

Estimation of Graphlet Counts in Massive Networks

Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed

Abstract—Graphlets are induced subgraphs of a large network and are important for understanding and modeling complex networks. Despite their practical importance, graphlets have been severely limited to applications and domains with relatively small graphs. Most previous work has focused on *exact algorithms*, however, it is often too expensive to compute graphlets exactly in massive networks with billions of edges, and finding an approximate count is usually sufficient for many applications. In this work, we propose an *unbiased graphlet estimation framework* that is (a) fast with large speedups compared to the state-of-the-art, (b) parallel with nearly linear-speedups, (c) accurate with less than 1% relative error, (d) scalable and space-efficient for massive networks with billions of edges, and (e) effective for a variety of real-world settings, as well as estimating global and local graphlet statistics (e.g., counts). On 300 networks from 20 domains, we obtain <1% relative error for all graphlets. This is vastly more accurate than existing methods while using less data. Moreover, it takes a few seconds on billion edge graphs (as opposed to days/weeks). These are by far the largest graphlet computations to date.

Index Terms—Graphlets, network motifs, induced subgraphs, estimation methods, unbiased graphlet estimation, local graphlet count estimation, graphlet statistics, parallel algorithms, higher-order network analysis, machine learning.

I. INTRODUCTION

GRAPHLETS are small *induced subgraphs*¹ and are important for many predictive and descriptive modeling and learning systems/tasks [1]–[8] such as image processing and computer vision learning systems that use neural networks [1], [9], network alignment [6], [10]–[12], classification [2], [3], visualization and sensemaking [13], [14], dynamic network analysis [15], [16], community detection [17]–[19], role discovery [20], anomaly detection [21], [22], and link prediction [8], [23], [24]. Unfortunately, the application and general use of graphlets (especially those of size $k = 4$ nodes and larger) remains severely limited to networks that are small enough to avoid the scalability and performance limitations of *exact* algorithms [13], [25]–[28]. For instance, Shervashidze *et al.* [3] takes hours to count graphlets on small networks (i.e., few hundreds/thousands of nodes/edges) for graph classification [2].

In many applications, finding an ‘approximate’ answer is usually sufficient where the exact answer is not worth the extra cost and time. The recent rise of Big Data [29] has made approximation methods even more critical [30], especially for practical applications [31]–[35]. More recently, approximation methods have been proposed for important problems such as

triangle counting [36]–[40], shortest path problems [33], [41], finding max cliques [42], and many others.

This work aims to overcome the above computational limitations to make graphlets more accessible to other applications/domains with much larger graphs. In particular, this work proposes a general graphlet estimation framework for deriving unbiased estimates² of a variety of graphlet statistics (e.g., frequency of an arbitrary k -vertex induced subgraph) from a small set of edge-induced neighborhoods. The graphlet estimators provide accurate and fast approximations of a variety of global and local graphlet properties³. Intuitively, a global graphlet property assigns a single value (or distribution/map) to a graph G , whereas a local graphlet property assigns a single value (or distribution/map) to a particular graph element such as an edge or node of G [43]. An example of a global graphlet statistic is the total number of 4-cliques in G , whereas an example of a local statistic is the number of 4-cliques containing a certain graph element such as an edge or node.⁴ Furthermore, a number of important machine learning tasks are likely to benefit from the proposed graphlet estimation framework, including graph anomaly detection [21], [22], entity resolution [44], role discovery [45], and relational classification [46].

The key contributions of this work are as follows:

- **Graphlet estimation framework:** A general unbiased estimation framework is proposed for approximating global and local graphlet properties (such as counts) in massive networks with billions of edges. The framework is shown to be accurate, fast, and scalable for *both* dense and sparse networks of arbitrary size.
- **Accurate:** For all graphlets and data (300 graphs from 20 domains), the methods are more accurate than existing state-of-the-art methods (<1% relative error) while using only a small fraction of the data. Provable error bounds are also derived and shown to be tight (see Section IV-B).
- **Efficient:** The proposed estimation algorithms are orders of magnitude faster than the recent state-of-the-art algorithm and take a few seconds as opposed to days/months.
- **Parallel methods:** This work proposes parallel graphlet estimation methods for shared and distributed-memory architectures. Strong scaling results with nearly linear speedups are observed across a variety of networks.
- **Estimation of graphlet statistics – beyond counts:** This work proposes estimation methods for both global and local graphlet counts, as well as other graphlet properties beyond simple counts (See Section II-B). This is in contrast to existing

R. A. Rossi is with Adobe Research, 345 Park Ave, San Jose, CA 95110
Email: rrossi@adobe.com

R. Zhou is with Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043
Email: rongzhou@google.com

N. K. Ahmed is with Intel Labs, 3065 Bowers Ave, Santa Clara, CA 95052
Email: nesreen.k.ahmed@intel.com

¹The terms graphlet, induced subgraph, and network motif are interchangeable.

²A graphlet estimate X_i is unbiased if $\mathbb{E}[X_i] = Y$ [36] and Y is the actual.

³The term *graphlet properties* is used more generally to refer to graphlet (single-valued) statistics and distributions.

⁴Note that the total number of 4-cliques in G is an example of a global graphlet statistic since it is computed over all graph elements (edges, nodes) in G , as opposed to an individual graph element in G .

estimation methods for graphlets [47]–[49] that focus only on approximating global graphlet counts.

- **Largest investigation and graphlet computations:** To the best of our knowledge, this work provides the (i) largest graphlet computations to date *and* the (ii) largest empirical investigation using 300+ networks from 20+ domains.

II. LOCALIZED GRAPHLET ESTIMATION FRAMEWORK

In this section, we propose a new family of graphlet estimation methods based on selecting a set of localized neighborhoods. This gives rise to the *localized graphlet estimation framework* (LGE) which serves as a basis for deriving unbiased and consistent estimators that are fast, accurate, and scalable for massive networks. As shown later in Section IV, LGE has many interchangeable components and is effective for a wide variety of networks, applications, and domains (e.g., biological, social, and infrastructure/physical networks), which have fundamentally different structural properties.

A. Preliminaries

Let $G = (V, E)$ be an undirected simple graph with $N = |V|$ vertices and $M = |E|$ edges. Sets are *ordered*, unless otherwise mentioned. Given a vertex $v \in V$, let $\Gamma(v) = \{w \mid (v, w) \in E\}$ be the set of vertices adjacent to v in G . Similarly, given an edge $e = (u, v) \in E$, let $\Gamma(e)$ denote the edge neighborhood of e defined as:

$$\Gamma(e) = \Gamma(u, v) = \Gamma(u) \cup \Gamma(v) \setminus \{u, v\}, \quad (1)$$

where $\Gamma(u)$ and $\Gamma(v)$ are the neighbors of u and v , respectively. The (explicit) edge neighborhood is $\Gamma_e = G(\{\Gamma(v) - u\} \cup \{\Gamma(u) - v\})$. The subgraph Γ_e consists of the set of vertices adjacent to v or u (non-inclusive) and all edges between that set. Moreover, the degree of a vertex v denoted as $d_v = |\Gamma(v)|$ is equal to the number of edges that contain v . We also denote $\Delta(G)$ as the maximum vertex degree.

Definition 1 (GRAPHLET): A graphlet $G_i = (V_k, E_k)$ is a connected induced subgraph consisting of a subset $V_k \subset V$ of k vertices from $G = (V, E)$ together with all edges whose

TABLE I
SUMMARY OF GRAPHLET PROPERTIES AND NOTATION

Summary of the notation and properties for graphlets of size $k = \{2, 3, 4\}$. Note that ρ denotes density; Δ and \bar{d} denote the max and mean degree; r denotes assortativity; $ T $ is the total number of triangles in G .							
	Description	ρ	Δ	\bar{d}	r	$ T $	
	G_1 edge	1.00	1	1.0	1.00	0	
	G_2 triangle	1.00	2	2.0	1.00	1	
	G_3 2-star	0.67	2	1.33	-1.00	0	
	G_4 4-clique	1.00	3	3.0	1.00	4	
	G_5 chordal-cycle	0.83	3	2.5	-0.66	2	
	G_6 tailed-triangle	0.67	3	2.0	-0.71	1	
	G_7 4-cycle	0.67	2	2.0	1.00	0	
	G_8 3-star	0.50	3	1.5	-1.00	0	
	G_9 4-path	0.50	2	1.5	-0.50	0	

TABLE II
COMPARISON OF GRAPHLET ESTIMATION METHODS.

Qualitative and quantitative comparison of the two main classes of graphlet estimation methods, namely, *direct methods* are those that sample k -vertices directly and retrieve the graphlet induced by that subset, whereas the proposed family of *localized graphlet estimation* (LGE) methods select ℓ -neighborhoods for estimation (and sample from them). The first six columns refer to the *global* and *local* graphlet estimation problems. In particular, columns 1-3 refer to the *global* graphlet statistics and distributions estimated by the methods (counts, GFD, and others such as extremal statistics/distributions, etc.), whereas columns 4-6 refer to the *local* graphlet statistics and distributions. “Parallel” refers to parallel estimation methods. “Space efficient” holds true if the space requirements of the algorithm are sublinear (preferably poly-logarithmic in the size of the input). “Streaming” holds true if the method is amenable to streaming implementation. “Position-aware” is true if the algorithm supports position-aware graphlets (orbits). “Sparse & dense” is true if the method has limited assumptions, and designed/capable of handling both sparse and dense graphs. “Parameter-free” methods are those that do not expect any user-specified input parameters (though they can be set, but is not required). “All graphlets” holds true if the method computes graphlet statistics and distributions for all graphlets up to size k .

	Method	GLOBAL			LOCAL			COMPUTATIONAL			
		Counts	GFD	Others	Counts	GFD	Others	Parallel	Space efficient	Streaming	Position-aware
DIRECT	SHERV. <i>et al.</i> [3]		✓						✓	✓	
	GUISE [47]		✓								
	GRAFT [48]		✓								
	3-PATH SAMP. [49]	✓									
LGE	UNIFORM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	KCORE	✓	✓	✓	✓	✓	✓	✓		✓	✓

endpoints are both in this subset $E_k = \{\forall e \in E \mid e = (u, v) \wedge u, v \in V_k\}$. By definition, a graphlet has one connected component.

A k -graphlet is defined as an *induced subgraph* with k -vertices. Further, we define $\mathcal{G}^{(k)}$ as the set of k -vertex induced subgraphs and $\mathcal{G} = \mathcal{G}^{(2)} \cup \dots \cup \mathcal{G}^{(k)}$. A graphlet frequency distribution is defined as:

Definition 2 (GRAPHLET FREQUENCY DISTRIBUTION): Given a graph (or graph element such as an edge, node, or subgraph), the graphlet frequency distribution (GFD) is defined as $f_i = \frac{X_i}{\sum_i X_i}$ for all $i = 1, \dots, |\mathcal{G}|$ where X_i is the frequency of graphlet G_i . The resulting vector $\mathbf{f} = [\dots f_i \dots]$ is the GFD.

A summary of the graphlet notation and important properties are provided in Table I.

