Estimation of Local Subgraph Counts

Nesreen K. Ahmed
Intel Labs
nesreen.k.ahmed@intel.com

Theodore L. Willke
Intel Labs
ted.willke@intel.com

Ryan A. Rossi
Palo Alto Research Center
rrossi@parc.com

***Abstract*—Graphlets represent small induced subgraphs and are becoming increasingly important for a variety of applications. Despite the importance of the local subgraph (graphlet) counting problem, existing work focuses mainly on counting graphlets globally over the entire graph. These global counts have been used for tasks such as graph classification as well as for understanding and summarizing the fundamental structural patterns in graphs. In contrast, this work proposes an *accurate*, *efficient*, and *scalable parallel* framework for the more challenging problem of counting graphlets locally for a given edge or set of edges. The local graphlet counts provide a topologically rigorous characterization of the local structure surrounding an edge. The aim of this work is to obtain the count of every graphlet of size *k* for each edge. The framework gives rise to efficient, parallel, and accurate unbiased estimation methods with provable error bounds, as well as exact algorithms for counting graphlets locally. Experiments demonstrate the effectiveness of the proposed exact and estimation methods on various datasets. In particular, the exact methods show strong scaling results (11–16x on 16 cores). Moreover, our estimation framework is accurate with error less than 5% on average.**

***Keywords*-Graphlets; edge graphlet counts; statistical estimation; relational learning; link classification; parallel algorithms.**

I. Introduction

As part of the recent surge on large-scale network analysis and its wide applications in social, biological, and information networks, a considerable amount of effort has been devoted to the analysis of the local structure of these networks and their properties (*e.g.*, at the node/edge level). Some popular examples of well-studied local properties include the number of k-vertex small induced subgraphs (*i.e.*, graphlets) in node neighborhoods. For example, number of links, triangles, and wedges surrounding a particular node. Despite the importance of mining the local structure surrounding nodes and edges and its applications in various domains, the analysis has been largely limited to small networks with a few hundred/thousand nodes and edges. Moreover, incorporating higher-order properties (network graphlets/motifs of $k \geq 3$) is also more challenging and computationally intensive even for relatively small graphs.

We define and study a fundamental computational problem at the heart of many network analysis and mining tasks, the *local graphlet decomposition problem*: Given an input graph, we seek to accurately compute the counts of all possible k-vertex induced subgraphs (whether frequent or not) incident to an arbitrary node/edge.

Graphlets are small induced subgraphs that were found to be useful for many predictive and descriptive modeling tasks [1], [2] in a variety of disciplines including bioinformatics [3], cheminformatics [4], and image processing and computer vision [5], [6]. Given a network G, our approach counts the frequency of each $k \in \{3, 4\}$-vertex induced subgraph patterns (See Table II). These counts represent powerful features that succinctly characterize the fundamental network structure [3]. Indeed, it has been shown that such features accurately capture the local network structure in a variety of domains [7], [8], [9]. As opposed to global graph parameters such as diameter for which two or more networks may have global graph parameters that are nearly identical, yet their local structural properties may be significantly different.

While most previous work focused on global macro-level graphlet statistics [3], [10], there are significantly fewer methods for local graphlet statistics [11], even despite its fundamental importance for a variety of machine learning tasks. This is likely due to the fact that counting graphlets locally is even more computational challenging (in terms of both time and space) than the global graphlet counting problem, which has only recently seen algorithms capable of handling large networks with hundreds of millions of vertices [10], [12]. In this paper however, we focus on local micro-level graphlet statistics. Micro-level graphlet statistics $\mathbf{x}_j$ of an individual edge $e_j \in E$ 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 [13] such as relational classification [14], link prediction and weighting tasks (e.g., recommending items, friends, web sites, music, events, etc.) [15], detecting anomalies in graphs (e.g., detecting fraud, or attacks/malicious behavior in computer networks) [16], network alignment in biological networks [1], [2], among many others [17], [18], [19], [20]. More generally, these edge graphlet counts provide a topologically rigorous characterization of the local structure surrounding an edge. See Figure 1 for intuition on the problem solved in this work and potential applications.

