Estimation of Graphlet Statistics

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 significant speedups compared to the state-of-the-art, (*b*) parallel with nearly linear-speedups, (*c*) accurate with <1% relative error, (*d*) scalable and space-efficient for massive networks with billions of edges, and (*e*) flexible for a variety of real-world settings, as well as estimating macro and micro-level graphlet statistics (*e.g.*, counts) of both connected *and* disconnected graphlets. In addition, an adaptive approach is introduced that finds the smallest sample size required to obtain estimates within a given user-defined error bound. On 300 networks from 20 domains, we obtain <1% relative error for all graphlets. This is significantly 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, motifs, statistical estimation, unbiased estimators, subgraph counts, network motifs, motif statistics, massive networks, parallel algorithms, graph mining, graph kernels, machine learning.

1 Introduction

Graphlets are *induced subgraphs*[1] and are important for many predictive and descriptive modeling tasks [1], [2], [3]. More recently, graphlets have been used to solve important and challenging problems in a variety of disciplines including image processing and computer vision [4], [5], bioinformatics [6], [7], and cheminformatics [8]. Unfortunately, the application and general use of graphlets remains severely limited to a few specialized problems/domains where the networks are small enough to avoid the scalability and performance limitations of existing methods. For instance, Shervashidze *et al.* [7] takes hours to count motifs on small biological networks (*i.e.*, few hundreds/thousands of nodes/edges) and uses such counts as features for graph classification [6]. Thus, this work provides a foundation for using graphlets to solve countless other important and unsolved problems, especially those with data that is large, massive, or streaming, as well as those with space- or time-constraints (real-time settings, interactive queries).

In many applications, finding an 'approximate' answer is usually sufficient where the extra cost and time in finding the exact answer is often not worth the extra accuracy. The recent rise of Big Data [9] has made approximation methods even more important and critical [10], especially for many practical applications [11], [12], [13], [14], [15]. More recently, approximation methods have been proposed for numerous important problems including triangle counting [16], [17], [18], [19], [20], shortest path problems [13], [21], finding max cliques [22], and many others.

This work aims to overcome the above limitations to make graphlets more accessible to other applications/domains with much larger graphs. In particular, this work proposes a general estimation framework for computing unbiased estimates 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 macro and micro-level graphlet statistics for both connected and disconnected graphlets. Moreover, the estimation framework is also scalable to massive networks with billions of edges and nodes. We also propose an approach for automatically determining the appropriate sample size for estimating various graphlet statistics within a given error bound. Parallel methods are introduced for each of the proposed techniques. Furthermore, a number of important machine learning tasks are likely to benefit from the proposed methods, including graph anomaly detection [23], [24], entity resolution [25], as well as features for improving community detection [26], role discovery [27], and relational classification [28].

Summary of contributions. The key contributions of this work are as follows:

- **Novel graphlet estimation framework and algorithms:** A general unbiased edge-centric estimation framework is proposed for approximating macro and micro graphlet 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.
- **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.
- **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

- *R. A. Rossi and R. Zhou are with Palo Alto Research Center (Xerox PARC), 3333 Coyote Hill Rd, Palo Alto, CA 94304*
  Email: {rrossi, rzhou}@parc.com
- *N. K. Ahmed is with Intel Labs, 3065 Bowers Ave, Santa Clara, CA 95052*
  Email: Nesreen.K.Ahmed@intel.com

1. The terms graphlet and induced subgraph are interchangeable.

also derived and shown to be tight (see Section 6.2).

- **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 wide variety of graphs from 20 domains.
- **Adaptive estimation:** While existing work requires the number (or proportion) of samples to be given as input, we instead introduce an approach that automatically determines the number of samples required to obtain estimates within a given error bound. Thus, this approach effectively balances the trade-offs between accuracy, time, and space.
- **Full spectrum of graphlets and novel sufficient statistics:** Our algorithms provide efficient computation of the full spectrum of graphlets including both connected and disconnected graphlets. Existing work has mainly focused on connected graphlets [29], [30], [31], [32], despite the importance of disconnected graphlets. For instance, Shervashidze *et al.* [7] found that disconnected graphlets are *essential* for correct classification on some datasets (See [7] pp. 495 where disconnected graphlets lead to a 10% improvement in accuracy).
- **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.

The proposed *localized graphlet estimation* (LGE) framework is flexible and gives rise to many important estimation methods for approximating a wide range of graphlet statistics (*e.g.*, frequency of all k-vertex induced subgraphs) and distributions including (i) macro-level graphlet statistics for the graph G as well as (ii) micro-level statistics for individual edges. Furthermore, we also propose estimators for both connected *and* disconnected graphlet counts (as opposed to only connected graphlet counts). The framework naturally allows for both uniform and weighted sampling designs, and has many other interchangeable components as well.

TABLE 1: 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, whereas assortativity is denoted by r. Also, $|T|$ is the total number of triangles, $\mathbb{K}$ is the max k-core number, χ denotes the Chromatic number, whereas $\mathbb{D}$ denotes the diameter.

		Description	Comp.	ρ	Δ	$\bar{d}$	r	$\lvert T\rvert$	$\mathbb{K}$	χ	$\mathbb{D}$
	G_1	edge		1.00	1	1.0	1.00	0	1	2	1
	G_2	2-node-independent		0.00	0	0.0	0.00	0	0	1	∞
	G_3	triangle		1.00	2	2.0	1.00	1	2	3	1
	G_4	2-star		0.67	2	1.33	-1.00	0	1	2	2
	G_5	3-node-1-edge		0.33	1	0.67	1.00	0	1	2	1
	G_6	3-node-independent		0.00	0	0.00	0.00	0	0	1	∞
CONNECTED	G_7	4-clique		1.00	3	3.0	1.00	4	3	4	1
CONNECTED	G_8	chordal-cycle		0.83	3	2.5	-0.66	2	2	3	2
CONNECTED	G_9	tailed-triangle		0.67	3	2.0	-0.71	1	2	3	2
CONNECTED	G_{10}	4-cycle		0.67	2	2.0	1.00	0	2	2	2
CONNECTED	G_{11}	3-star		0.50	3	1.5	-1.00	0	1	2	2
CONNECTED	G_{12}	4-path		0.50	2	1.5	-0.50	0	1	2	3
DISCONNECTED	G_{13}	4-node-1-triangle		0.50	2	1.5	1.00	1	2	3	1
DISCONNECTED	G_{14}	4-node-2-star		0.33	2	1.0	-1.00	0	1	2	2
DISCONNECTED	G_{15}	4-node-2-edge		0.33	1	1.0	1.00	0	1	2	1
DISCONNECTED	G_{16}	4-node-1-edge		0.17	1	0.5	1.00	0	1	2	1
DISCONNECTED	G_{17}	4-node-independent		0.00	0	0.0	0.00	0	0	1	∞

2 Localized Estimation Framework

In this section, we propose a new family of graphlet estimation methods based on selecting a set of localized edge-centric neighborhoods $\{\mathbf{\Gamma}(e_1), \ldots, \mathbf{\Gamma}(e_K)\}$. 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. Moreover, the LGE framework is also flexible with many interchangeable components. As shown later in Section 6, LGE is useful for a wide variety of networks, applications, and domains (*e.g.*, biological, social, and infrastructure/physical networks), which have fundamentally different structural properties.

2.1 Preliminaries

Let $G = (V, E)$ be an undirected simple graph with $N = |V|$ vertices and $M = |E|$ edges. Sets are *ordered*. Given a vertex $v \in V$, let $\Gamma(v) = \{w \,|\, (v, w) \in E\}$ be the set of vertices adjacent to v in G. We also define d_v as the degree of $v \in V$, where the degree d_v of $v \in V$ is defined as the size of the neighborhood of v, *i.e.*, $d_v = |\Gamma(v)|$. Further, let $\Delta(G)$ be the maximum vertex degree in G. Let $\mathcal{G}^{(k)}$ denote the set of k-vertex subgraphs and $\mathcal{G} = \mathcal{G}^{(1)} \cup \mathcal{G}^{(2)} \cup \cdots \cup \mathcal{G}^{(k)}$. Given a set $U = \{u_1, ..., u_k\} \subset V$ of k vertices, we define a k-graphlet as any k-vertex induced subgraph $G_i = (U, E[U])$ where $G_i \subset \mathcal{G}^{(k)}$. Note that $E[U]$ is the set of edges between the vertices in U. This work focuses on estimating statistics of these induced subgraphs called graphlets.

It is important to distinguish between *connected* and *disconnected* graphlets (see Table 1). A graphlet is connected if there is a path from any node to any other node in the graphlet, otherwise it is disconnected. Table 1 summarizes the important and fundamental properties of all graphlets of size $k \in \{2, 3, 4\}$.

