{N}, u, v$)
  for each $w \in \text{Tri}_e$ **do**
    for each $r \in \mathcal{N}_S(w) \backslash \{u, v\}$ **do**
      if $X[r] == 0$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_3,\ \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_3$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_8 - \boldsymbol{e}_4,\ \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_6 - \boldsymbol{e}_3$
      if $X[r] == 2$ and $w < r$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= (\boldsymbol{e}_{11} - \boldsymbol{e}_9 - \boldsymbol{e}_{10} + \boldsymbol{e}_5)$
        $\boldsymbol{N}(v) \mathrel{+}= (\boldsymbol{e}_{11} - \boldsymbol{e}_9 - \boldsymbol{e}_{10} + \boldsymbol{e}_5)$
        $\boldsymbol{N}(w) \mathrel{+}= (\boldsymbol{e}_{11} - \boldsymbol{e}_{10} - \boldsymbol{e}_9 + \boldsymbol{e}_5)$
        $\boldsymbol{N}(r) \mathrel{+}= (\boldsymbol{e}_{11} - \boldsymbol{e}_{10} - \boldsymbol{e}_9 + \boldsymbol{e}_5)$
function StarOrbit($G_S, \text{Star}, \boldsymbol{N}, u, v, s$)
  for $w \in \text{Star}$ **do**
    for $r \in \mathcal{N}_S(w) \backslash \{u, v\}$ **do**
      if $X[r] == 0$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_2, \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_1$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_2, \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_1$
      if $|X[r] - s| == 1$ and $w < r$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_8 - \boldsymbol{e}_4, \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_6 - \boldsymbol{e}_3$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_3, \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_3$
      if $|X[r] - s| == 2$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_{10} - \boldsymbol{e}_7 - \boldsymbol{e}_8 + \boldsymbol{e}_2$
        $\boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_9 - \boldsymbol{e}_6 - \boldsymbol{e}_7 + \boldsymbol{e}_1$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_9 - \boldsymbol{e}_6 - \boldsymbol{e}_7 + \boldsymbol{e}_1$
        $\boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_{10} - \boldsymbol{e}_7 - \boldsymbol{e}_8 + \boldsymbol{e}_2$
      if $|X[r] - s| == 3$ and $w < r$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_5 - \boldsymbol{e}_1 - \boldsymbol{e}_2$
        $\boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_5 - \boldsymbol{e}_1 - \boldsymbol{e}_2$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_5 - \boldsymbol{e}_2 - \boldsymbol{e}_1$
        $\boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_5 - \boldsymbol{e}_2 - \boldsymbol{e}_1$
function NbrNbrOrbit($G_s, \text{Tri}, \boldsymbol{N}, u, v$)
  for $w \in \mathcal{N}_S(u)$ **do**
    for $r \in \mathcal{N}_S(v) \backslash \{w\}$ **do**
      if $X[w] == 2$ and $X[r] == 2$ and $w < r$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_{10} - \boldsymbol{e}_5, \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_{10} - \boldsymbol{e}_5$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_9 - \boldsymbol{e}_5, \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_9 - \boldsymbol{e}_5$
      if $X[w] == 1$ and $X[r] == 3$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_2, \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_2$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_1, \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_1$
      if $X[w] == 1$ and $X[r] == 2$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_8 - \boldsymbol{e}_2, \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_1$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_6 - \boldsymbol{e}_1, \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_2$
      if $X[w] == 2$ and $X[r] == 3$ **then**
        $\boldsymbol{N}(u) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_1, \boldsymbol{N}(v) \mathrel{+}= \boldsymbol{e}_8 - \boldsymbol{e}_2$
        $\boldsymbol{N}(w) \mathrel{+}= \boldsymbol{e}_7 - \boldsymbol{e}_2, \boldsymbol{N}(r) \mathrel{+}= \boldsymbol{e}_6 - \boldsymbol{e}_1$
function NbrOrbit($G_S, \boldsymbol{N}, \text{Star}, u, v$)
  $\boldsymbol{N}_4(u) \mathrel{+}= \binom{|\text{Star}|}{2}, \boldsymbol{N}_3(v) \mathrel{+}= \binom{|\text{Star}|}{2}$
  for $w \in \mathcal{N}_s(u)$ **do**
    $\boldsymbol{N}_3(w) \mathrel{+}= |\text{Star}| - 1$