B. Problem Formulation

The goal of this work is to obtain fast and accurate estimates of a variety of graphlet statistics (e.g., counts). In particular, we focus on four basic types of graphlet statistics that can be described by two pairs of exclusive attributes: single-valued vs. distribution *and* global vs. local (See [43] for further details). Intuitively, the four types of graphlet properties are as follows:

- P1 Global single-valued statistics** such as the total number of 4-cycles in G
- P2 Global distributions**, e.g., a GFD computed using the total graphlet frequencies of G

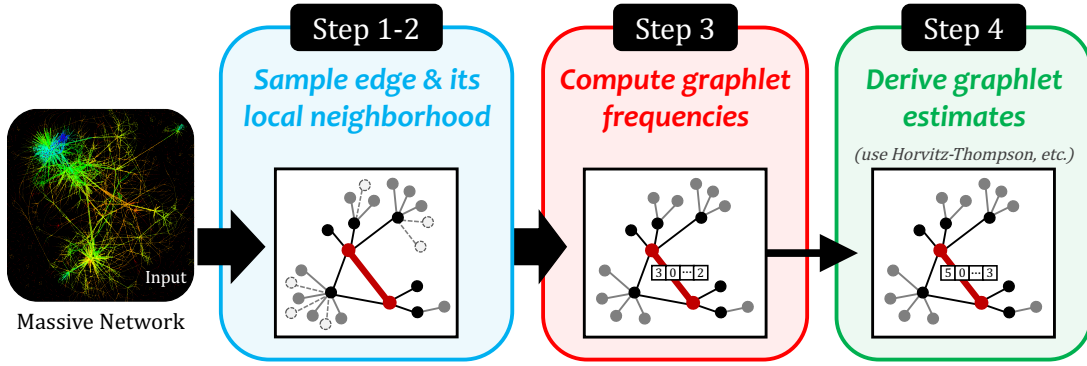


Fig. 1. Overview of the Localized Graphlet Estimation (LGE) framework for estimating local graphlet counts. The sampled edge from Step 1 is shown in red (—) and highlighted. Unsampling neighbors shown in step 1-2 above are dashed and light gray.

P3 *Local single-valued statistics*, e.g., the number of 4-cycles containing a specific graph element such as an edge (node)

P4 *Local distributions*, e.g., a GFD of an individual graph element such as an edge (or node)

The proposed framework gives rise to graphlet estimation methods that are fast and accurate for a variety of these four types of graphlet properties. Obviously, estimation methods for global/local graphlet *statistics* return a single-value (scalar value), whereas methods for global/local distributions return the estimated distribution.

Now we define the specific graphlet statistics and distributions (from the four types of graphlet properties above) that are estimated in this work including:

- Counts of graphlets $G_i \in \mathcal{G}$, for all $i = 1, 2, \dots, |\mathcal{G}|$ or the count of a specific graphlet $G_i \in \mathcal{G}$. In particular, this work proposes estimators for both global $\mathbf{X}_G = [x_1 \ x_2 \ \dots]$ and local graphlet counts $\mathbf{x}_i = [x_1 \ x_2 \ \dots]$ for edge $e_i \in E$.
- Graphlet frequency distributions (Definition 2)
- Aggregate (single-valued) statistics such as the max, mean, median, variance, etc. These aggregate statistics are derived by calculating, choosing, or constructing a single value from a local statistic or distribution.

Estimators are derived for each of the graphlet problems above. Existing estimation methods such as [47]–[49] are limited to simple count statistics, whereas this paper instead proposes a unifying unbiased estimation framework that generalizes for a variety of other important graphlet properties beyond simple counts.⁵ In addition, previous work has mainly focused on estimating *global statistics* (e.g., total count of a graphlet in G), whereas this work introduces estimators for both global and local graphlet statistics and distributions.

C. Framework Outline and Intuition

An overview of the localized graphlet estimation (LGE) framework is provided in Fig. 1 for intuition. The LGE framework is based on the following general steps:

S1 Sampling a graph element (edge, node).

S2 Obtaining a localized subgraph H by sampling the local neighborhood (egonet) of that graph element.

S3 Given H , compute the graphlets containing the graph element (sampled from step S1).

S4 Use these counts to derive unbiased estimates (e.g., using the Horvitz-Thompson construction [36], [50]). Note if estimating global graphlet statistics, then steps S1–S3 are repeated K times prior to deriving estimates.

In this work, we sample edge neighborhoods (as opposed to node neighborhoods). In particular, an edge neighborhood $\Gamma(e)$ is sampled with some probability from the set of all edge-induced neighborhoods (See Alg. 1). Using the (potentially partial) edge neighborhood $\Gamma(e)$ centered at $e \in E$ as a basis, we compute the frequency of each graphlet $G_i \in \mathcal{G}$, for $i = 1, \dots, |\mathcal{G}|$. Let us note that the edge (or node) may be selected uniformly at random or by an arbitrary weighted distribution F (as shown in Alg. 1). The weighted distribution could be based on degrees, k -core numbers, or any attribute of interest. Further, an edge (or node) neighborhood may be selected with replacement or without. Selecting an edge neighborhood with replacement allows each edge neighborhood $\Gamma(e)$ to be used multiple times, whereas sampling without replacement ensures that each edge neighborhood included in the sample is unique (by label) and never repeated. Edge-centric graphlet decomposition algorithms⁶ also lend themselves for (parallel) implementation on both shared-memory and distributed-memory architectures (see Section III). Additionally, the fastest state-of-the-art subgraph counting approach can always be used in the proposed framework (e.g., Step 3 in Fig. 1) to speedup the estimation even further. For instance, instead of using PGD [13] to count graphlets (as done in this work), one can always use the fastest state-of-the-art subgraph counting algorithm.

A taxonomy for graphlet estimation is proposed in Table II. In particular, existing graphlet estimation methods can be categorized as *direct graphlet estimation methods* since they sample a set of k -vertices *directly* from G and retrieve the k -graphlet induced by that set, whereas the proposed class of localized graphlet estimation (LGE) methods select an edge and sample locally from the neighborhood. Table II also

⁵As an aside, counts have been used for many important measures in computational biology such as the graphlet degree distribution (GDD) [11] and agreements (GDDA) [11].

⁶The term edge-centric refers to algorithms that iterate over edges as opposed to nodes, see [51].

summarizes existing estimation methods as well as our proposed approach according to the global and local graphlet (single-valued) statistics and distributions estimated by each, as well as computational and algorithmic properties offered by each approach. Section II-D introduces a general estimation framework for global graphlet statistics, whereas Section II-E proposes a framework for estimating local graphlet statistics.

D. Estimation of Global Graphlet Statistics

Given the sampled set of edge-centric neighborhood, we show how to compute the estimated graphlet counts in Alg. 2. More formally, let $T_e = \Gamma(u) \cap \Gamma(v)$ be the set of nodes that complete triangles with $e(v, u) \in J$. Likewise, $S_u = \{w \in \Gamma(u) \setminus \{v\} | w \notin \Gamma(v)\}$ and $S_v = \{w \in \Gamma(v) \setminus \{u\} | w \notin \Gamma(u)\}$, and thus $|S_v|$ and $|S_u|$ are the number of 2-stars centered at v and u , respectively. These quantities are computed in Line 5-9 of Alg. 2. For further intuition, see Fig. 2. Let us also note that $\Psi(\cdot)$ is a hash table for checking edge existence in $o(1)$ time (see Alg. 2). As an aside, this is an implementation detail and $\Psi(\cdot)$ can easily be replaced with another data structure (bloom filters, etc) or even removed entirely in favor of binary search (which may be favorable in situations where memory is limited). These possibilities are discussed in detail later. Notice that $\Psi(\cdot)$ is also used as a way to encode the different types of nodes. Thus, nodes are hashed using λ_1 , λ_2 , and λ_3 , which may be defined as any unique symbol. In our implementation, we avoid the cost of resetting by ensuring that each λ_i is unique for each edge-centric neighborhood. In Alg. 2 Line 4, we mark the neighbors $\Gamma(v)$ of v as λ_1 . Later in Line 8 a triangle is marked with λ_3 , whereas Line 9 encodes a wedge as λ_2 .

Next, Alg. 2 maintains the *unrestricted graphlet counts*⁷ (Eq. 2-5):

$$C_5 \pm \binom{|T_e|}{2} \quad (2)$$

$$C_6 \pm |T_e| \cdot (|S_u| + |S_v|) \quad (3)$$

$$C_8 \pm \binom{|S_u|}{2} + \binom{|S_v|}{2} \quad (4)$$

$$C_9 \pm |S_u| \cdot |S_v| \quad (5)$$

The quantities C_5 , C_6 , C_8 , and C_9 are later used to derive chordal-cycles, tailed-triangles, 3-stars, and 4-paths in $o(1)$ time, respectively. In addition, we maintain C_2 , C_3 , C_4 , and C_7 (see Alg. 2). These quantities are computed for each edge-centric neighborhood in the sample, and then used for estimation. In particular, the 3-vertex graphlet counts are estimated as follows:

$$X_2 = W_2 \sigma_2 C_2 \quad (6)$$

$$X_3 = W_3 \sigma_3 C_3 \quad (7)$$

where X_2 and X_3 are the estimated counts of graphlets G_2 and G_3 , respectively. Similarly, the 4-vertex graphlet counts are

estimated via Eq 8-13 as follows:

$$X_4 = W_4 \sigma_4 C_4 \quad (8)$$

$$X_5 = W_5 \sigma_5 (C_5 - C_4) \quad (9)$$

$$X_6 = W_6 (\sigma_6 C_6 - 4X_5) \quad (10)$$

$$X_7 = W_7 \sigma_7 C_7 \quad (11)$$

$$X_8 = W_8 (\sigma_8 C_8 - X_6) \quad (12)$$

$$X_9 = W_9 \sigma_9 (C_9 - C_7) \quad (13)$$

where X_4 – X_9 are the estimated counts of the graphlets G_4 – G_9 , respectively. Further, $\mathbf{W} \in \mathbb{R}^\kappa$ is a weight vector to account for the edge multiplicities:

$$\mathbf{W} = \left[1 \frac{1}{3} \frac{1}{3} \frac{1}{6} \frac{1}{6} 1 \frac{1}{2} \frac{1}{4} \frac{1}{3} 1 \right]^T \quad (14)$$

Algorithm 1 Localized graphlet estimation (LGE) framework for estimating *global graphlet statistics*.

Input:

a graph $G = (V, E)$

a sample size K , or sample probability p

- 1 **parallel for** $j = 1, 2, \dots, K$ **do**
 - 2 Select e via an arbitrary (weighted/uniform) distribution F
 - 3 Set $J \leftarrow J \cup \{e\}$
 - 4 Obtain estimated graphlet counts \mathbf{X} for J via Alg. 2
 - 5 **return** \mathbf{X} – the estimated graphlet counts
-

Algorithm 2 A family of parallel LGE methods for deriving unbiased estimates of *global graphlet statistics*

Input: a graph G and a set of sampled edges J

- 1 **parallel for each** $e = (v, u) \in J$ **in order do**
 - 2 Reset $T_e = \emptyset$ and $S_u = \emptyset$
 - 3 **for** $w \in \Gamma(v)$ **do**
 - 4 **if** $w \neq u$ **then** $\Psi(w) = \lambda_1$
 - 5 **for** $w \in \Gamma(u)$ **do**
 - 6 **if** $w = v$ **then continue**
 - 7 **if** $\Psi(w) = \lambda_1$ **then**
 - 8 $T_e \leftarrow T_e \cup \{w\}$ and set $\Psi(w) = \lambda_3$ ▷ triangle
 - 9 **else** $S_u \leftarrow S_u \cup \{w\}$ and set $\Psi(w) = \lambda_2$ ▷ wedge
 - 10 Update unrestricted counts via Eq. 2-5
 - 11 $C_2 \pm |T_e|$ ▷ Note \pm is the addition sum $C_2 = C_2 + |T_e|$
 - 12 $C_3 \pm C_3(e) = (d_u + d_v - 2)|T_e|$ ▷ equiv. $|S_u| + |S_v|$
 - 13 $C_4 \pm C_4(e) = \text{CLIQUE}(\Psi, T_e)$ ▷ in parallel (Alg. 3)
 - 14 $C_7 \pm C_7(e) = \text{CYCLE}(\Psi, S_u)$ ▷ in parallel (Alg. 4)
 - 15 **end parallel**
 - 16 Compute estimated graphlet counts \mathbf{X} via Eq. 6-13
 - 17 **return** \mathbf{X} , where X_i is the estimate for graphlet G_i
-

where each W_i is used to correct the counts of graphlet G_i [52]. Further, we also define $\mathbf{p} \in \mathbb{R}^\kappa$ as a vector of sampling probabilities where p_i is the sampling probability for graphlet $G_i \in \mathcal{G}$. Note that p_i can be proportional to any arbitrary function/weight computed on the graph G . One possibility is to use uniform sampling probabilities such that each p_i is:

$$p_i = |J|/|E| \quad (15)$$

Intuitively, p_i is the fraction of edge neighborhoods selected thus far. Results for both uniform and non-uniform sampling probabilities are discussed and investigated in Section IV. In

⁷Note all count variables are initialized to zero; and \pm is the addition assignment operator.