2.2 Objective

The goal of this work is to obtain fast and accurate estimates of a variety of *macro* and *micro* graphlet properties (including both single-valued network statistics as well as distributions, that is, multi-valued network statistics) for both connected and disconnected graphlets (See Table 1) that include: (a) frequency of graphlets $G_i \in \mathcal{G}$, for all $i = 1, 2, ...|\mathcal{G}|$ or frequency of a specific graphlet $G_i \in \mathcal{G}$, (b) graphlet frequency distributions (GFD) including the connected, disconnected, and combined GFD containing both. (c) univariate statistics such as mean, median, min, max, variance, Q1, Q3, etc. (d) probability distribution (PDF, CDF, CCDF) of a particular graphlet G_i.

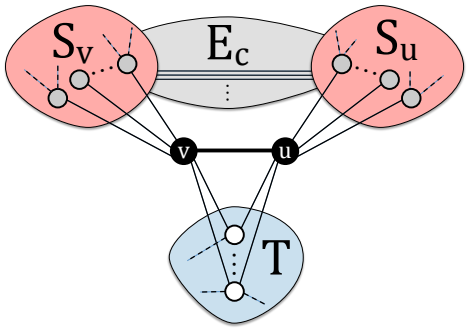


Fig. 1: Let T be the set of nodes completing a triangle with the edge $(v, u) \in E$, and let S_v and S_u be the set of nodes that form a 2-star with v and u, respectively. Note that $S_u \cap S_v = \emptyset$ by construction and $|S_u \cup S_v| = |S_u| + |S_v|$. Further, let E_c be the set of edges that complete a cycle (of size 4) *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 $(r \cap S_u) \cup (s \cap S_v) = \emptyset$, that is, r is not adjacent to u ($r \notin \Gamma(u)$) and s is not adjacent to v ($s \notin \Gamma(v)$).

Despite the practical importance of these graphlet properties, this work is the first to propose and investigate many of these novel problem variants. In addition, this paper proposes and investigates methods for estimating not only connected graphlet counts, but also disconnected graphlet counts, as well as a variety of other important and novel graphlet properties beyond simple counts. Disconnected graphlets are vital for many problems, including graph and node classification, graph kernels, among others [7], [33]. For instance, Shervashidze *et al.* [7] find that disconnected graphlets are essential for correct classification on some datasets (See [7] pp. 495). Nevertheless, this leads us to define and investigate many novel problem variants. For instance, this work estimates the three possible GFD variations (connected, disconnected, and a GFD containing both).

While previous methods have been proposed for computing counts of connected induced subgraphs, we instead propose a unifying unbiased estimation framework that is robust and generalizes for connected *and* disconnected graphlets. In addition to the above limitation, previous work has also been limited to simple count statistics (frequency or proportion of a graphlet). This work introduces and investigates estimators for a number of important macro *and* micro-level graphlet statistics (*e.g.*, graphlet counts for individual edges, as well as the total frequency of a graphlet in G). In particular, the framework gives rise to graphlet estimation methods that are fast and accurate for both (a) macro and (b) micro-level graphlet properties including (i) single-valued graphlet statistics and (ii) distributions (multiple-valued network statistics). For each of these new problem variants, we introduce fast, parallel, and accurate techniques and demonstrate their effectiveness and utility on a variety of networks.

Algorithm 1 Edge-centric graphlet estimators

Input:

a graph $G = (V, E)$

a sample size K, or sample probability p

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

TABLE 2: Qualitative and quantitative comparison of the two main classes of graphlet estimation methods, namely, *direct graphlet sampling* and *localized graphlet estimation* (LGE) methods. Direct methods are those that sample k-vertices directly and retrieve the graphlet induced by that subset. This work proposes the family of *localized graphlet estimation* (LGE) methods that select (sample) localized ℓ-neighborhoods for estimation. The first two columns refer to the type of graphlets estimated (connected and/or disconnected graphlets). The next six columns refer to the *macro* and *micro* graphlet estimation problems. In particular, columns 3-5 refer to the *macro* graphlet statistics and distributions estimated by the methods (counts, GFD, and others such as extremal stats./distributions, etc.), whereas columns 6-8 refer to the *micro* 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). "Massive 1B+" holds true if the methods is capable of handling massive graphs of 1 billion or more edges. "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.

				MACRO			MICRO			COMPUTATIONAL							
	Method	Connected	Disconnected	Counts	GFD	Others	Counts	GFD	Others	Parallel	Space efficient	Massive 1B+	Streaming	Position-aware	Sparse & dense	Parameter-free	All graphlets
DIRECT	SHERV. *et al.* [7]	✓	✓		✓						✓		✓				
	GUISE [29]	✓			✓												
	GRAFT [30]	✓			✓												
	3-PATH SAMP. [31]	✓		✓													
LGE	UNIFORM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	KCORE	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓		✓	✓

Table 2 summarizes existing related methods as well as our proposed approach according to the types of graphlets computed (connected and/or disconnected), the macro and micro-level graphlet statistics estimated by each (including single-valued statistics and distributions), as well as computational and algorithmic aspects/features.

2.3 Class of Localized Graphlet Estimation Algorithms

The general unbiased graphlet estimation framework is based on sampling (or selecting) edge-induced neighborhoods $\Gamma(e)$. Given an edge $e = (u, v) \in E$, let $\Gamma(e)$ denote the edge neighborhood of e defined as:

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

where $\Gamma(u)$ and $\Gamma(v)$ are the neighbors of u and v, respectively. The (explicit) edge-induced neighborhood is $\mathbf{\Gamma}_e = G(\{\Gamma(v) - u\} \cup \{\Gamma(u) - v\})$. The subgraph $\mathbf{\Gamma}(e)$ consists of the set of vertices adjacent to v or u (non-inclusive) and all edges between that set.

In particular, given an edge-centric neighborhood $\Gamma(e)$, we compute all graphlets G_i that include e. Intuitively, an edge neighborhood $\Gamma(e)$ is sampled with some probability from the set of all edge-induced neighborhoods (See Alg 1). Using the 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 $i = 1, ..., |\mathcal{G}|$. Let us note that edge neighborhoods may be selected uniformly at random or by an arbitrary

Algorithm 2 Family of edge-centric parallel *localized graphlet estimation* (LGE) algorithms

```
1  procedure LocalizedGraphletEst(G, J)
2    parallel for each e = (v, u) ∈ J in order do
3      Reset T_e = ∅ and S_u = ∅
4      for w ∈ Γ(v) do
5        if w ≠ u then Ψ(w) = λ1
6      for w ∈ Γ(u) do
7        if w = v then continue
8        if Ψ(w) = λ1 then
9          T_e ← T_e ∪ {w} and set Ψ(w) = λ3          ▷ triangle
10       else S_u ← S_u ∪ {w} and set Ψ(w) = λ2       ▷ wedge
11     Update unrestricted connected counts via Eq. 2–5 and
       unrestricted disconnected counts via Eq. 6–9
12     C3  ±  |Te|   ▷ Note ± is the addition sum C3 = C3 + |Te|
13     C4  ±  C4(e) = (du + dv − 2) − 2|Te|   ▷ equiv. |Su| + |Su|
14     C5  ±  C5(e) = n − C4(e) + |Te| − 2
15     C7  ±  C7(e) = Clique(Ψ, Te)                   ▷ in parallel
16     C10 ±  C10(e) = Cycle(Ψ, Su)                   ▷ in parallel
17   end parallel
18   Compute estimated graphlet counts X via Eq. 10-24
19   return X, where Xi is the estimate for graphlet Gi
```

weighted distribution F (as shown in Alg 1). For instance, edge neighborhoods may be sampled uniformly at random or according to an arbitrary weight/property such as k-core numbers, degrees, or any attribute of interest. Further, an edge 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. We have experimented with both on a few networks and there was no significant difference (using a fixed sampling probability p and number of trials S). Edge-centric graphlet decomposition algorithms also lend themselves for (parallel) implementation on both shared-memory and distributed-memory architectures (see Section 5).

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. Note that $S_u \cap S_v = \emptyset$ by construction and

$$S_u \cup S_v = \{\underbrace{w_1, \ldots, w_i}_{S_u}, \underbrace{w_{i+1}, \ldots, w_n}_{S_v}\}.$$

Thus, $|S_u \cup S_v| = |S_u| + |S_v|$. These quantities are computed in Line 6-10 of Alg 2. For further intuition, see Figure 1. Let us also note that $\mathbf{\Psi}(\cdot)$ is a hash table for checking edge existence in $o(1)$ time (see Alg 2). As an aside, this is an implementation detail and $\mathbf{\Psi}(\cdot)$ 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 $\mathbf{\Psi}(\cdot)$ 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 5, we mark the neighbors $\Gamma(v)$ of v as λ_1. Later in Line 9 a triangle is marked with λ_3, whereas Line 10 encodes a wedge as λ_2.