We propose an exact, fast, efficient, and parallel framework for computing local graphlet (counts) statistics. Moreover, we combine our proposed framework with a statistical unbiased estimation framework with provable error bounds for computing local graphlet statistics approximately. The proposed methods were shown to be extremely effective across a wide variety of networks with fundamentally different structural properties. In particular, the estimation methods are strikingly accurate and fast with little noticeable difference between the exact and estimated graphlet counts. The paper is organized as follows. Section II provides background and problem definition. Then, Section III derives the computational framework and algorithms. Section IV shows a theoretical analysis, unbiased estimation method, and provable error bounds. Sections VI and VII provide experiments and applications, and Section VIII reviews related work. Finally, Section IX concludes.

II. Local Graphlet Counting

This section formulates the general problem of local (micro[1]) graphlet estimation, then derives a flexible computational framework. Preliminaries are given in Section II-A and the problem formulation is provided in Section II-B.

A. Preliminaries

Let $G = (V, E)$ be an undirected graph where V is the set of vertices and E is the set of edges. The number of vertices is $N = |V|$ and number of edges is $M = |E|$. We assume all vertex and edge sets are *ordered*, i.e., $V = \{v_1, v_2, ..., v_i, ..., v_N\}$ such that v_{i-1} appears before v_i and so forth. Similarly, the ordered edges are denoted $E = \{e_1, e_2, ..., e_i, ..., e_M\}$. Given a vertex $v \in V$, let $\Gamma(v) = \{w | (v, w) \in E\}$ be the set of vertices adjacent to v in G. The degree d_v of $v \in V$ is the size of the neighborhood $|\Gamma(v)|$ of v. We also define $\Delta(G)$ to be the largest degree in G (See Table I for a summary of the key notation).

Given a graph G and an *edge* $e = (v, u) \in E$, the edge-induced subgraph is simply $H = (W, E[W])$ where $W = \Gamma(v) \bigcup \Gamma(u)$ is the set of vertices adjacent to v and u and $E[W]$ is the set of edges between any pair of vertices $r, s \in W$ such that $(r, s) \in E$. A graphlet $G_i = (V_k, E_k)$ is a subgraph consisting of a subset $V_k \subset V$ of the k vertices from $G = (V, E)$ together with all edges whose endpoints are both in this subset $E_k = \{\forall e \in E \,|\, e = (u, v) \wedge u, v \in V_k\}$. Let $\mathcal{G}^{(k)}$ denote the set of all possible k-vertex induced subgraphs and $\mathcal{G} = \mathcal{G}^{(2)} \cup \cdots \cup \mathcal{G}^{(k)}$ is the union of all sets for any $2 \leq k \leq N$. Given the graph $G = (V, E)$ and a set $W = \{w_1, \ldots, w_k\} \subset V$ of k-vertices (*i.e.* $|W| = k$), we define a k-graphlet as any k-vertex induced subgraph $G_i = (W, E[W])$ where $G_i \subset \mathcal{G}^{(k)}$.

It is important to distinguish between the two fundamental classes of graphlets, namely, *connected* and *disconnected* graphlets (see Table II). A k-graphlet $G_i = (V_k, E_k)$ is connected if there exists a path from any vertex to any other vertex in the graphlet G_i, $\forall u, v \in V_k, \exists P_{u-v} : u, \ldots, w, \ldots, v$, where $d(u, v)$ is the distance (number of hops) between u and v, and $d(u, v) \geq 0 \wedge d(u, v) \neq \infty$. By definition, a connected graphlet G_i has only one connected component (*i.e.*, $|C| = 1$). A k-graphlet $G_i = (V_k, E_k)$ is disconnected if there is no existing path from any vertex $v \in G_i$ to any other vertex $u \in G_i$. The goal of this work is to compute *local edge-centric* induced subgraph statistics for both *connected* and *disconnected graphlets* of size $k \in \{3, 4\}$. As an aside, the terms graphlet, motif, induced subgraph, and orbit have been used interchangeably in the literature.

[1]The terms *local* and *micro* are used interchangeably and refer to the problem of computing graphlet statistics for individual graph elements such as an edge or even a node.

Table I
Summary of notation. All sets are ordered. Whenever possible, we use standard terminology in the literature.