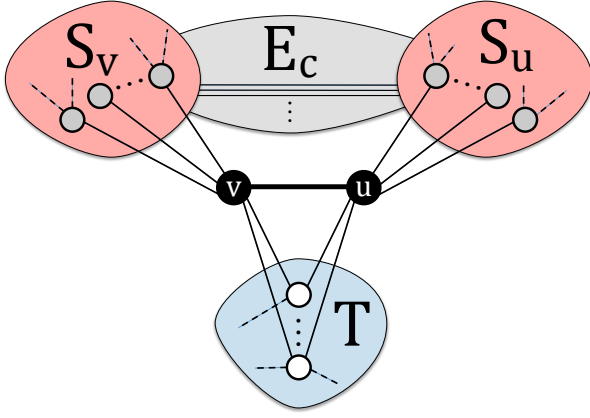


Fig. 2. Let T be the set of nodes completing a triangle with the edge $(v, u) \in E$, and let S_v and S_u be the set of nodes that form a 2-star with v and u , respectively. Further, E_c is the set of edges that complete a 4-cycle w.r.t. the edge $e = (v, u)$ where for each edge $(r, s) \in E_c$ such that $r \in S_v$ and $s \in S_u$ and both $(r \cap S_u) \cup (s \cap S_v) = \emptyset$.

addition, let σ_i be defined as:

$$\sigma_i = \frac{1}{p_i}$$

where σ_i is the inverse sampling probability of graphlet i used to correct the sampling bias. This corresponds to the Horvitz-Thompson construction [36], [50]. Let us note that in Alg. 2, the cliques and cycles are computed via Alg. 3 and Alg. 4 using information from the $(k-1)$ -graphlets. However, when memory is limited, then Alg. 5–6 should be used. These methods search over the sets T_e , S_u , S_v from the $(k-1)$ -graphlets directly using binary search.

Error Analysis: Let $Y_i(e)$ be the true count of an arbitrary induced subgraph $G_i \in \mathcal{G}$ iff the subgraph is incident to e , then $Y_i = \sum_{e \in E} Y_i(e)$. Assume we sample a set of edge neighborhoods with probability ϕ , then $X_i = \sum_{e \in J} \frac{Y_i(e)}{\phi}$ where X_i is an estimator for \mathcal{G}_i . $\mathbb{E}[X_i] = Y_i$ is an unbiased estimate.

Algorithm 3 Clique counts via neigh-iter.

```

1 procedure CLIQUE( $\Psi, T_e$ )
2   Set  $K_e \leftarrow 0$ 
3   parallel for each  $w \in T_e$  do
4     for each  $r \in \Gamma(w)$  where  $\Psi(r) = \lambda_3$  do set  $K_e \leftarrow K_e + 1$ 
5     Reset  $\Psi(w)$  to 0
6   return  $K_e$ 

```

Algorithm 4 Cycle counts via neigh-iter.

```

1 procedure CYCLE( $\Psi, S_u$ )
2   Set  $C_e \leftarrow 0$ 
3   parallel for each  $w \in S_u$  do
4     for each  $r \in \Gamma(w)$  where  $\Psi(r) = \lambda_2$  do set  $C_e \leftarrow C_e + 1$ 
5     Reset  $\Psi(w)$  to 0
6   return  $C_e$ 

```

Algorithm 5 Clique counts restricted to searching T_e

```

1 procedure CLIQUERES( $\Psi, T_e$ )
2   Set  $K_e \leftarrow 0$ 
3   parallel for each vertex  $w_i$  in an ordering  $w_1, w_2, \dots$  of  $T_e$  do
4     for all each  $w_j \in \{w_{i+1}, \dots, w_{|T_e|}\}$  in order do
5       if  $w_i \in \Gamma(w_j)$  via binary search then  $K_e \leftarrow K_e + 1$ 
6   return  $K_e$ 

```

Algorithm 6 Cycle counts restricted to S_u and S_v

```

1 procedure CYCLERES( $\Psi, S_u, S_v$ )
2   Set  $C_e \leftarrow 0$ 
3   parallel for each  $w \in S_u$  do
4     for all  $r \in S_v$  do
5       if  $r \in \Gamma(w)$  via binary search then  $C_e \leftarrow C_e + 1$ 
6   return  $C_e$ 

```

The proof is as follows.

$$\mathbb{E}[X_i] = \mathbb{E}\left[\sum_{e \in J} \frac{Y_i(e)}{\phi}\right] = \sum_{e \in J} \mathbb{E}\left[\frac{Y_i(e)}{\phi}\right] \quad (16)$$

$$= \sum_{e \in E} \frac{\mathbb{E}[\mathcal{I}_e]}{\phi} \cdot Y_i(e) = \sum_{e \in E} \frac{Y_i(e)}{\phi} \cdot \phi = Y_i \quad (17)$$

since \mathcal{I}_e is a Bernoulli random variable that indicates whether e and its neighborhood is sampled. Further, the mean squared error $\text{MSE}(X_i)$ is:

$$\mathbb{E}[(X_i - Y_i)^2] = \underbrace{\mathbb{V}[X_i]}_{\text{Variance}} + \underbrace{(\mathbb{E}[X_i] - Y_i)^2}_{\text{Bias}} \quad (18)$$

where $\mathbb{V}[X_i]$ is the variance component and $(\mathbb{E}[X_i] - Y_i)^2$ is the bias component of the estimator X_i . Therefore, $\text{MSE}(X_i) = \mathbb{V}[X_i]$ since X_i is an *unbiased estimator*.

Complexity: Let T_{\max} and S_{\max} denote the maximum number of triangles and stars incident to a selected edge $e \in J$. Note that S_{\max} in reality is much smaller since for each edge $e = (v, u) \in J$, Alg. 2 computes only S_u ⁸ such that $d_u \leq d_v$, and thus $|S_u| \leq |S_v|$. For a single $\Gamma(e)$, Alg. 2 counts 4-cliques and 4-cycles centered at e in $\mathcal{O}(\Delta T_{\max})$ and $\mathcal{O}(\Delta S_{\max})$, respectively. From either 4-cliques/cycles, we derive all other graphlet counts in $o(1)$ using combinatorial relationships along with the $(k-1)$ -graphlets. Thus, Alg. 2 counts all graphlets for $\{\Gamma(e_1), \dots, \Gamma(e_K)\}$ up to $k = 4$ in:

$$\mathcal{O}(K \Delta T_{\max} + K \Delta S_{\max}) = \mathcal{O}(K \Delta (T_{\max} + S_{\max}))$$

Using K processing units (cores, workers), this reduces to $\mathcal{O}(\Delta (T_{\max} + S_{\max}))$. Our approach is also space-efficient and requires a lot less space than existing approaches [25], [26], [47], [48]. In particular, the space complexity of Alg. 2 is $\mathcal{O}(N + 2\Delta - 1) = \mathcal{O}(N)$ using a hash table Φ of size $n = |V|$ (Alg. 3–4) or $\mathcal{O}(3\Delta - 1) = \mathcal{O}(\Delta)$ using binary search over T or S_u and S_v directly (Alg. 5–6).

⁸As opposed to both S_u and S_v

E. Estimation of Local Graphlet Statistics

This section formulates the local graphlet estimation problem, then describes a computational framework for estimating local graphlet statistics. The experiments in Section IV-E demonstrate the effectiveness of these methods. Computing *local graphlet statistics* \mathbf{x}_i for an individual edge $e_i \in E$ (or node) in G (as opposed to the global graph G) is important with numerous potential applications. For instance, they can be used as powerful discriminative features $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ for improving statistical relational learning (SRL) tasks [53] such as relational classification [46], link prediction and weighting tasks (e.g., recommending items, friends, web sites, music, events, etc.) [54], detecting anomalies in graphs (e.g., detecting fraud, or attacks/malicious behavior in computer networks) [21], [22], among many others [44], [45], [55]. However, it is often expensive and computationally prohibitive to compute such local graphlet statistics in large networks. In this work, we propose a local graphlet estimation framework that derives accurate approximations of the local graphlet statistics while being orders of magnitude faster than exact methods.

Definition 3 (LOCAL GRAPHLET ESTIMATION): *Given a graph $G = (V, E)$ and an edge $e_i = (v, u) \in E$, the local graphlet estimation problem is to find*

$$\mathbf{x}_i = [x_1 \ x_2 \ x_3 \ \cdots \ x_6 \ x_7 \ \cdots]^T \quad (19)$$

where \mathbf{x}_i is an approximation of the exact local graphlet statistics denoted by \mathbf{y}_i for edge e_i such that $\mathbb{D}(\mathbf{x}_i \parallel \mathbf{y}_i)$ is minimized (i.e., $\mathbf{x}_i \approx \mathbf{y}_i$) as well as the computational cost associated with the estimation.

The aim of the *local graphlet estimation problem* is to compute a fast yet accurate approximation of the graphlet statistics (such as counts) centered at an individual edge. Note that $\mathbb{D}(\mathbf{x}_i \parallel \mathbf{y}_i)$ in Definition 3 can be any loss function. Instead of approximating all graphlets up to size k , one may relax the above problem to estimate a single graphlet pattern $G_k \in \mathcal{G}$ of interest (e.g., 4-cliques).

A general estimation framework for the *local graphlet estimation problem* is shown in Alg. 7. In particular, Alg. 7 takes as input an edge e_i , a graph G or $\Gamma(e_i)$ (neighborhood subgraph of e_i), a sampling probability p_e , and it returns the graphlet feature vector $\mathbf{x}_i \in \mathbb{R}^\kappa$ for $e_i \in E$ where $\kappa = |\mathcal{G}|$. This generalization gives rise to a highly expressive unifying *local graphlet estimation framework* with many interchangeable components. For computing the local graphlet statistics in Alg. 7, any edge-centric method can be used such as PGD [13], [14], VCP [8], [23], Orca [28], or any future state-of-the-art approach. In this work, we use the PGD library [13] to compute the local graphlet statistics required for estimation. Moreover, the class of local graphlet approximation methods have many attractive properties such as unbiasedness and consistency.