Moreover, given that all count variables are initialized to zero, Alg 2 maintains the *unrestricted connected graphlet counts*[2] (Eq. 2-5):

$$C_8 \pm \binom{|T_e|}{2} \tag{2}$$

$$C_9 \pm |T_e| \cdot \Big(|S_u| + |S_v|\Big) \tag{3}$$

$$C_{11} \pm \binom{|S_u|}{2} + \binom{|S_v|}{2} \tag{4}$$

$$C_{12} \pm |S_u| \cdot |S_v| \tag{5}$$

where C_8, C_9, C_{11}, and C_{12} are later used for computing chordal-cycles, tailed-triangles, 3-stars, and 4-paths in constant time, respectively. For clarity, C_i represents the unrestricted counts for graphlet $G_i \in \mathcal{G}^3$. Let us note that C_7 and C_{10} (4-cliques and 4-cycles, respectively) are computed in Alg 2 Line 15-16. Further, C_3, C_4, and C_5 are computed in Line 12-14 and represent the ($k = 3$)-graphlets of triangles, 2-stars, and 3-node-1-edge, respectively. Note that we refer to the unrestricted counts as the combinatorial counts that can be computed in constant time and using only the knowledge obtained from the quantities discussed above (*i.e.*, triangle and 2-star counts in the edge-centric neighborhood of an edge e).

Similarly, we compute the *unrestricted disconnected counts* (Eq. 6-9):

$$C_{13} \pm |S_u| \cdot (N - |\Gamma(u) \cup \Gamma(v)|) + |S_v| \cdot (N - |\Gamma(u) \cup \Gamma(v)|) \tag{6}$$

$$C_{14} \pm |T_e|(N - |\Gamma(u) \cup \Gamma(v)|) \tag{7}$$

$$C_{15} \pm \binom{N - |\Gamma(u) \cup \Gamma(v)|}{2} \tag{8}$$

$$C_{16} \pm M - |\Gamma(u) \setminus \{v\}| - |\Gamma(v) \setminus \{u\}| - 1 \tag{9}$$

In addition, we maintain C_3, C_4, C_5, C_7, and C_{10} (see Alg 2). Note that the $(k$–$1)$-graphlets are used to compute the k-clique/k-cycle counts directly. 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 from their counts via Eq. 10-13 as follows:

$$X_3 = W_3 \sigma_3 C_3 \tag{10}$$

$$X_4 = W_4 \sigma_4 C_4 \tag{11}$$

$$X_5 = W_5 \sigma_5 C_5 \tag{12}$$

$$X_6 = W_6 \cdot \left[\binom{n}{3} - X_3 - X_4 - X_5\right] \tag{13}$$

where X_3, X_4, X_5, X_6 are the estimated counts of the graphlets G_3, G_4, G_5, G_6 respectively, and W, σ are the weights used to fix the sampling bias.

2. Note $\pm$ is the addition assignment operator.
3. Recall the graphlet notation summarized in Table 1)

Similarly, the 4-vertex *connected graphlet* counts are estimated via Eq 14-19 as follows:

$$X_7 = W_7\sigma_7 C_7 \tag{14}$$
$$X_8 = W_8\sigma_8(C_8 - C_7) \tag{15}$$
$$X_9 = W_9(\sigma_9 C_9 - 4X_8) \tag{16}$$
$$X_{10} = W_{10}\sigma_{10} C_{10} \tag{17}$$
$$X_{11} = W_{11}(\sigma_{11} C_{11} - X_9) \tag{18}$$
$$X_{12} = W_{12}\sigma_{12}(C_{12} - C_{10}) \tag{19}$$

and the 4-vertex *disconnected graphlet* counts are estimated via Eq 20-24 as follows:

$$X_{13} = W_{13}(\sigma_{13} C_{13} - X_9) \tag{20}$$
$$X_{14} = W_{14}(\sigma_{14} C_{14} - 2X_{12}) \tag{21}$$
$$X_{15} = W_{15}(\sigma_{15} C_{15} - 6X_7 - 4X_8 - 2X_9 - 4X_{10} - 2X_{12}) \tag{22}$$
$$X_{16} = W_{16}(\sigma_{16} C_{16} - 2X_{15}) \tag{23}$$
$$X_{17} = W_{17} \cdot \left[\binom{n}{4} - \sum_{i=7}^{16} X_i \right] \tag{24}$$

where X_7–X_{17} are the estimated counts of the graphlets G_7–G_{17} respectively. Further, $\mathbf{W} \in \mathbb{R}^{\kappa}$ is a vector of weights defined as:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & \frac{1}{3} & \frac{1}{2} & 1 & 1 & \frac{1}{6} & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & 1 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{3} & 1 \end{bmatrix}^T \tag{25}$$

where each W_i is a scalar that aims to correct the bias for the induced subgraph G_i (See Table 1 to determine the corresponding induced subgraph for each $G_i \in \mathcal{G}$). However, $\mathbf{W}$ can be adapted to account for other known biases. Further, $\mathbf{p} \in \mathbb{R}^{\kappa}$ is a vector of sampling probabilities for all graphlets where p_i is the sampling probability of 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:

$$p_i = |J|/|E|$$

where 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 6. In 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.

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 compute them directly. However, in situations where memory is limited, then Alg. 5 and Alg. 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. See Section 2.5 for further details.

2.4 Error Analysis

Let $Y_i(e)$ be the total 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 $X = \sum_{e \in J} \frac{Y_i(e)}{\phi}$. $\mathbb{E}[X_i] = Y$ is an unbiased estimate. The proof is as follows.

Algorithm 3 Clique counts via neigh-iter.

```
procedure CLIQUE(Ψ, T_e)
    Set K_e ← 0
    parallel for each w ∈ T_e do
        for each r ∈ Γ(w) where Ψ(r) = λ_3 do set K_e ← K_e + 1
        Reset Ψ(w) to 0
    return K_e
```

Algorithm 4 Cycle counts via neigh-iter.

```
procedure CYCLE(Ψ, S_u)
    Set C_e ← 0
    parallel for each w ∈ S_u do
        for each r ∈ Γ(w) where Ψ(r) = λ_2 do set C_e ← C_e + 1
        Reset Ψ(w) to 0
    return C_e
```

$$\mathbb{E}[X_i] = \mathbb{E}\left[\sum_{e \in J} \frac{X_i(e)}{\phi}\right] = \sum_{e \in J} \mathbb{E}\left[\frac{X_i(e)}{\phi}\right] \tag{26}$$
$$= \sum_{e \in E} \frac{\mathbb{E}[\mathcal{I}_e]}{\phi} \cdot X_i(e) = \sum_{e \in E} \frac{X_i(e)}{\phi} \cdot \phi = Y \tag{27}$$

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

$$\mathbb{E}[(X_i - Y_i)^2] = \underbrace{\mathbb{V}[X_i]}_{\text{Variance}} + \underbrace{(\mathbb{E}[X_i] - Y_i)^2}_{\text{Bias}} \tag{28}$$

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*.

2.5 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 significantly smaller since for each edge $e = (v, u) \in J$, Alg. 2 computes only S_u[4] 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 $\{\Gamma(e_1), ..., \Gamma(e_K)\}$ up to $k = 4$ in:

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

Using K processing units (cores, workers), this reduces to $\mathcal{O}(\Delta(T_{\max} + S_{\max}))$.

Space-efficient algorithms are crucial when dealing with such massive networks. Thus, our method is designed to be space-efficient, and as we shall see requires significantly less space than existing approaches [29], [30], [32], [34], [35]. Space complexity of Alg 2 is $\mathcal{O}(N + 2\Delta - 1) = \mathcal{O}(N)$ using a hash table Φ of size $n = |V|$. However, this can be reduced to $\mathcal{O}(3\Delta - 1) = \mathcal{O}(\Delta)$ using binary search over T or S_u and S_v directly, e.g., see Alg. 5 and Alg. 6. Note that Alg 3–4 assumes

4. As opposed to both S_u and S_v

Algorithm 5 Clique counts restricted to searching T_e

```
1 procedure CLIQUERES(Ψ, T_e)
2   Set K_e ← 0
3   parallel for each vertex w_i in an ordering w_1, w_2, ··· of T_e do
4     for all each w_j ∈ {w_{i+1}, ..., w_{|T_e|}} in order do
5       if w_i ∈ Γ(w_j) via binary search then K_e ← K_e + 1      ▷ 4-clique
6   end parallel
7   return K_e
```

Algorithm 6 Cycle counts restricted to S_u and S_v

```
1 procedure CYCLERES(Ψ, S_u, S_v)
2   Set C_e ← 0
3   parallel for each w ∈ S_u do
4     for all r ∈ S_v do
5       if r ∈ Γ(w) via binary search then C_e ← C_e + 1      ▷ 4-cycle
6   end parallel
7   return C_e
```

a fast hash table-like data structure to efficiently check the existence of a neighbor.

2.6 Discussion

The 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:

$$\Gamma_\ell(v,u) = \Big\{ w \in V \setminus \{v,u\} \,|\, D(v,w) \leq \ell \vee D(u,w) \leq \ell \Big\}$$

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. 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 processing (for graphs to 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 $\mathbf{\Gamma}(e_i)$ required for estimation.

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 ops. We use a fast $O(d)$ 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.).

3 Estimating Micro Graphlet Counts

This section formulates the micro-level graphlet estimation problem, then derives a flexible computational framework. The experiments in Section 6.7 demonstrate the effectiveness of these methods. Computing *micro-level 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 $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ for improving statistical relational learning (SRL) tasks [36] such as relational classification [28], link prediction and weighting tasks (e.g., recommending items, friends, web sites, music, events, etc.) [37], detecting anomalies in graphs (e.g., detecting fraud, or attacks/malicious behavior in computer networks) [23], [24], among many others [25], [26], [27].

Problem. (Micro-level Graphlet Estimation) Given a graph $G = (V,E)$ and an edge $e_i = (v,u) \in E$, the *micro graphlet estimation problem* is to find

$$\mathbf{x}_i = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_6 & x_7 & \cdots & x_{17} \end{bmatrix}^T$$

where $\mathbf{x}_i$ is an approximation of the exact micro-level graphlet statistics denoted by $\mathbf{y}_i$ for edge e_i such that $\mathbb{D}\left(\mathbf{x}_i \parallel \mathbf{y}_i\right)$ is minimized (*i.e.*, $\mathbf{x}_i \approx \mathbf{y}_i$) as well as the computational cost associated with the estimation. Note that $\mathbb{D}\left(\mathbf{x}_i \parallel \mathbf{y}_i\right)$ can be any loss function. The aim of the *micro graphlet estimation problem* is to compute a fast approximation of the graphlet statistics (such as counts) centered at an individual edge.

Algorithm 7 Micro-level graphlet estimation framework

```
1  procedure MICROGRAPHLETESTIMATION(Γ(e_k) or G, e_k, p_e)
2    Initialize variables
3    parallel for each w ∈ Γ(v) do
4      if w ≠ u then S_v ← S_v ∪ {w} and Ψ(w) = λ_1
5    parallel for each w ∈ Γ(u) and w ≠ v do
6      if Ψ(w) = λ_1 then
7        T_e ← T_e ∪ {w} and set Ψ(w) = λ_3                    ▷ triangle
8        S_v ← S_v \ {w}
9      else S_u ← S_u ∪ {w} and set Ψ(w) = λ_2                  ▷ wedge
10   x_3 = |T_e|                                               ▷ triangles/3-cliques
11   x_4 = (d_u + d_v − 2) − 2|T_e|                            ▷ 2-stars
12   x_5 = n − (|S_v| + |S_u| + |T_e| − 2)                     ▷ 3-node-1-edge
13   x_6 = (N choose 3) − x_3 − x_4 − x_5                      ▷ 3-node-indep.
14   parallel for each w ∈ T_e do
15     for j = 1, ..., ⌈d_w · p_e⌉ do
16       Select a vertex r ∈ Γ(w) via an arbitrary distribution F
17       if Ψ(r) = λ_3 then Set x_7 ← x_7 + (d_w/⌈d_w · p_e⌉)   ▷ 4-clique
18     Set Ψ(w) to λ_4
19   x_8 = (|T_e| choose 2) − x_7                              ▷ chordal-cycles
20   parallel for each w ∈ S_u do
21     for j = 1, ..., ⌈d_w · p_e⌉ do
22       Select a vertex r ∈ Γ(w) via an arbitrary distribution F
23       if Ψ(r) = λ_1 then set x_10 ← x_10 + (d_w/⌈d_w · p_e⌉)  ▷ 4-cycle
24       if Ψ(r) = λ_2 then set x_9 ← x_9 + (d_w/⌈d_w · p_e⌉)   ▷ tailed-tri
25       if Ψ(r) = λ_4 then set ω ← ω + (d_w/⌈d_w · p_e⌉)
26     Set Ψ(w) to 0
27   parallel for each w ∈ S_v do
28     for j = 1, ..., ⌈d_w · p_e⌉ do
29       Select a vertex r ∈ Γ(w) via an arbitrary distribution F
30       if Ψ(r) = λ_1 then set x_9 ← x_9 + (d_w/⌈d_w · p_e⌉)   ▷ tailed-tri
31       if Ψ(r) = λ_4 then set ω ← ω + (d_w/⌈d_w · p_e⌉)
32     Set Ψ(w) to 0
33   x_11 = (|S_v| choose 2) + (|S_v| choose 2) − x_9           ▷ 3-stars
34   x_12 = (|S_v| · |S_u|) − x_10                             ▷ 4-paths
35   x_13 = |T_e| · [n − (|T_e| + |S_u| + |S_v| + 2)]          ▷ 4-node-1-tri
36   x_14 = (|S_u| + |S_v|) · [n − (|T_e| + |S_u| + |S_v| + 2)]  ▷ 4-node-2-star
37   x_15 = m − (|T_e| + d_u + d_v + 1) − ω                    ▷ 4-node-2-edge
38   x_16 = (n − ⌊|T_e| + |S_u| + |S_v| + 2⌋ choose 2)         ▷ 4-node-1-edge
39   x_17 = (n choose 4) − Σ_{i=7}^{16} x_i                    ▷ 4-node-indep.
40   return x, where x_i is the estimate of graphlet G_i for e_k
```

Instead of approximating all graphs 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 generalized and flexible framework for the *micro graphlet estimation problem* is given in Alg 7. In particular, Alg. 7 takes as input an edge e_i, a graph G or $\mathbf{\Gamma}(e_i)$ (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 flexible and expressive unifying framework and serves as a basis for investigating this novel graphlet estimation problem. Moreover, the class of micro graphlet approximation methods have many attractive properties such as unbiasedness, consistency, among others. The algorithm estimates micro graphlet properties including micro single-valued statistics and multi-valued distributions (for a given edge or set of edges).

Alg. 7 shows how to efficiently count 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. Afterwards, Lines 10-13 compute all graphlets of size $k = 3$ exactly. Next, we compute 4-cliques in Lines 14-18. In particular, Line 14 searches each vertex $w \in T_e$ in parallel. Given $w \in T_e$, we select 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. Finally, Line 18 ensures that the same 4-clique is not counted twice. Chordal-cycles are derived in Line 19. Further, 4-cycles are computed in Lines 20-26 as well as a fraction of the tailed-triangles. The remaining tailed-triangles are computed in Lines 27-32. As an aside, ω is also computed (Lines 20-32) and used for estimating G_{15} (Line 37). Finally, the remaining graphlets $\{x_{11}, \ldots, x_{17}\}$ are estimated in $o(1)$ time (Lines 33-39) using knowledge from the previous steps. Notably, Alg. 7 gives rise to an efficient exact method, *e.g.*, if $p_e = 1$ and selection is performed without replacement.

The computational complexity is summarized in Table 3. 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 micro-level graphlet statistics for a given edge, it takes: $\mathcal{O}\Big(\Delta_{\text{ub}}(|S_u| + |S_v| + |T_e|)\Big)$ 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 reduce the time quite significantly. 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.

4 Adaptive Graphlet Estimation

In previous work, the user must specify the number (or proportion) of samples to use. This is impractical in real-world settings, since the appropriate sample size is intrinsically tied to the required accuracy sufficient for a given problem or application. Thus, we introduce an adaptive optimization scheme for graphlet estimation where the user can specify a bound on the accuracy and the technique automatically finds an approximation that is within the desired accuracy. Thus, since the exact graphlet counts $\mathbf{Y}$ are unknown, we use the error between $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$ as a proxy (See Alg 8). It is straightforward to see that $\mathbb{D}\left(\mathbf{X}_t \parallel \mathbf{X}_{t-1}\right)$ decreases as K increases. Therefore, $\mathbb{D}\left(\mathbf{X}_{t-1} \parallel \mathbf{X}_{t-2}\right) \geq \mathbb{D}\left(\mathbf{X}_t \parallel \mathbf{X}_{t-1}\right)$. Intuitively, as K increases the variance $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$ shrinks toward zero.

Algorithmic Template: Alg 8 learns the appropriate sample size automatically given the desired error. A parallel scheme to solve the graphlet optimization problem is also proposed (see Section 5 for further details) and used in Section 6.

Objective Function: The objective function aims to minimize an arbitrary loss (See Alg 8 Line 21–23). For this, we use the maximum relative error.

$$\min_{\mathbf{X}^{(t)}, \mathbf{X}^\star} \left\{ \max_{G_i \in \mathcal{G}^{(4)}} \frac{|X_i^{(t)} - X_i^\star|}{X_i^\star} \right\} \tag{29}$$

where $X_i^\star$ is the best solution found thus far. The inner part computes the maximum graphlet estimation error using relative error. However, we also investigated KS-statistic, KL/Skew-divergence, and squared-loss.

Adaptive estimation: Given a set J (from the t-th iteration),

TABLE 3: Computational complexity

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

Algorithm 8 Adaptive graphlet estimation.