We now discuss Alg. 7 in detail. In particular, Alg. 7 shows how to efficiently estimate all graphlets of size $k \in \{2, 3, 4\}$ for an edge $e_i \in E$. First, we compute T_e , S_u , and S_v in Lines 3-9. We then use these sets to derive all graphlets of size $k = 3$ exactly in constant time (Lines 10-11) as done in [13], [14]. Notice that these sets are computed exactly and up to this point,

Algorithm 7 A family of localized graphlet estimation (LGE) methods for accurate and unbiased estimation of *local graphlet properties*. Given an edge e_i or edge-induced subgraph $\Gamma(e_i)$ and a sampling probability p_e , the general approach returns the estimated graphlet feature vector $\mathbf{x} = \mathbf{x}_i$ for $e_i \in E$.

```

1  procedure LOCALGRAPHLESTIMATION( $\Gamma(e_i)$  or  $G, e_i, p_e$ )
2      Initialize variables
3      parallel for each  $w \in \Gamma(v)$  do
4          if  $w \neq u$  then  $S_v \leftarrow S_v \cup \{w\}$  and  $\Psi(w) = \lambda_1$ 
5      parallel for each  $w \in \Gamma(u)$  and  $w \neq v$  do
6          if  $\Psi(w) = \lambda_1$  then
7               $T_e \leftarrow T_e \cup \{w\}$  and set  $\Psi(w) = \lambda_3$  ▷ triangle
8               $S_v \leftarrow S_v \cup \{w\}$ 
9          else  $S_u \leftarrow S_u \cup \{w\}$  and set  $\Psi(w) = \lambda_2$  ▷ wedge
10          $x_2 = |T_e|$  ▷ triangles/3-cliques
11          $x_3 = (d_u + d_v - 2) - 2|T_e|$  ▷ 2-stars
12     parallel for each  $w \in T_e$  do
13         for  $j = 1, \dots, \lceil d_w \cdot p_e \rceil$  do
14             Select a vertex  $r \in \Gamma(w)$  via an arbitrary distribution F
15             if  $\Psi(r) = \lambda_3$  then Set  $x_4 \leftarrow x_4 + (d_w / \lceil d_w \cdot p_e \rceil)$  ▷ 4-clique
16             Set  $\Psi(w)$  to  $\lambda_4$ 
17          $x_5 = \binom{|T_e|}{2} - x_4$  ▷ chordal-cycles
18     parallel for each  $w \in S_u$  do
19         for  $j = 1, \dots, \lceil d_w \cdot p_e \rceil$  do
20             Select a vertex  $r \in \Gamma(w)$  via an arbitrary distribution F
21             if  $\Psi(r) = \lambda_1$  then set  $x_7 \leftarrow x_7 + (d_w / \lceil d_w \cdot p_e \rceil)$  ▷ 4-cycle
22             if  $\Psi(r) = \lambda_2$  then set  $x_6 \leftarrow x_6 + (d_w / \lceil d_w \cdot p_e \rceil)$  ▷ tailed-tri
23         Set  $\Psi(w)$  to 0
24     parallel for each  $w \in S_v$  do
25         for  $j = 1, \dots, \lceil d_w \cdot p_e \rceil$  do
26             Select a vertex  $r \in \Gamma(w)$  via an arbitrary distribution F
27             if  $\Psi(r) = \lambda_1$  then set  $x_6 \leftarrow x_6 + (d_w / \lceil d_w \cdot p_e \rceil)$  ▷ tailed-tri
28         Set  $\Psi(w)$  to 0
29          $x_8 = \binom{|S_v|}{2} + \binom{|S_v|}{2} - x_6$  ▷ 3-stars
30          $x_9 = (|S_v| \cdot |S_u|) - x_7$  ▷ 4-paths
31     return  $\mathbf{x}$ , where  $x_k$  is the estimate of graphlet  $G_k$  for  $e_i$ 

```

corresponds exactly to the exact algorithm given in [13]. Next, we estimate 4-cliques in Lines 12-16. In particular, Line 12 searches each vertex $w \in T_e$ in parallel. Given $w \in T_e$, we sample a neighbor $r \in \Gamma(w)$ with probability p_e accordingly to an arbitrary weighted/uniform distribution F. Then, we check if r is of type λ_3 (from Line 7), as this indicates that r also participates in a triangle with $e = (v, u)$, and since $r \in \Gamma(w)$, then $\{v, u, w, r\}$ is a 4-clique. Line 16 ensures that the same 4-clique is not counted twice. Chordal-cycles are estimated in Line 17. Further, 4-cycles are estimated in Lines 18-23 as well as a fraction of the tailed-triangles. The other tailed-triangles are estimated in Lines 24-28. Finally, the remaining graphlets are estimated in $o(1)$ time (Lines 29-30) using knowledge from the previous steps. Notice that if $p_e = 1$ and sampling is performed without replacement, then Alg. 7 reverts back to the exact method proposed in [13], [14].

We can also use the Hoeffding's inequality [56] to obtain estimates with provable error bounds. For instance, if we replace $\lceil d_w \cdot p_e \rceil$ everywhere in Alg. 7 with $\lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$ then the error rate of the estimate is at most ϵ with confidence at least $1 - \delta$ [57]. The error rate ϵ and confidence level δ are specified

by the user and given as input into Alg. 7 (instead of p_e).

Complexity: The computational complexity is summarized in Table III. Note that just as before, we only need to compute a few graphlets and can directly obtain the others in constant time. To compute all local graphlet statistics for a given edge, it takes: $\mathcal{O}(\Delta_{\text{ub}}(|S_u| + |S_v| + |T_e|))$ where Δ_{ub} is the maximum degree from any vertex in S_v , S_u , and T_e . Alternatively, we can place an upper bound Δ_{ub} on the number of neighbors searched from any vertex in S_v , S_u , and T_e . This can drastically reduce the time. The intuition is that for vertices with large neighborhoods we only need to observe a relatively small (but representative) fraction of it to accurately extrapolate to the unobserved neighbors and their structure.

TABLE III
COMPUTATIONAL COMPLEXITY

Graphlet	Global	Local
4-clique	$\mathcal{O}(K\Delta T_{\max})$	$\mathcal{O}(\Delta_{\text{ub}} \cdot T_e)$
4-cycle	$\mathcal{O}(K\Delta S_{\max})$	$\mathcal{O}(\Delta_{\text{ub}} \cdot S_u)$
tailed-tri	$\mathcal{O}(K\Delta S_{\max})$	$\mathcal{O}(\Delta_{\text{ub}} \cdot (S_u + S_v))$
all	$\mathcal{O}(K\Delta(S_{\max} + T_{\max}))$	$\mathcal{O}(\Delta_{\text{ub}}(S_u + S_v + T_e))$

F. Discussion

The proposed family of localized graphlet estimation methods easily generalize to graphlets of arbitrary size by replacing the definition of an edge-centric neighborhood with the more general and suitable notion of an edge ℓ -neighborhood:

$$\Gamma_\ell(v, u) = \left\{ w \in V \setminus \{v, u\} \mid D(v, w) \leq \ell \vee D(u, w) \leq \ell \right\}$$

where $\Gamma_\ell(v, u)$ represents the set of vertices with distance less than or equal to ℓ from $e = (v, u) \in E$. Thus, we set $\ell = 1$ for graphlets of size $k \leq 4$, and $\ell = 2$ for graphlets of size $k = 5$, and so on. Investigating the methods for graphlets of size 5 and above is left for future work. The LGE framework also naturally allows for both uniform and weighted sampling designs, and has many other interchangeable components as well. Note that if the total number of edges is unknown (due to streaming, problem constraints, or other issues), then Alg. 1 is easily adapted, e.g., one may simply specify the number of graphlets to sample (instead of the fraction of graphlets to sample denoted by ϕ in Alg. 1). Unlike existing work, the proposed LGE methods are naturally amenable to streaming graphs and for graphs too large to fit into memory. For instance, we do not need to read the entire graph into memory, as long as there is an efficient way to obtain the ℓ -neighborhood subgraph $\Gamma(e_i)$ required for estimation.

In this work, we leveraged PGD [13] to count the graphlets in the sampled neighborhood subgraph. However, the proposed estimation framework is flexible for use with other exact subgraph counting algorithms including Orca [28], VCP [8], [23], or any future state-of-the-art approach. Therefore, the fastest state-of-the-art enumeration approach can always be used by the framework (e.g., Step 3 in Fig. 1) to speedup the estimation even further. In other words, the framework is independent of the exact enumeration approach used.

In the interest of space and to keep the presentation simple, we have left out several details on performance enhancement that we have in our implementation. To give a small example, we use an adjacency matrix structure for small graphs in order to facilitate $o(1)$ edge checks. For larger graphs, we efficiently encode the neighbors of the top- k vertices with largest degree (and relabel to save space/time) for $o(1)$ graph operations. We use a fast $\mathcal{O}(d)$ neighborhood set intersection procedure, dynamically select local search procedures over T_e , S_u , and have many other optimization's throughout the code (bit-vector graph representation, etc.).

III. PARALLELIZATION

Estimation methods from the framework are parallelized via independent edge-centric graphlet computations over the selected set of edge-induced neighborhoods $\{\Gamma(e_i), \dots, \Gamma(e_K)\}$. The parallelization is described such that it could be used for both shared and distributed memory architectures⁹. The parallel constructs used are a worker task-queue and a global broadcast channel. Multi-threaded MPI is used for inter-machine communication. We assume each machine q has a queue and a copy of the graph¹⁰ shared among the set of local workers (processing units). For global graphlet statistics, the communication cost for a single worker is $\mathcal{O}(|\mathcal{G}|)$.

The main parallel loop can be viewed as a task generator that farms the next b edges out to a worker, which then computes the graphlets centered at each of the b edge neighborhoods. Edge neighborhoods are dynamically partitioned to workers by “hardness” where the most difficult edge neighborhood is assigned to the first worker, the second most difficult is assigned to the second worker, and so on. This ensures we avoid common problems present in other approaches such as the curse of the last reducer [58] (due to the power-law observation [59]).

The existing state-of-the-art estimation methods are based on sequential algorithms which are inherently slow, difficult to parallelize, and have t dependent parts due to implementation issues, among others. Furthermore, our edge-centric parallel estimation method provides better load balancing (compared to vertex-based approaches). It is straightforward to see that if $N < M$, then our approach requires less computations per-edge than per-vertex since:

$$X_i = \sum_{e \in E} X_i(e) = \sum_{v \in V} X_i(v). \quad (20)$$

Parallelizing via edge-induced neighborhoods provides much better load balancing for real-world sparse graphs that follow a power-law. The time taken to count graphlets for each edge obeys a power-law with only a few edges taking much longer than the others (as observed in [59]). In addition, each $\Gamma(e)$ graphlet computation may be easily split into t independent tasks, e.g., k -cliques (Alg. 3), cycles (Alg. 4), solving the linear

⁹In the context of message-passing and distributed memory parallel computing, a node refers to another machine on the network (or bus) with its own set of memory, and multi-core CPUs, etc.

¹⁰For implementation on parallel computing architectures with limited memory, one only needs to transfer the set of edge-induced neighborhood subgraphs, which can be streamed if needed.