Input:
a graph $G = (V, E)$
an arbitrary loss $\mathcal{L}(\cdot)$ ▷ for instance, max. relative error
an error bound β such that $0 \leq \beta \leq 1$
max number of iteration $t_{\max}$
Output: $\mathbf{X}$, where X_i is the estimate for the graphlet G_i

1 $\phi = \frac{1}{\left(1/(\delta_{\text{err}} + \epsilon)\right) \cdot \sqrt{m}}$ ▷ Set ϕ if not specified by user
2 Set $J \leftarrow \emptyset$, $t \leftarrow 0$, $\delta_{\text{err}} \leftarrow 1$
3 Initialize $\mathbf{X}$ uniformly at randomly
4 **while** $\delta_{\text{err}} - \epsilon > \beta$ **and** $t < t_{\max}$ **do**
5 $K_t = \lceil \phi \cdot (M - |J|) \rceil$ ▷ Update sample size
6 Set $J_t = \emptyset$
7 **parallel for** $\tau = 1, 2, \ldots, K_t$ **do**
8 $e \sim$ UniformDiscrete $\{1, 2, \ldots, M\}$
9 **if** sampling without replacement **then**
10 **while** $\mathbf{\Psi}(e) > 0$ **do** ▷ edge has been sampled
11 $e \sim$ UniformDiscrete $\{1, 2, \ldots, M\}$
12 **end while**
13 Mark edge e in $\mathbf{\Psi}(e)$
14 Set $J_t \leftarrow J_t \cup \{e\}$
15 Obtain $\mathbf{C}(e)$ for e via Alg 2 Line 2–16
16 Set $C_i^{(t)} \leftarrow C_i^{(t)} + C_i(e)$, for all $i = 1, 2, \ldots, |\mathcal{G}|$
17 **end parallel**
18 Set $C_i \leftarrow C_i + C_i^{(t)}$, for all $i = 1, 2, \ldots, |\mathcal{G}|$ **in parallel**
19 Obtain updated graphlet estimates $\mathbf{X}^{(t)}$ using $\mathbf{C}$ via Eq. 10–24
20 $J \leftarrow J \cup J_t$
21 **parallel for** $G_i \in \mathcal{G}$ **do**
22 Compute loss $\mathrm{w}_i \leftarrow \mathcal{L}\left(X_i^{(t)} \parallel X_i\right)$
23 Update δ_{err} via w_i if required by obj. func.
24 **end parallel**
25 Set $\phi = \phi/2$ ▷ Update sampling probability
26 $X \leftarrow X^{(t)}$ ▷ Update current graphlet estimates
27 $t \leftarrow t + 1$
28 **end while**

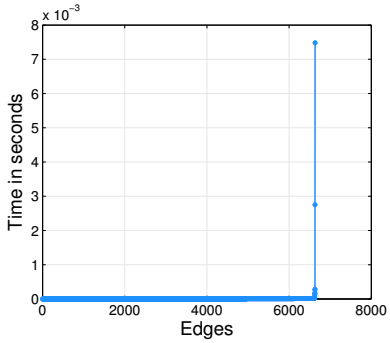


Fig. 2: Power-law relation is observed between the graphlet edge computation time. The time taken to count $k = \{2, 3, 4\}$ graphlets for each edge in `tech-routers-rf` is shown above. See text for discussion.

the goal is to find the minimum set of edge neighborhoods such that $\mathbb{D}\left(\, X \parallel X_t \,\right) \leq \beta$. At each iteration, how many additional elements $\eta_t = |J_t| - |J_{t-1}|$ should be included in J? There are two general approaches. First, the set J can be increased by a percent ϕ of the remaining edges at each iteration, that is, $\eta_t = \lceil \phi \cdot |E| - |J| \rceil$ where $\eta_1 \geq \cdots \geq \eta_{t-1} \geq \eta_t \geq \cdots \geq \eta_{t_{\max}}$. Hence, the number of samples to increase J by is monotonically decreasing with respect to the iteration t. Clearly, as the sample size increases, the estimation variance decreases. As a result of the above fact, the growth of J is inversely related since the larger J becomes, the less variance in estimation. Thus, J should grow at a rate that is inversely proportional to its size. Alternatively, J may increase by a fixed number of samples at each iteration.

Complexity: The adaptive approach minimizes the relative error between the current best solution $\mathbf{X}$ and the previous best $\mathbf{X}_{t-1}$. Alg 8 finds an estimate for each $G_i \in \mathcal{G}$ in:

$$\mathcal{O}(K\Delta(S_{\max} + T_{\max}) + |\mathcal{G}|t^\star) \;=\; \mathcal{O}\big(K\Delta\big(S_{\max} + T_{\max}\big)\big)$$

where $t^\star$ is the number of iterations and K is the number of selected edge neighborhoods.

5 Parallel Algorithm

Estimation methods from the framework are parallelized via independent edge-centric graphlet computations over the selected set of edge-induced neighborhoods $\{\Gamma(e_i), ..., \Gamma(e_K)\}$. The parallelization is described such that it could be used for both shared and distributed memory architectures[5]. 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 graph[6] shared among the set of local workers (processing units). For macro-level graphlet statistics, the communication cost for a single worker is $O(|\mathcal{G}|)$.

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

5. In 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.

6. For 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.

TABLE 4: 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), T=trillion (tera), P=quadrillion (peta).

	graph	Y	X	$\frac{\lvert Y-X\rvert}{Y}$	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-UIllinois	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
4-NODE-1-TRI	ca-citeseer	616.6B	616.7B	0.0003	611.7B	621.8B
	ca-dblp-2012	705.1B	705.1B	$<10^{-4}$	696.2B	714B
	soc-flickr	30T	30T	0.0005	29.7T	30.4T
	soc-friendster	273.8P	274.4P	0.0023	271.2P	277.6P
	soc-gowalla	443.5B	443.7B	0.0004	438.1B	449.3B
	soc-orkut	1.9P	1.9P	0.0012	1.9P	1.9P
	soc-pokec	53.1T	53.1T	0.0003	52.6T	53.7T
	socfb-Berkeley13	119.8B	119.8B	0.0002	119B	120.6B
	socfb-Indiana	274.6B	274.6B	$<10^{-4}$	272.7B	276.5B
	socfb-MIT	14B	14B	0.0002	13.9B	14.1B
	socfb-OR	220.8B	220.8B	$<10^{-4}$	219.2B	222.5B
	socfb-Texas84	397.7B	397.6B	0.0003	394.8B	400.4B
	socfb-UCLA	102.3B	102.3B	0.0002	101.6B	103B
	socfb-UCSB37	44.7B	44.7B	$<10^{-4}$	44.4B	45B
	socfb-UF	418B	418B	$<10^{-4}$	415.2B	420.9B
	socfb-UIllinois	283.3B	283.2B	0.0004	281.3B	285B
	socfb-Wisconsin87	113.6B	113.6B	0.0005	112.9B	114.3B
	web-wikipedia2009	4.1T	4.1T	$<10^{-4}$	4T	4.2T

to workers by "hardness" (Figure 3) 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. Furthermore, recall that a handful of edge neighborhoods require a lot of work, whereas the vast majority require only a small amount of work; as observed in Figure 2. This ensures we avoid common problems present in other approaches such as the curse of the last reducer [38]. However, notice that computing such a partitioning (Figure 3) is computationally intractable and thus we use edge degree (or volume) as an efficient proxy for "hardness".