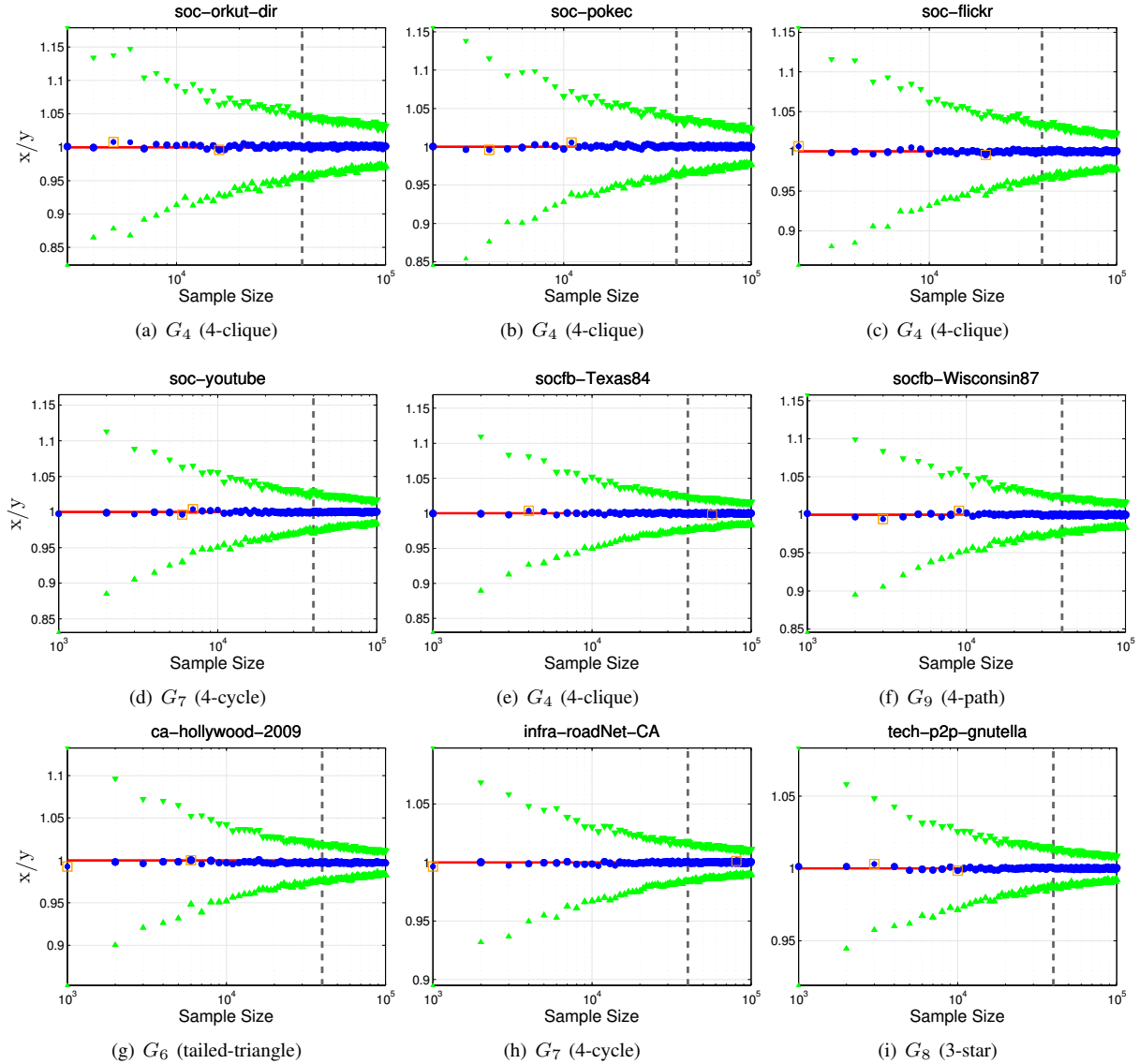


Fig. 3. Confidence bounds for a variety of graphlets. We used graphs from a variety of domains and types. Note that 4-cliques is often the most difficult to estimate and thus we have dedicated more results for these hard instances. The properties of the sampling distribution and convergence of the estimates are investigated as the sample size increases. The circle (\bullet) represents X/Y (y-axis) whereas \blacktriangle and \blacktriangledown are β_{lb}/Y and β_{ub}/Y , respectively. The square (\square) represents $\min/\max X/Y$. Dashed vertical line (grey) refers to the sample at 40K edges. Notably, the method has excellent accuracy even at this small sample size.

system, etc. Moreover, the edge-centric estimation methods are useful for situations where one might only be able to retrieve the (induced-) neighborhood of an edge due to privacy or data collection issues, etc. In addition, our approach does not require storage, knowledge, and preprocessing of the entire graph (as opposed to existing work). Other important properties include the neighborhood search order Π , the batch size b , and the dynamic assignment of jobs (for load balancing). As an aside, there have been a few distributed memory [60] and shared memory [61], [62] exact algorithms. However, these algorithms are based on older inefficient *exact enumeration* algorithms, whereas this work is focused on *estimation* methods. In addition, these approaches are all vertex-centric, as opposed to our edge-centric approach, and mainly focus on finding network motifs, *i.e.*, statistically significant subgraph patterns.

IV. EXPERIMENTS

In this section, we evaluate the empirical error and performance of the methods with extensive experiments on over 300 networks (real-world and synthetic) from 20+ domains with different structural characteristics. All data is available at NR [63]. Unless otherwise mentioned, we use PGD [13] to compute the exact graphlet counts for comparison.

A. Estimating Global Graphlet Statistics

We proceed by first demonstrating the effectiveness of the proposed methods for estimating the frequency of graphlets up to size $k = 4$. Given an estimated statistic X_i of an arbitrary graphlet $G_i \in \mathcal{G}$, we consider the relative error:

$$\mathbb{D}(X_i \parallel Y_i) = \frac{|X_i - Y_i|}{Y_i} \quad (21)$$

TABLE IV
ESTIMATES OF EXPECTED VALUE AND RELATIVE ERROR

Estimates of expected value and relative error using 100K samples. The graphlet statistic for the full graph is shown in the first column. β_{lb} and β_{ub} are 95% lower and upper bounds, respectively. Note M=million (mega), B=billion (giga).

	graph	Y	X	$\frac{ Y-X }{Y}$	β_{lb}	β_{ub}
4-CLIQUE ✕	ca-citeseer	18.7M	18.7M	0.0004	18.3M	19M
	ca-dblp-2012	16.7M	16.7M	0.0004	16M	17.3M
	soc-flickr	1.7B	1.7B	0.0003	1.7B	1.7B
	soc-friendster	9B	9B	0.0038	8.9B	9.1B
	soc-gowalla	6M	6M	0.0009	5.9M	6.2M
	soc-orkut	3.2B	3.2B	0.0016	3.1B	3.3B
	soc-pokec	42.9M	42.9M	0.0002	41.9M	43.9M
	socfb-Berkeley13	26.6M	26.6M	0.0007	26.2M	27M
	socfb-Indiana	60.1M	60.1M	0.0004	59.3M	61M
	socfb-MIT	13.6M	13.6M	0.0004	13.5M	13.8M
	socfb-OR	13.3M	13.3M	0.0005	13.1M	13.5M
	socfb-Texas84	70.7M	70.7M	0.0002	69.6M	71.8M
	socfb-UCLA	28.6M	28.6M	0.0005	28.2M	29M
	socfb-UCSB37	18.1M	18.1M	$<10^{-4}$	17.9M	18.4M
	socfb-UF	97.9M	97.9M	0.0001	96.5M	99.3M
	socfb-Ullinois	64M	63.9M	0.0008	63M	64.9M
	socfb-Wisconsin87	23M	23M	0.0011	22.7M	23.3M
	web-wikipedia2009	1.4M	1.4M	0.0004	1.3M	1.5M

where Y_i is the actual statistic (e.g., frequency) of G_i . Thus, this is a measure of how far the estimated statistic is from the actual graphlet statistic. Note X_i is the mean estimated value across 100 independent runs. The relative error indicates the quality of an estimated graphlet statistic relative to the magnitude of the exact statistic. Results for the most difficult graphlet (4-clique) are provided in Table IV for a wide range of graphs from various domains. Note the approach provides even better estimates for the other graphlets. Overall, the results demonstrate the effectiveness of the estimation methods as they have excellent empirical accuracy. Further, the estimation error for the disconnected graphlets is considerably smaller than the error for connected graphlets.

We also estimated univariate graphlet statistics beyond simple global counts such as the median, standard deviation, variance, IQR, Q1, Q3, and others. Overall, the methods are found to be accurate for many of the new graphlet statistics as shown in Fig. 4. For estimating the max 4-cliques, we also observed that selecting edges via the k -core distribution resulted in high accuracy at very low sample rates.

B. Confidence Bounds

Given an arbitrary graphlet $G_i \in \mathcal{G}$, we compute X_i using the estimators from the framework derived in Section II and construct confidence bounds for the unknown Y_i . Using the large sampling distribution, we derive lower and upper bounds such that

$$\beta_{lb} \leq Y_i \leq \beta_{ub} \quad (22)$$

$$\text{where } \beta_{lb} = X_i - z_{\alpha/2} \cdot \sqrt{\mathbb{V}[X_i]} \quad (23)$$

$$\text{and } \beta_{ub} = X_i + z_{\alpha/2} \cdot \sqrt{\mathbb{V}[X_i]} \quad (24)$$

The estimates X_i and $\mathbb{V}(X_i)$ are computed using the equations of the unbiased estimators of counts and their variance. Thus, $\alpha = 0.05$ and $z_{\alpha/2} = z_{0.025} = 1.96$ for a 95% confidence interval for the unknown Y_i . This gives

$$X_i - 1.96\sqrt{\mathbb{V}[X_i]} \leq Y_i \leq X_i + 1.96\sqrt{\mathbb{V}[X_i]} \quad (25)$$

Further, the sample size needed is $K = (z_{\alpha/2} \cdot \sqrt{\mathbb{V}[X_i]} / \alpha/2)^2$.

	mean	median	std	var	iqr	q1	q3
0.1	0.0002	0.0009	0.0009	0.002	0.010	0.015	0.0046
0.01	0.0026	0.0167	0.0125	0.025	0.003	0.020	0.0051

Fig. 4. Estimation error for a variety of univariate statistics for the local 4-clique graphlet distribution. These results are from socfb-MIT and thus even a sample size of 1% is small.

The 95% upper and lower bounds (i.e., β_{ub} and β_{lb}) for the 4-clique are shown in Table IV (other graphlet results were removed due to space). In all cases, the actual graphlet statistics lie inside the error bounds, $\beta_{lb} \leq Y_i \leq \beta_{ub}$. Fig. 3 investigates the properties of the sampling distribution as the sample size increases. The circle (blue) in Fig. 3 represents the fraction X_i/Y_i . Further, β_{lb}/Y_i and β_{ub}/Y_i are represented in Fig. 3 by Δ and ∇ , respectively. The key findings are summarized below.

- The sampling distribution is centered and balanced over the actual graph statistic (represented by the red line).
- Upper and lower bounds always contain the actual value.
- As the sample size increases, the bounds *converge* to the actual value of the graphlet statistic. The *estimated* variance decreases as k grows larger.
- Confidence bounds are within 5% of the actual for all graphs and subgraph patterns.
- Thus, the sampling distribution of the estimation framework has many attractive properties including unbiased estimates for all subgraph patterns and low variance even for very small sample sizes (and variance decreases as a function of the sample size).

Let $\mathbb{P}(\beta_{lb} \leq Y \leq \beta_{ub})$ be the exact coverage probability of our bounds. We observe that the confidence bounds are tight (for all subgraph patterns) and holds to a good approximation that is within $5\% \pm$ of the actual value for all 300+ graphs.

C. Estimating Graphlet Frequency Distributions (GFD)

We investigate the methods for estimating the graphlet frequency distributions (GFD) from a wide variety of networks with different

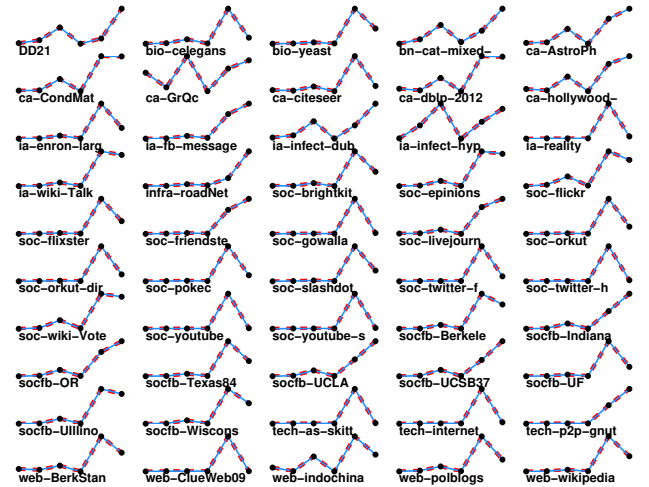


Fig. 5. Estimated GFD is indistinguishable from the actual (larger dotted red line), even across a wide variety of graphs with fundamentally different structural characteristics. The y-axis is the normalized 4-vertex graphlet counts $\mathbf{x}' = \mathbf{x} - \min(\mathbf{x}) / \max(\mathbf{x}) - \min(\mathbf{x})$ where \mathbf{x} is the vector of graphlet counts. Nevertheless, similar results were found for other graphlet sizes and GFD variants.

TABLE V
GFD ESTIMATES FOR A WIDE VARIETY OF SPARSE REAL-WORLD NETWORKS

All graphlet estimates have less than 10^{-3} relative error and those with relative error $<10^{-4}$ are highlighted. The KS test was used to test for significance of differences between the estimated and true distributions, and the test results show that they are significantly similar with 99% confidence (p-value < 0.01). Exact graphlet counts are computed using PGD [13] for comparison.













Graph	$ E $							KS-Stat.
ca-AstroPh	196.9K	0.010	0.016	0.193	0.001	0.324	0.455	$<10^{-4}$
ca-MathSciNet	820.6K	0.001	0.003	0.077	<0.001	0.461	0.457	$<10^{-4}$
ia-email-EU	54.3K	<0.001	0.001	0.031	<0.001	0.715	0.252	0.0005
ia-enron-large	180.8K	<0.001	0.004	0.060	0.001	0.716	0.219	$<10^{-4}$
rt-retweet-crawl	2.2M	<0.001	<0.001	<0.001	<0.001	0.898	0.101	$<10^{-4}$
soc-douban	327.1K	<0.001	<0.001	0.012	<0.001	0.436	0.552	0.0005
soc-youtube-s	2.9M	<0.001	<0.001	0.002	<0.001	0.982	0.016	0.0003
soc-flickr	3.1M	0.003	0.020	0.132	0.010	0.477	0.358	0.0007
soc-twitter-higgs	14.8M	<0.001	<0.001	0.003	<0.001	0.972	0.024	$<10^{-4}$
soc-friendster	1.8T	<0.001	<0.001	0.009	<0.001	0.400	0.590	$<10^{-4}$
socfb-Ullinois	1.2M	0.001	0.005	0.071	0.002	0.499	0.422	0.0001
socfb-Indiana	1.3M	0.001	0.006	0.089	0.003	0.300	0.600	$<10^{-4}$
socfb-Penn94	1.3M	<0.001	0.002	0.039	0.001	0.652	0.304	$<10^{-4}$
socfb-Texas84	1.5M	<0.001	0.002	0.043	0.001	0.667	0.287	0.0007
tech-internet-as	85.1K	<0.001	<0.001	0.005	<0.001	0.963	<0.001	0.0003

TABLE VI

GFD ESTIMATES FOR DENSE SYNTHETIC GRAPHS FROM DIMACS

GFD estimates for dense synthetic graphs from the DIMACS NP-Hard Problem Challenge described in [64], [65]. All graphlet estimates have less than 10^{-3} relative error and those with relative error $<10^{-4}$ are highlighted. The KS test was used to test for significance of differences between the estimated and true distributions, and the test results show that they are significantly similar with 99% confidence (p-value < 0.01).

Graph	$ E $							KS-Stat.
C2000-5	999.8K	0.026	0.158	0.316	0.079	0.105	0.316	$<10^{-4}$
C4000-5	4M	0.026	0.158	0.316	0.079	0.105	0.316	$<10^{-4}$
p-hat1500-1	284.9K	0.004	0.047	0.218	0.048	0.190	0.494	0.0010
johnso32-2-4	107.8K	0.446	0.428	0.066	0.033	0.023	0.005	$<10^{-4}$
brock800-3	207.3K	0.089	0.290	0.314	0.079	0.057	0.170	$<10^{-4}$
brock800-1	207.5K	0.090	0.291	0.314	0.079	0.057	0.170	$<10^{-4}$
sanr1000	250.5K	0.120	0.192	0.274	0.037	0.063	0.315	$<10^{-4}$
sanr400-0-5	39.9K	0.027	0.159	0.316	0.079	0.105	0.314	0.0005
DSJC500-5	62.6K	0.027	0.159	0.316	0.079	0.105	0.314	$<10^{-4}$
hamming6-4	704	0.001	0.025	0.140	0.106	0.240	0.487	$<10^{-4}$

structural characteristics including real-world and synthetic graphs. Exact graphlet counts are computed using PGD [13] for comparison. Strikingly, the estimated GFD from our approach almost perfectly matches the actual GFD (Fig. 5). Observe that the methods are evaluated by how well they estimate the entire exact GFD and thus Fig. 5 indicates that the proposed methods estimate all such induced subgraphs from Table I with excellent accuracy (matching the actual GFD in all cases). Results for a variety of sparse real-world graphs from different domains are shown in Table V. Overall, most graphlet estimates in Table V have relative error less than 10^{-4} . In all cases, we find no significant difference between the estimate and actual (Table V). In Table VI, we also report results on a standard collection synthetic benchmark graphs from the DIMACS NP-Hard Problem Challenge [64]. These graphs are *dense* and used extensively for evaluating NP-hard problems such as finding the largest clique as described in [64], [65]. Nevertheless, the estimators are shown in Table VI to be highly accurate across all graphlets and graphs. Notably, the KS-Statistic is very small for all graphs in both Table V and Table VI.

In addition, we also studied the effectiveness of the estimation methods on synthetic graphs from a variety of well-known graph models including Erdős-Rényi (ER) [68] graphs, the geometric random graphs (GEO) [66], scale-free Barabási-Albert (BA) [69] preferential attachment model, and the Kronecker graph model [67]. Results are reported in Table VII. The geometric random graph model networks GEO-15 to GEO-20 in Table VII are from the popular DIMACS 10 challenge [70], [71], whereas the Kronecker graphs 16-18 are from the Graph 500 supercomputer benchmark [72]. See [70]–[72] for more details. We also included a few other geometric random graphs (GEO 1-3) in Table VII; these graphs all have the same number of nodes but a different number of edges going from 100K to 200K in 50K increments. Overall, the estimation methods have excellent accuracy (with very small relative error) across all synthetic graphs from the various graph models as observed in Table VII. Similar results were observed using other graph models such as Chung-Lu and Block Erdos-Renyi and therefore were removed for brevity. Many graphs and results were also removed due to space.

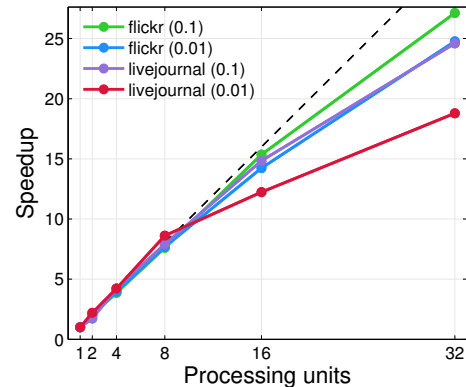








Fig. 6. Strong scaling results are observed across various sample sizes. See text for discussion.

TABLE VII

GFD ESTIMATES OF SYNTHETIC GRAPHS FROM A VARIETY OF WELL-KNOWN SYNTHETIC GRAPH MODELS

GFD estimates of synthetic graphs from a variety of well-known *synthetic graph models* including the geometric random graph model (GEO) [66], Kronecker graph model [67], Erdős-Rényi (ER) [68], and the Barabási-Albert (BA) [69] scale-free preferential attachment model. All graphlet estimates have less than 10^{-3} relative error and those with relative error $<10^{-4}$ are highlighted. The KS test was used to test for significance of differences between the estimated and true distributions, and the test results show that they are significantly similar with 99% confidence (p-value < 0.01). Exact graphlet counts are computed using PGD [13] for comparison.

Graph	V	E							KS-Stat.
GEO-1	1K	100K	0.051	0.114	0.343	0.006	0.080	0.407	$<10^{-4}$
GEO-2	1K	150K	0.067	0.137	0.365	0.007	0.074	0.349	$<10^{-4}$
GEO-3	1K	200K	0.092	0.166	0.383	0.008	0.065	0.286	$<10^{-4}$
GEO-15	32.7K	160.2K	0.057	0.111	0.326	0.003	0.048	0.455	0.0004
GEO-16	65.5K	342.1K	0.056	0.110	0.325	0.003	0.048	0.457	0.0008
GEO-20	1M	6.8M	0.056	0.111	0.325	0.003	0.048	0.456	$<10^{-4}$
Kron-16	55.3K	2.4M	<0.001	0.009	0.091	0.002	0.737	0.161	$<10^{-4}$
Kron-17	107.9K	5.1M	<0.001	0.006	0.076	0.002	0.765	0.151	0.0007
Kron-18	210.1K	10.5M	<0.001	0.005	0.063	0.001	0.790	0.141	$<10^{-4}$
ER-1	1K	100K	<0.001	0.019	0.149	0.037	0.199	0.596	$<10^{-4}$
ER-2	1K	150K	0.003	0.047	0.218	0.055	0.169	0.508	$<10^{-4}$
ER-3	1K	37.3K	<0.001	0.002	0.055	0.015	0.233	0.694	$<10^{-4}$
ER-4	10K	2.5M	<0.001	<0.001	0.038	0.009	0.238	0.714	$<10^{-4}$
ER-5	1M	5M	<0.001	<0.001	<0.001	<0.001	0.250	0.750	$<10^{-4}$
BA-1	1K	50.1K	0.002	0.021	0.153	0.021	0.278	0.525	$<10^{-4}$
BA-2	1K	100K	0.006	0.052	0.225	0.033	0.232	0.452	$<10^{-4}$
BA-3	1K	150K	0.015	0.092	0.275	0.042	0.199	0.377	$<10^{-4}$
BA-20-1	10K	199.6K	<0.001	<0.001	0.023	0.002	0.481	0.494	$<10^{-4}$
BA-40-1	10K	398.4K	<0.001	0.001	0.037	0.003	0.442	0.516	$<10^{-4}$
BA-80-1	10K	793.6K	<0.001	0.003	0.056	0.005	0.394	0.542	0.0006
BA-20-2	100K	1.9M	<0.001	<0.001	0.005	<0.001	0.657	0.338	$<10^{-4}$
BA-40-2	100K	3.9M	<0.001	<0.001	0.009	<0.001	0.601	0.390	$<10^{-4}$
BA-80-2	100K	5.1M	<0.001	<0.001	0.015	<0.001	0.607	0.378	0.0005







D. Scalability Results

This section investigates the scalability of the *parallel graphlet estimation methods*. We use speedup to evaluate the effectiveness of the parallel algorithm. Speedup is simply $S_p = \frac{T_1}{T_p}$ where T_1 is the execution time of the sequential algorithm, and T_p is the execution time of the parallel algorithm with p processing units. For the results in Fig. 6, we used a 4-processor Intel Xeon E5-4627 v2 3.3GHz CPU. Overall, the methods show strong scaling (See Fig. 6). Similar results were found for other graphs and sample sizes.