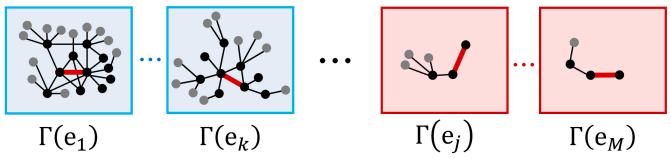
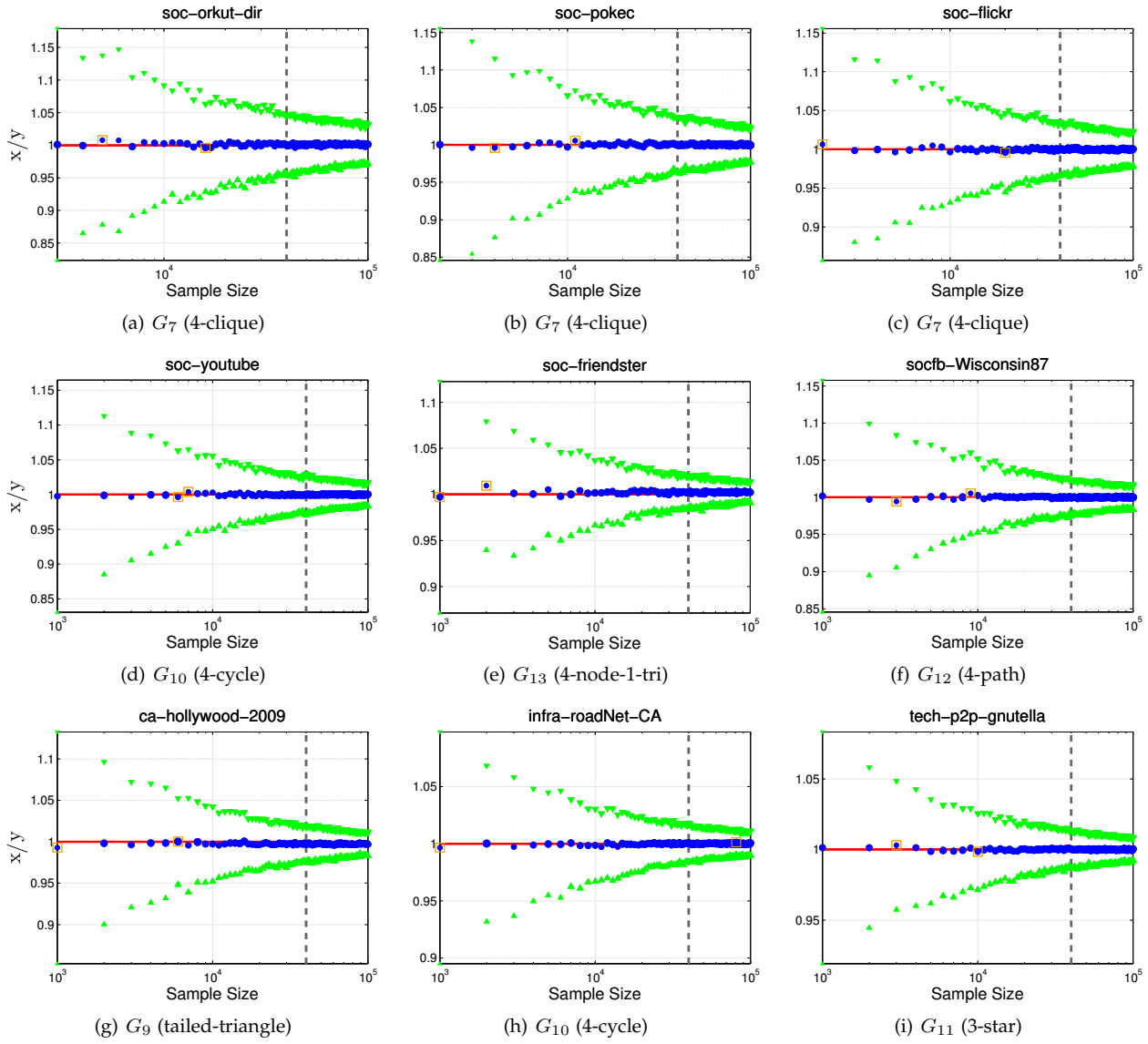


Fig. 3: Edge neighborhoods are ordered and dynamically partitioned to workers by "hardness".

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 significantly better load balancing (compared to vertex-based approaches). It is straightforward to see that if $N < M$, then our approach requires significantly



(a) G_7 (4-clique) (b) G_7 (4-clique) (c) G_7 (4-clique)

(d) G_{10} (4-cycle) (e) G_{13} (4-node-1-tri) (f) G_{12} (4-path)

(g) G_9 (tailed-triangle) (h) G_{10} (4-cycle) (i) G_{11} (3-star)

Fig. 4: Confidence bounds for connected and disconnected graphlets. We used graphs from a variety of domains and types. Note that 4-cliques is understood to be 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 (blue) represents $^X/_Y$ (y-axis) whereas $\triangle$ and $\triangledown$ are $^{\beta_{lb}}/_Y$ and $^{\beta_{ub}}/_Y$, respectively. The 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.

less computations per-edge than per-vertex since

$$X_i = \sum_{e \in E}^{M} X_i(e) = \sum_{v \in V}^{N} X_i(v).$$

Parallelizing via edge-induced neighborhoods provides significantly better load balancing for real-world sparse graphs that follow a power-law. The time taken to count $k = \{2, 3, 4\}$ graphlets for each edge is shown in Fig. 2 and clearly obeys a power-law with only a few edges taking significantly longer than the others. 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 system, etc. Moreover, the edge-centric estimation methods are flexible 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 [39] and shared memory [40], [41] 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.

6 Experiments

In this section, we evaluate the empirical error and performance of the methods with extensive experiments. We use over 300 real-world networks from 20+ domains with different structural characteristics. All data has been made available at NR [42].

TABLE 5: Connected GFD and disconnected GFD estimates for a wide variety of sparse graphs. All estimates have less than 10^{-3} relative error and there is no significant difference between the estimate and actual. Graphlet estimates with relative error less than 10^{-4} are highlighted.

		Connected GFD						Disconnected GFD					KS-Statistic	
Graph	$\|E\|$												**Conn.**	**Disconn.**
ca-AstroPh	196.9K	0.010	0.016	0.193	0.001	0.324	0.455	<0.001	<0.001	<0.001	0.007	0.993	$<10^{-4}$	$<10^{-4}$
ca-MathSciNet	820.6K	0.001	0.003	0.077	<0.001	0.461	0.457	<0.001	<0.001	<0.001	<0.001	0.999	$<10^{-4}$	$<10^{-4}$
ia-email-EU	54.3K	<0.001	0.001	0.031	<0.001	0.715	0.252	<0.001	<0.001	<0.001	<0.001	0.999	0.0005	$<10^{-4}$
ia-enron-large	180.8K	<0.001	0.004	0.060	0.001	0.716	0.219	<0.001	<0.001	<0.001	0.002	0.998	$<10^{-4}$	$<10^{-4}$
rt-retweet-crawl	2.2M	<0.001	<0.001	<0.001	<0.001	0.898	0.101	<0.001	<0.001	<0.001	<0.001	0.999	$<10^{-4}$	$<10^{-4}$
soc-douban	327.1K	<0.001	<0.001	0.012	<0.001	0.436	0.552	<0.001	<0.001	<0.001	<0.001	0.999	0.0005	$<10^{-4}$
soc-youtube-s	2.9M	<0.001	<0.001	0.002	<0.001	0.982	0.016	<0.001	<0.001	<0.001	<0.001	0.999	0.0003	$<10^{-4}$
soc-flickr	3.1M	0.003	0.020	0.132	0.010	0.477	0.358	<0.001	<0.001	<0.001	<0.001	0.999	0.0007	$<10^{-4}$
soc-twitter-higgs	14.8M	<0.001	<0.001	0.003	<0.001	0.972	0.024	<0.001	<0.001	<0.001	<0.001	0.999	$<10^{-4}$	$<10^{-4}$
soc-friendster	1.8T	<0.001	<0.001	0.009	<0.001	0.400	0.590	<0.001	<0.001	<0.001	<0.001	0.999	$<10^{-4}$	$<10^{-4}$
socfb-UIllinois	1.2M	0.001	0.005	0.071	0.002	0.499	0.422	<0.001	<0.001	<0.001	0.016	0.984	0.0001	$<10^{-4}$
socfb-Indiana	1.3M	0.001	0.006	0.089	0.003	0.300	0.600	<0.001	<0.001	<0.001	0.017	0.982	$<10^{-4}$	$<10^{-4}$
socfb-Penn94	1.3M	<0.001	0.002	0.039	0.001	0.652	0.304	<0.001	<0.001	<0.001	0.009	0.991	$<10^{-4}$	$<10^{-4}$
socfb-Texas84	1.5M	<0.001	0.002	0.043	0.001	0.667	0.287	<0.001	<0.001	<0.001	0.014	0.986	0.0007	$<10^{-4}$
tech-internet-as	85.1K	<0.001	<0.001	0.005	<0.001	0.963	<0.001	0.000	<0.001	<0.001	<0.001	0.999	0.0003	$<10^{-4}$

6.1 Estimating Macro Graphlet Statistics

We proceed by first demonstrating the effectiveness of the proposed methods for estimating the frequency of both connected and disconnected 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:

$$\mathbb{D}\left(X_i \parallel Y_i \right) = \frac{|X_i - Y_i|}{Y_i}$$

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 of interest, where 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 both connected and disconnected graphlets are provided in Table 4 for a wide range of graphs from various domains. 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 macro-level global counts such as the median, standard deviation, variance, irq, Q1, Q3, and others. Overall, the methods are found to be accurate for many of the new graphlet statistics as shown in Figure 5. As an aside, for estimating the max 4-cliques, we found that selecting edges via the k-core distribution resulted in high accuracy at very low sample rates.