TABLE VIII

LOCAL GRAPHLET ESTIMATION RESULTS

For each graph problem, we report the relative error averaged over 500 randomly selected edges. These experiments use $p_e = 0.001$ (See Section II-E for more details). In addition to the high accuracy, the local graphlet estimation methods are between 900-1000 times faster, and thus fast and highly scalable.

graph	RELATIVE ERROR						KL	L1
								
soc-flickr	0.001	0.001	0.001	0.001	0.001	0.001	0.0001	$<10^{-4}$
bio-human-gene1	0.002	0.002	0.001	0.001	0.001	0.001	0.0004	$<10^{-4}$
tech-internet-as	0.0001	0.0001	0.0012	0.0002	0.001	0.0002	0.001	$<10^{-4}$
sc-nasasrb	0.004	0.004	0.001	0.002	0.003	0.002	0.004	0.001

E. Local Graphlet Estimation Experiments

This section investigates the accuracy, runtime, and scalability of the computational framework presented in Section II-E for estimating local graphlet statistics and distributions of individual graph elements such as an edge (or node, path, or subgraph) as opposed to estimating global graphlet statistics over the

entire graph G . Results are shown in Table VIII. Note that for simplicity, nodes are selected uniformly at random, thus F in Alg. 7 represents a uniform distribution over the neighbors.

F. Extremal Graphlet Estimation

Given a graph G , and a graphlet G_j of size k , the extremal (max) graphlet estimation problem is to find

$$Z_j = \max_{e_i \in \{e_1, \dots, e_m\}} [X_j(e_i)] \quad (26)$$

where Z_j is the maximum number of times graphlet G_j occurs at any edge $e_i \in E$ in G . The aim is to compute the maximum frequency that graphlet G_j occurs at any edge $e_i \in E$ in G . For this problem, we leverage the proposed LGE framework from Section II and bias the estimation method towards selecting a small set of edge J where G_j is most likely to appear at larger frequencies. The set of edges J are sampled via a graph parameter/distribution that appropriately biases selection of edges that are most likely to induce large quantities of the graphlet G_j . For relatively dense graphlets such as the k -clique (chordal-cycle/diamond, etc.), we investigated sampling edges from the largest k -core subgraphs. More specifically, instead of selecting edge neighborhoods via a uniform distribution F, our approach replaces F in Line 2 of Alg. 1 with a weighted distribution that biases the selection of edge neighborhoods towards those in large k -core subgraphs (i.e., edge neighborhoods centered at edges with large k -core numbers). Similarly, one may also use the triangle-core subgraphs if computed to obtain an estimate with lower error. Results demonstrate the effectiveness of this approach in Table IX. Strikingly, the above approach finds the

optimal solution (while taking only a fraction of the time) for many graphs as well as many of the k -vertex graphlets.

TABLE IX
RESULTS FOR EXTREMAL GRAPHLET ESTIMATION

Results for two of the proposed techniques for estimating the maximum frequency of an arbitrary induced subgraph centered at an edge in G . The results below use $p_i = 0.005$ and are for socfb-Texas. Similar results were found with different graphs and sampling probabilities, and thus, removed for brevity. Note that the runtime is the total time taken to estimate all graphlet statistics. Clearly, selecting edge neighborhoods using the weighted probability distribution based on k -core numbers gives much better estimates for the vast majority of statistics below. In particular, uniform does better only for estimating the maximum 3-stars centered at any edge in G . Nevertheless, both are orders of magnitude faster than the exact method. For instance, the k -core approach is 157x faster than the exact method (on average), whereas the uniform method is 185x faster. Note that the best result among the estimation methods is bold, whereas * indicates that the estimate returned by the method is optimal (that is, it matches the actual maximum returned by the exact PGD [13] algorithm).

Method	Speedup	Maximum graphlet counts					
KCORE	157x	45650*	3.85M*	26509*	50351*	19.51M	11.01M*
UNIFORM	185x	8172	22180	12112	24429	19.89M	3.35M
Exact	—	45650	3.85M	26509	50351	19.91M	11.01M

G. Comparison to Previous Work

To compare with previous estimation methods, we measure the time required by each method to obtain an estimate with relative error less than 0.01 (accuracy greater than 0.99). This ensures the estimation methods are compared fairly. Notice that it doesn't make sense to compare the accuracy of an estimation method without taking into account runtime, since the accuracy (relative error) of an estimation method increases (decreases) as a function of time (or work performed). Obviously, if time is not considered then one could simply use an exact method to achieve perfect accuracy. We also note that fixing the number of samples used by each method and measuring accuracy often leads to incorrect and misleading results since the accuracy depends on what each method calls a sample, and thus a method may use significantly more work than another.

TABLE X
RESULTS COMPARING LGE TO OTHER METHODS

Results for counting graphlets for four massive networks and one smaller graph (see text for discussion). For each method, we report the time required until the relative error is less than 0.01 (i.e., accuracy is greater than 0.99). A hyphen (—) indicates that the method did not terminate within 12 hours (with relative error less than 0.01). The best time for each problem instance is in bold.

graph	$ E $	Time in seconds				
		LGE	3-PATH	GUISE	GRAFT	PGD (exact)
soc-sinaweibo	261M	12.3	—	—	—	33359
web-ClueWeb09	7.81B	65.6	—	—	—	—
soc-friendster	1.81B	44.1	—	—	—	—
soc-twitter	1.20B	341.2	—	—	—	—
wiki-Talk	4.6M	0.0007	1.04	—	—	0.14

In Table X, we report the time each method takes to obtain an estimate with relative error less than 0.01 (accuracy greater than 0.99). As an aside, it is worth mentioning that existing work is fundamentally different than ours, both in techniques, as well

as in the estimation problems themselves. For instance, these methods estimate only simple global counts of graphlets, whereas the proposed class of LGE methods accurately estimates a wide variety of global and local statistics (including simple counts) and distributions. See Table II for a summary of the differences. Note that the 3-path sampling heuristic by Jha *et al.* [49] requires a lot more samples to obtain estimates with similar accuracy. In addition, that approach requires two different methods for estimating graphlet counts of size 4, and thus requires 2x the samples. In particular, we find that 3-path sampling, GUISE [47], and GRAFT [48] are unable to obtain accurate estimates within a reasonable amount of time. Furthermore, GUISE and GRAFT did not converge, even despite using millions of samples, which is consistent with recent findings [49], and especially true for the massive networks used in this work. See Table X. In some cases, the runtime of these methods even exceeded an exact graphlet algorithm, and thus not useful in practice. Notably, our method is not only more accurate at lower sampling rates, but significantly faster than these methods. For instance, on soc-flickr we are 8047x faster than the path-sampling heuristic. In some cases, we even find that our exact method is significantly faster than the 3-path heuristic (for instance, on wiki-talk and others). We also investigated selecting node-centric neighborhoods and other methods based on sampling graphlets directly, though, the accuracy was worse in all cases, and thus removed for brevity.

H. Applications

This section uses the novel statistics and estimators for real-time interactive graph visualization and exploratory analysis. Graphlet estimators are implemented in a web-based *interactive* visual graph mining platform [73] called GraphVis (Fig. 7). Across all experiments, the graphlet methods are fast and scalable taking <1 ms for 99% of the interactive queries and graphs, while also accurate (with no observable difference). Thus, the graphlet estimation methods are able to support *real-time* interactive queries for visual graph mining, exploration, and predictive modeling tasks (such as relational classification).

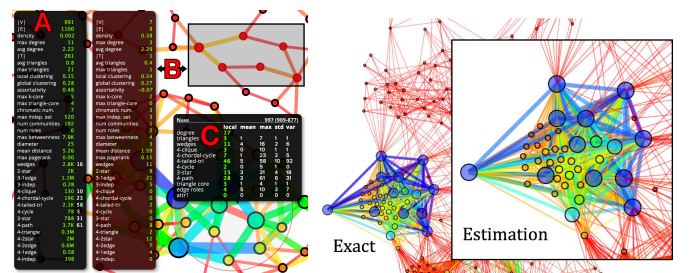


Fig. 7. Application of the estimation methods for real-time interactive graph mining and learning. **Left:** Edges and nodes from a power-grid network [63] are colored/weighted by the estimated local 4-path count. (A) Estimated global graphlet counts and other statistics; (B) summary statistics of the selected subgraph (rectangular region); and (C) local graphlet statistics (including frequency, mean, max, standard deviation, ...) of the selected edge. **Right:** Edges/nodes from tech-routers [63] are colored/weighted by the local 4-clique count. We observe that the visualizations using the exact and estimated graphlet features are strikingly similar, with trivial differences.

V. CONCLUSION

This paper proposed a general unbiased estimation framework called LGE for estimation of global and local graphlet properties (such as counts) in massive networks with billions of edges. The methods are shown to be accurate, fast, and scalable for both sparse and dense real-world and synthetic graphs of arbitrary size. Moreover, LGE has many interchangeable components and is effective for a wide variety of networks, applications, and domains, which have fundamentally different structural properties. We have shown that even for large networks with over a billion edges, one can compute graphlets fast and with low error. The newly introduced family of graphlet estimators greatly improves the scalability, flexibility, and utility of graphlets. Finally, the methods give rise to new opportunities and applications for graphlets (as shown in Section IV-H).

REFERENCES

- [1] L. Zhang, R. Hong, Y. Gao, R. Ji, Q. Dai, and X. Li, "Image categorization by learning a propagated graphlet path," *TNNLS*, vol. 27, no. 3, pp. 674–685, 2016.
- [2] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, vol. 11, pp. 1201–1242, 2010.
- [3] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *AISTATS*, vol. 5, 2009, pp. 488–495.
- [4] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
- [5] N. Pržulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?" *Bioinfo.*, vol. 20, no. 18, pp. 3508–3515, 2004.
- [6] T. Milenković and N. Pržulj, "Uncovering biological network function via graphlet degree signatures," *Cancer Informatics*, vol. 6, p. 257, 2008.
- [7] W. Hayes, K. Sun, and N. Pržulj, "Graphlet-based measures are suitable for biological network comparison," *Bioinfo.*, vol. 29, no. 4, 2013.
- [8] R. N. Lichtenwalter and N. V. Chawla, "Vertex Collocation Profiles: Subgraph Counting for Link Analysis and Prediction," in *WWW*, 2012, pp. 1019–1028.
- [9] L. Zhang, M. Song, Z. Liu, X. Liu, J. Bu, and C. Chen, "Probabilistic graphlet cut: Exploiting spatial structure cue for weakly supervised image segmentation," in *CVPR*, 2013, pp. 1908–1915.
- [10] M. Koyutürk, Y. Kim, U. Topkara, S. Subramaniam, W. Szpankowski, and A. Grama, "Pairwise alignment of protein interaction networks," *Journal of Computational Biology*, vol. 13, no. 2, pp. 182–199, 2006.
- [11] N. Pržulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, 2007.
- [12] J. Crawford and T. Milenković, "GREAT: Graphlet Edge-based network AlignmentT," in *BIBM*, 2015, pp. 220–227.
- [13] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *ICDM*, 2015, p. 10.
- [14] N. K. Ahmed, J. Neville, R. A. Rossi, N. Duffield, and T. L. Willke, "Graphlet Decomposition: Framework, Algorithms, and Applications," *KAIS*, pp. 1–32, 2016. [Online]. Available: <http://github.com/nkahmed/pgd>
- [15] L. Kovanen, M. Karsai, K. Kaski, J. Kertész, and J. Saramäki, "Temporal motifs in time-dependent networks," *JSTAT*, vol. 2011, no. 11, 2011.
- [16] Y. Hulovatyy, H. Chen, and T. Milenković, "Exploring the structure and function of temporal networks with dynamic graphlets," *Bioinformatics*, vol. 31, no. 12, pp. i171–i180, 2015.
- [17] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *PNAS*, vol. 101, no. 9, pp. 2658–2663, 2004.
- [18] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.
- [19] R. W. Solava, R. P. Michaels, and T. Milenković, "Graphlet-based edge clustering reveals pathogen-interacting proteins," *Bioinformatics*, vol. 28, no. 18, pp. i480–i486, 2012.
- [20] N. K. Ahmed, R. A. Rossi, T. L. Willke, and R. Zhou, "A higher-order latent space network model," in *AAAI PAIR*, 2017, pp. 1–7.
- [21] C. Noble and D. Cook, "Graph-based anomaly detection," in *SIGKDD*, 2003, pp. 631–636.
- [22] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *DMKD*, vol. 29, no. 3, pp. 626–688, 2015.
- [23] R. N. Lichtenwalter and N. V. Chawla, "Vertex Collocation Profiles: Theory, Computation, and Results," *SpringerPlus*, vol. 3, no. 1, pp. 1–27, 2014.
- [24] Y. Hulovatyy, R. W. Solava, and T. Milenković, "Revealing missing parts of the interactome via link prediction," *PloS one*, vol. 9, no. 3, 2014.
- [25] D. Marcus and Y. Shavitt, "RAGE—A rapid graphlet enumerator for large networks," *Computer Networks*, vol. 56, no. 2, pp. 810–819, 2012.
- [26] S. Wernicke and F. Rasche, "FANMOD: A tool for fast network motif detection," *Bioinformatics*, vol. 22, no. 9, pp. 1152–1153, 2006.
- [27] T. Milenković, J. Lai, and N. Pržulj, "GraphCrunch: A tool for large network analyses," *BMC Bioinformatics*, vol. 9, no. 1, p. 1, 2008.
- [28] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, vol. 30, no. 4, pp. 559–565, 2014.
- [29] D. Boyd and K. Crawford, "Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon," *Information, Communication & Society*, vol. 15, no. 5, pp. 662–679, 2012.
- [30] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159*, 2013.
- [31] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "Approximation of graph edit distance based on Hausdorff matching," *Pattern Recognition*, vol. 48, no. 2, pp. 331–343, 2015.
- [32] M. Bădoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," in *STOC*, 2002, pp. 250–257.
- [33] M. Henzinger, S. Krinninger, and D. Nanongkai, "An almost-tight distributed algorithm for computing single-source shortest paths," *arXiv*, 2015.
- [34] J. Pfeffer and K. M. Carley, "k-Centralities: Local approximations of global measures based on shortest paths," in *WWW*, 2012.
- [35] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," in *SIGCOMM*, 2006, pp. 27–40.
- [36] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *SIGKDD*, 2014, pp. 1446–1455.
- [37] Y. Lim and U. Kang, "MASCOT: Memory-efficient and accurate sampling for counting local triangles in graph streams," in *SIGKDD*, 2015.
- [38] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "DOULION: Counting triangles in massive graphs with a coin," in *SIGKDD*, 2009.
- [39] R. Pagh and C. E. Tsourakakis, "Colorful triangle counting and a mapreduce implementation," *IPL*, vol. 112, no. 7, pp. 277–281, 2012.
- [40] M. Rahman and M. Al Hasan, "Approximate triangle counting algorithms on multi-cores," in *Big Data*, 2013, pp. 127–133.
- [41] L. Roditty and U. Zwick, "Dynamic approximate all-pairs shortest paths in undirected graphs," *SICOMP*, vol. 41, pp. 670–683, 2012.
- [42] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SISC*, vol. 37, no. 5, p. 28, 2015.
- [43] U. Brandes and T. Erlebach, *Network Analysis: Methodological Foundations*. Springer, 2005, vol. 3418.
- [44] I. Bhattacharya and L. Getoor, "Entity resolution in graphs," *Mining graph data*, p. 311, 2006.
- [45] R. Rossi and N. Ahmed, "Role discovery in networks," *TKDE*, vol. 27, no. 4, pp. 1112–1131, 2015.
- [46] L. Getoor and B. Taskar, *Introduction to SRL*. MIT press, 2007.
- [47] M. Rahman, M. A. Bhuiyan, M. Rahman, and M. Al Hasan, "GUISE: A uniform sampler for constructing frequency histogram of graphlets," *KAIS*, vol. 38, no. 3, pp. 511–536, 2014.
- [48] M. Rahman, M. Bhuiyan, M. Al Hasan *et al.*, "GRAFT: An efficient graphlet counting method for large graph analysis," *TKDE*, vol. 26, no. 10, pp. 2466–2478, 2014.
- [49] M. Jha, C. Seshadhri, and A. Pinar, "Path sampling: A fast and provable method for estimating 4-vertex subgraph counts," in *WWW*, 2015.
- [50] D. G. Horvitz and D. J. Thompson, "A generalization of sampling without replacement from a finite universe," *JASA*, vol. 47, no. 260, pp. 663–685, 1952.
- [51] A. Roy, I. Mihailovic, and W. Zwaenepoel, "X-Stream: Edge-centric graph processing using streaming partitions," in *SOSP*, 2013, pp. 472–488.
- [52] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *ICDM*, 2015, pp. 1–10.
- [53] R. A. Rossi, L. K. McDowell, D. W. Aha, and J. Neville, "Transforming graph data for statistical relational learning," *JAIR*, vol. 45, no. 1, pp. 363–441, 2012.
- [54] N. N. Liu, L. He, and M. Zhao, "Social temporal collaborative ranking for context aware movie recommendation," *TIST*, vol. 4, no. 1, p. 15, 2013.
- [55] S. E. Schaeffer, "Graph clustering," *Comp. Sci. Rev.*, vol. 1, no. 1, 2007.

- [56] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *JASA*, vol. 58, no. 301, pp. 13–30, 1963.
- [57] N. K. Ahmed, T. L. Willke, and R. A. Rossi, "Estimation of local subgraph counts," in *IEEE Big Data*, 2016, pp. 586–595.
- [58] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *WWW*, 2011, pp. 607–614.
- [59] R. A. Rossi and R. Zhou, "Leveraging Multiple GPUs and CPUs for Graphlet Counting in Large Networks," in *CIKM*, 2016, pp. 1–9.
- [60] P. Ribeiro, F. Silva, and L. Lopes, "Parallel discovery of network motifs," *JPDC*, vol. 72, no. 2, pp. 144–154, 2012.
- [61] D. O. Aparício, P. M. P. Ribeiro, and F. M. A. da Silva, "Parallel subgraph counting for multicore architectures," in *ISPA*, 2014, pp. 34–41.
- [62] T. Wang, J. W. Touchman, W. Zhang, E. B. Suh, and G. Xue, "A parallel algorithm for extracting transcriptional regulatory network motifs," in *BIBE*, 2005, pp. 193–200.
- [63] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015, pp. 4292–4293. [Online]. Available: <http://networkrepository.com>
- [64] D. S. Johnson and M. A. Trick, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. AMS, 1996, vol. 26.
- [65] DIMACS Graphs: Benchmark Instances, <http://www.info.univ-angers.fr/pub/porumbel/graphs/>.
- [66] M. Penrose, *Random geometric graphs*. Oxford Univ. Press, 2003, no. 5.
- [67] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker Graphs: An Approach to Modeling Networks," *JMLR*, vol. 11, pp. 985–1042, 2010.
- [68] P. Erdős and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.
- [69] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.
- [70] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, "Graph partitioning and graph clustering," in *10th DIMACS Implementation Challenge Workshop*, 2012.
- [71] M. Holtgrewe, P. Sanders, and C. Schulz, "Engineering a scalable high quality graph partitioner," in *IPDPS*, 2010, pp. 1–12.
- [72] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, "Introducing the Graph 500," *Cray Users Group (CUG)*, 2010.
- [73] N. K. Ahmed and R. A. Rossi, "Interactive visual graph analytics on the web," in *ICWSM*, 2015.



Ryan A. Rossi is a machine learning research scientist at Adobe Research. His research lies in the fields of machine learning; and spans theory, algorithms, and applications of large complex relational (network/graph) data from social and physical phenomena. Before joining Adobe Research, he had the opportunity to work at a number of industrial, government, and academic research labs including Palo Alto Research Center (Xerox PARC), Lawrence Livermore National Laboratory (LLNL), Naval Research Laboratory (NRL), NASA Jet Propulsion Laboratory (JPL)/California Institute of Technology, University of Massachusetts Amherst, among others. Ryan earned his Ph.D. and M.S. in Computer Science at Purdue University. He was a recipient of the National Science Foundation Graduate Research Fellowship (NSF GRFP), National Defense Science and Engineering Graduate Fellowship (NDSEG), the Purdue Frederick N. Andrews Fellowship, and Bilsland Dissertation Fellowship awarded to Outstanding Ph.D. candidates.



Rong Zhou is currently at Google. Prior to that, he was a Senior Researcher and Manager of the High-Performance Analytics area of the Interaction and Analytics Laboratory at PARC. His research interests include large-scale graph algorithms, heuristic search, machine learning, automated planning and parallel model checking. Dr. Zhou has published extensively in top journals and conferences in the field of artificial intelligence, and his work received two Best Paper Awards from the International Conferences on Automated Planning and Scheduling (ICAPS) in 2004 and 2005. Dr. Zhou is the co-chair of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR) 2008, the co-chair of the International Symposium on Combinatorial Search (SoCS) 2009, and the tutorial co-chair of ICAPS 2011. He holds 21 US and 14 international patents in the areas of parallel algorithms, planning and scheduling, disk-based search, and diagnosis. Dr. Zhou is the recipient of four Golden Acorn Awards from PARC. He currently serves on the Editorial Board of the Journal of Artificial Intelligence Research.



Nesreen K. Ahmed is a senior research scientist at Intel Labs. She received her Ph.D. from the Computer Science Department at Purdue University in 2015, and her M.S. in statistics and computer science from Purdue University in 2014. In 2018, she is the PC Chair of the IEEE Big Data Conference. Dr. Ahmed was a visiting researcher at Facebook, Adobe research, Technicolor, and Intel analytics. Her research interests in machine learning and data mining spans the theory and algorithms of large-scale machine learning, graph theory, and their applications in social and information networks. Dr. Ahmed has authored numerous papers/tutorials in top-tier conferences/journals. Her research was selected among the best papers of ICDM in 2015, BigMine in 2012, and covered by popular press such as the MIT Technology Review. She was selected by UC Berkeley among the top female rising stars in computer science and engineering in 2014. Dr. Ahmed holds 2 U.S. patents filed by Adobe, and coauthored the open source network data repository (<http://networkrepository.com>).