6.2 Confidence Bounds

Given an arbitrary graphlet $G_i \in \mathcal{G}$, we compute X_i using the estimators from the framework derived in Section 2 and construct confidence bounds for the unknown Y_i. Using the large sampling distribution, we derive lower and upper bounds such that

$$\beta_{\text{lb}} \leq Y_i \leq \beta_{\text{ub}} \tag{30}$$

where

$$\beta_{\text{ub}} = X_i - z_{\alpha/2} \cdot \sqrt{\mathbb{V}[X_i]} \tag{31}$$

and

$$\beta_{\text{lb}} = X_i + z_{\alpha/2} \cdot \sqrt{\mathbb{V}[X_i]} \tag{32}$$

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

$$X_i - 1.96\sqrt{\mathbb{V}[X_i]} \;\leq\; Y_i \;\leq\; X_i + 1.96\sqrt{\mathbb{V}[X_i]} \tag{33}$$

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. 5: 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 (connected graphlet) and 4-node-1-triangle (disconnected graphlet) are shown in Table 4 (other graphlet results were removed due to space). In all cases, the actual graphlet statistics lie inside the error bounds, $\beta_{\text{lb}} \leq Y_i \leq \beta_{\text{ub}}$. Figure 4 investigates the properties of the sampling distribution as the sample size increases. The circle (blue) in Figure 4 represents the fraction X_i/Y_i. Further, β_{lb}/Y_i and β_{ub}/Y_i are represented in Figure 4 by $\triangle$ and $\triangledown$, 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 $\mathbb{P}(\beta_{\text{lb}} \leq Y \leq \beta_{\text{ub}})$ 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.

6.3 Graphlet Frequency Distribution (GFD)

We investigate the methods for approximating three different distributions: connected GFD, disconnected GFD, and the combined GFD consisting of both connected and disconnected graphlets. Strikingly, the estimated GFD from our approach almost perfectly matches the actual GFD (Figure 6). Observe that the methods are evaluated by how well they estimate the entire GFD and thus Figure 6 indicates that the proposed methods estimate all such induced subgraphs from Table 1 with excellent accuracy (matching the actual GFD in all cases). Results for sparse graphs are shown in Table 5 and dense graphs are shown in Table 6. The KS-Statistic for both the connected GFD and disconnected GFD is very small for all graphs.

6.4 Scalability

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 Figure 7, we used a 4-processor Intel Xeon E5-4627 v2 3.3GHz CPU. Overall, the methods show strong scaling (See Figure 7). Similar results were found for other graphs and sample sizes.

6.5 Runtime Comparison

This section investigates the performance of the proposed class of localized graphlet estimation methods.

– *Small and medium sized graphs:* For hundreds of small and medium sized graphs, our method is on average 2895x faster than other existing *exact* approaches [32], [33], [34], [35].

– *Large networks:* For larger networks with hundreds of millions of edges, our method is over $200K$ times. We observe that the speedup (relative to existing methods) increases with the size of the network. Nevertheless, if we exclude PGD, the difference in runtime between other approaches [32], [34], [35] is even larger. In many instances, these methods never finished and/or crashed after exceeding a day (even for relatively small graphs), whereas for these same graphs, our method takes less than a second to obtain accurate estimates $\leq 0.1\%$ for each graphlet $G_i \in \mathcal{G}$. Nevertheless, in all cases our approach is significantly faster, and most importantly, our approach is capable of computing graphlets on massive networks with more than a billion edges.

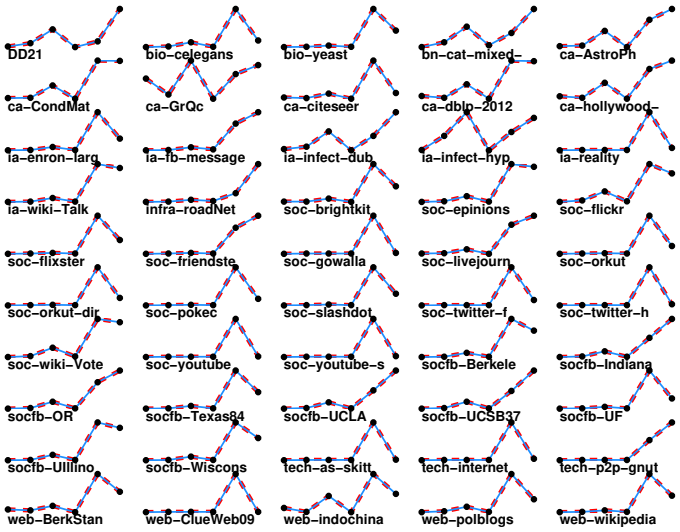


Fig. 6: Estimated GFD is indistinguishable from the actual (larger dotted red line), even across a wide variety of graphs with fundamentally different structural characteristics. The y-axis is the normalized 4-vertex connected graphlet counts $\mathbf{x}' = \mathbf{x} - \min(\mathbf{x}) / \max(\mathbf{x}) - \min(\mathbf{x})$ where $\mathbf{x}$ is the vector of graphlet counts. Nevertheless, similar results were found for other graphlet sizes and GFD variants such as the disconnected GFD and GFD consisting of both connected and disconnected graphlets.

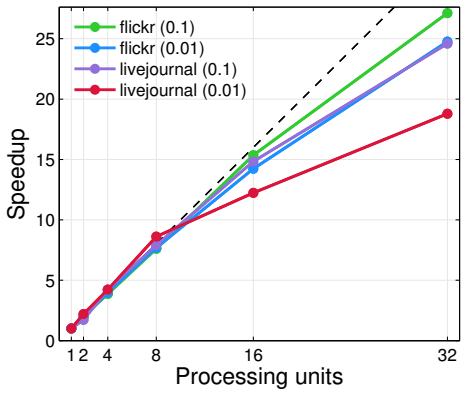


Fig. 7: Strong scaling results for various graphlet estimation problems. See text for discussion.

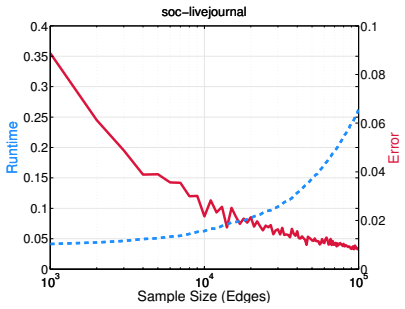


Fig. 8: Effectively balancing speed and accuracy (4-cycles).

6.6 Effectiveness of Adaptive Approach

Given an error bound (which may be specified by the user), the proposed method from Section 4 automatically finds estimates for all graphlets $G_i \in \mathcal{G}$ such that $\mathbb{D}\left(\hat{\mathbf{x}} \parallel \mathbf{y}\right) < \beta$ where β is usually small (e.g., $\beta = 10^{-4}$) but can be adjusted by the user to balance the trade-off between accuracy and time. For instance, many applications require fast methods that operate in real-time (with interactive rates). To achieve such rates, our approach trades off accuracy for time. Results are shown in Table 7 and Figure 8. Overall, the methods are fast, scalable (nearly linear scaling), and accurate with a very small KS and KL-divergence <10^{-4} from the actual. As expected, we find that the relative error between the actual graphlet statistics and the final estimates returned by the method are within the desired error bound (*e.g.*, user-specified).

TABLE 6: Connected and disconnected GFD estimates for graphs that are significantly more dense. All estimates have less than 10^{-3} relative error and there is no significant difference between the estimate and actual. Graphlet estimates with relative error $<10^{-4}$ are highlighted.

Graph	$\|E\|$	Connected GFD							Disconnected GFD				KS-Statistic	
													Conn.	Disconn.
johnson32-2-4	107.8K	0.446	0.428	0.066	0.033	0.023	0.005	<0.001	0.887	0.008	0.032	0.074	$<10^{-4}$	$<10^{-4}$
brock800-3	207.3K	0.089	0.290	0.314	0.079	0.057	0.170	0.285	0.463	0.116	0.125	0.011	$<10^{-4}$	0.0005
brock800-1	207.5K	0.090	0.291	0.314	0.079	0.057	0.170	0.285	0.463	0.116	0.125	0.011	$<10^{-4}$	0.0003
san1000	250.5K	0.120	0.192	0.274	0.037	0.063	0.315	0.367	0.247	0.277	0.093	0.017	$<10^{-4}$	0.0013
p-hat1500-1	284.9K	0.004	0.047	0.218	0.048	0.190	0.494	0.036	0.275	0.058	0.401	0.230	0.0010	0.0003
C2000-5	999.8K	0.026	0.158	0.316	0.079	0.105	0.316	0.154	0.462	0.115	0.231	0.038	$<10^{-4}$	0.0001
C4000-5	4M	0.026	0.158	0.316	0.079	0.105	0.316	0.154	0.462	0.115	0.231	0.038	$<10^{-4}$	$<10^{-4}$

TABLE 7: Adaptive estimation results for a variety of networks. The methods have excellent accuracy (very small KS/KL-div.). In all cases, the maximum relative error is <0.001 and usually much less. This method has been shown to be effective for both large sparse and dense networks that arise in many real-world applications. Recall that ϕ is the fraction of edge neighborhoods used (converged), $t^\star$ is the total number of steps from Alg 8, and $\delta^\star$ is the converged objective. The KS-stat. and KL-div. below is shown for connected graphlets, since it is even smaller for disconnected graphlets.

	$\|E\|$	ϕ	$t^\star$	$\delta^\star$	KS	KL
C4000-5	4M	0.0003	4	0.0003	$<10^{-4}$	$<10^{-4}$
soc-douban	327.1K	$<10^{-6}$	150	0.0007	0.0005	$<10^{-4}$
soc-friendster	1.8B	$<10^{-6}$	50	0.0006	$<10^{-4}$	$<10^{-4}$
soc-gowalla	950.3K	0.0283	287	0.0007	0.0002	$<10^{-4}$
soc-twitter-higgs	14.8M	0.0000	161	0.0007	$<10^{-4}$	$<10^{-4}$
socfb-Indiana	1.3M	0.0080	81	0.0009	$<10^{-4}$	$<10^{-4}$
socfb-Penn94	1.3M	0.0175	177	0.0007	$<10^{-4}$	$<10^{-4}$

6.7 Micro Graphlet Estimation Experiments

This section investigates the accuracy, runtime, and scalability of the computational framework presented in Section 3 for estimating micro graphlet statistics and distributions of individual graph elements such as an edge (or node, path, or subgraph) as opposed to estimating macro-level graphlet statistics over the entire graph G. Results are shown in Table 8. Note that for simplicity, nodes are selected uniformly at random, thus F in Alg 7 represents a uniform distribution over the neighbors.

6.8 Extremal Graphlet Estimation

Problem. (MAX GRAPHLET ESTIMATION) Given a graph G, and a graphlet pattern G_j of size k, find

$$Z_j = \max_{e_i \in \{e_1,\ldots,e_m\}} \Big[X_j(e_i) \Big] \tag{34}$$

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 2 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 9. 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 induced subgraphs.

TABLE 8: Micro graphlet estimation experiments. For each graph problem, we report the relative error averaged over 500 randomly selected edges. These experiments use $p_e = 0.001$ (See Section 3 for more details). In addition to the high accuracy, the micro graphlet estimation methods are between 900-1000K 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

6.9 Comparison to Previous Work

We compare to recent work done on approximating simple counts of a few connected graphlets. Results are provided in Table 10. 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 macro-level counts of connected graphlets, whereas the proposed class of LGE methods accurately estimate a wide variety of macro and micro-level statistics (including simple counts) and

TABLE 9: Results for two of our 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 significantly better estimates for the vast majority of statistics below. In particular, at $p_i = 0.005$, 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 using $p = 0.005$), 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 algorithm).

Method	Speedup	Maximum connected 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

TABLE 10: Results for counting connected 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 $\beta = 0.01$. A hyphen (−) indicates that the method did not terminate within 12 hours. The best time for each problem instance is bolded.

		Time in seconds				
graph	$\|E\|$	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

distributions for both connected and disconnected graphlets. See Table 2 for a summary of the differences. Note that the 3-path sampling heuristic by Jha *et al.* [31] requires significantly more samples to obtain estimates with similar accuracy. In addition, that approach requires two different methods for estimating connected graphlets counts of size 4, and thus requires 2x the samples. In particular, we find that 3-path sampling, GUISE, and GRAFT are unable to obtain accurate estimates within a reasonable amount of time[7]. See Table 10. 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.

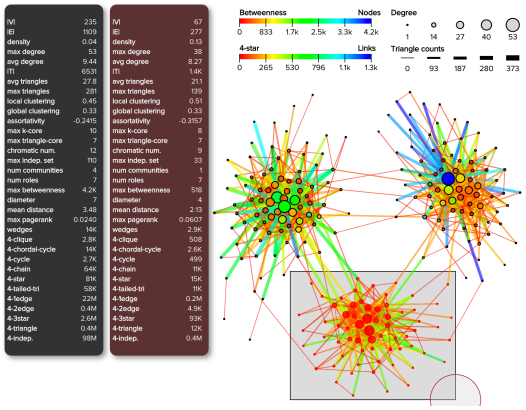


Fig. 9: Application of the fast and accurate approximation methods for real-time interactive graph mining and predictive modeling tasks (*e.g.*, node classification).

7 Applications

Due to the volume and the velocity of big data, approximate results are often a necessity. Graphlet estimators are implemented in a web-based visual graph analytics platform (Figure 9). Graphlet estimation methods (from the proposed estimation framework in Section 2) are implemented in a recent web-based visual graph analytics platform [43] called graphvis (Figure 9). Across all experiments, the graphlet methods are fast and scalable taking <1 ms for 99% of the interactive queries and graphs, while also accurate (no significant 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). Other applications were removed for brevity.

7. Furthermore, GUISE and GRAFT did not converge, even despite using millions of samples, which is consistent with recent findings [31], and especially true for the massive networks used in this work.

8 Conclusion

We have shown that even when dealing with massive networks with more than a billion edges, one can compute graphlets fast and with exceptional accuracy. The newly introduced family of graphlet estimators significantly improves the scalability, flexibility, and utility of graphlets. In addition, this paper studied and proposed estimators for new graphlet problems and statistics including methods for both connected and disconnected graphlets, as well as estimating a number of novel macro and micro-level graphlet statistics. Moreover, we proposed a fast and scalable parallel scheme that generalizes for the family of edge-centric estimation methods in the framework. In addition, an optimization method that automatically finds an estimate within a user-defined level of accuracy without requiring the user to input the sample size. Finally, the methods give rise to new opportunities and applications for graphlets (as shown in Section 7).

References

[1] N. Pržulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?" *Bioinfo.*, vol. 20, no. 18, pp. 3508–3515, 2004.

[2] T. Milenkoviæ and N. Pržulj, "Uncovering biological network function via graphlet degree signatures," *Cancer info.*, vol. 6, 2008.

[3] W. Hayes, K. Sun, and N. Pržulj, "Graphlet-based measures are suitable for biological network comparison," *Bioinformatics*, vol. 29, no. 4, pp. 483–491, 2013.

[4] 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.

[5] 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.

[6] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, vol. 11, pp. 1201–1242, 2010.

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

[8] M. Rupp and G. Schneider, "Graph kernels for molecular similarity," *Molecular Informatics*, vol. 29, no. 4, pp. 266–273, 2010.

[9] 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.

[10] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159*, 2013.

[11] 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.

[12] M. Bādoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," in *STOC*. ACM, 2002, pp. 250–257.

[13] M. Henzinger, S. Krinninger, and D. Nanongkai, "An almost-tight distributed algorithm for computing single-source shortest paths," *arXiv preprint arXiv:1504.07056*, 2015.

[14] J. Pfeffer and K. M. Carley, "k-centralities: local approximations of global measures based on shortest paths," in *WWW*, 2012.

[15] 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.

[16] 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.

[17] Y. Lim and U. Kang, "Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams," in *SIGKDD*, 2015.

[18] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: counting triangles in massive graphs with a coin," in *SIGKDD*, 2009.

[19] R. Pagh and C. E. Tsourakakis, "Colorful triangle counting and a mapreduce implementation," *IPL*, vol. 112, no. 7, pp. 277–281, 2012.

[20] M. Rahman and M. Al Hasan, "Approximate triangle counting algorithms on multi-cores," in *Big Data*, 2013, pp. 127–133.

[21] L. Roditty and U. Zwick, "Dynamic approximate all-pairs shortest paths in undirected graphs," *SICOMP*, vol. 41, pp. 670–683, 2012.

[22] 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.

[23] C. Noble and D. Cook, "Graph-based anomaly detection," in *SIGKDD*, 2003, pp. 631–636.

[24] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *DMKD*, pp. 1–63, 2014.

[25] I. Bhattacharya and L. Getoor, "Entity resolution in graphs," *Mining graph data*, p. 311, 2006.

[26] S. E. Schaeffer, "Graph clustering," *Comp. Sci. Rev.*, vol. 1, no. 1, 2007.

[27] R. Rossi and N. Ahmed, "Role discovery in networks," *TKDE*, vol. 27, no. 4, pp. 1112–1131, 2015.

[28] L. Getoor and B. Taskar, *Introduction to SRL*. MIT press, 2007.

[29] 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.

[30] 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.

[31] M. Jha, C. Seshadhri, and A. Pinar, "Path sampling: A fast and provable method for estimating 4-vertex subgraph counts," in *WWW*, 2015.

[32] D. Marcus and Y. Shavitt, "Rage–a rapid graphlet enumerator for large networks," *Computer Networks*, vol. 56, no. 2, pp. 810–819, 2012.

[33] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *ICDM*, 2015, p. 10.

[34] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, vol. 30, no. 4, pp. 559–565, 2014.

[35] S. Wernicke and F. Rasche, "Fanmod: a tool for fast network motif detection," *Bioinformatics*, vol. 22, no. 9, pp. 1152–1153, 2006.

[36] 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.

[37] 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.

[38] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *WWW*, 2011, pp. 607–614.

[39] P. Ribeiro, F. Silva, and L. Lopes, "Parallel discovery of network motifs," *JPDC*, vol. 72, no. 2, pp. 144–154, 2012.

[40] 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.

[41] 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.

[42] 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

[43] N. K. Ahmed and R. A. Rossi, "Interactive visual graph analytics on the web," in *ICWSM*, 2015.