N	Number of nodes in the graph G
M	Number of edges in the graph G
J	Set of selected edges via some mechanism
K	Number of selected edges
ϵ	Error rate.
δ	Confidence Level.
n	Number of samples.
d_v	Degree of vertex v where $d_v = \|\Gamma(v)\|$
Δ	Maximum degree of the graph G
T_e	Set of vertices that form a triangle with edge $e \in E$.
S_u	Set of vertices forming a 2-star (centered at vertex u) with edge $(v, u) \in E$.
$\Gamma(e_j)$	Edge neighborhood of $e_j = (v, u)$, also denoted as $\Gamma(v, u)$ for convenience. Let $\Gamma_h(e_j)$ be the set of neighbors within h-hops of the edge e_j.
$\mathbf{\Psi}(\cdot)$	Fast lookup table for checking edge existence in constant time (*i.e.*, $o(1)$)
G_i	The i^{th} induced subgraph, see Table II.
$\mathbf{X}$	Let $\mathbf{X} \in \mathbb{R}^{\kappa \times M}$ be a matrix representing the local graphlet counts for all edges $e_j \in E$.
x_{ji}	Count of graphlet G_i for edge $e_j \in E$
$\mathbf{x}_j$	A vector of local graphlet counts for edge e_j. We also denote the local graphlet counts for e_j as $\mathrm{X}_{j:} \in \mathbb{R}^{\kappa}$.

B. Problem Definition

Now, we formally define the local edge-centric graphlet counting problems: Given a graph $G = (V, E)$, an edge $e_j = (v, u) \in E$, find the number of induced subgraphs (*i.e.*, graphlets) that are incident to e_j and isomorphic to the graphlet pattern $G_i \in \mathcal{G}$ – *i.e.*, $|G_i(e_j)|$ (see Table II for all graphlet patterns of size $k = \{3, 4\}$ nodes). For example, $|G_1(e_j)|$ represents the number of triangles incident to edge $e_j \in E$. Note that counting k-vertex graphlets incident to any edge e_j can be performed *naively* in $\mathcal{O}(\Delta^{k-1})$ [3].

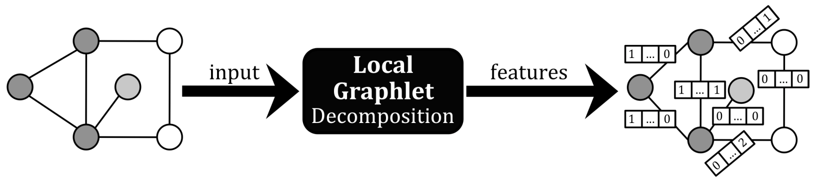
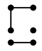


Figure 1. Local graphlet decomposition. Given a graph as input, the local graphlet decomposition methods are useful for computing graph features which in turn can be used for node and edge classification, relational representation discovery, role discovery, among other statistical relational learning (SRL) tasks. Node color above represents class labels. Notice that the *edge features* derived by the proposed methods can easily be transformed into *node features* or used directly by SRL algorithms [14].

Table II
SUMMARY OF THE GRAPHLETS OF SIZE $k \in \{3, 4\}$.

CONNECTED			DISCONNECTED		
	G_1	triangle		G_4	3-node-indep.
	G_2	2-star		G_3	3-node-1-edge
	G_5	4-clique		G_{15}	4-node-indep.
	G_6	chordal-cycle†		G_{14}	4-node-1-edge
	G_7	tailed-triangle‡		G_{13}	4-node-2-star
	G_8	4-cycle		G_{12}	4-node-2-edge
	G_9	3-star		G_{11}	4-node-1-triangle
	G_{10}	4-path			

Problem. (LOCAL EDGE-CENTRIC GRAPHLET ESTIMATION) Given a graph $G = (V, E)$ and an edge $e_j = (v, u) \in E$, the *local edge-centric graphlet estimation problem* is to find

$$\widehat{\mathbf{x}}_j = \begin{bmatrix} \widehat{x}_1 & \cdots & \widehat{x}_4 & \widehat{x}_5 & \cdots & \widehat{x}_{10} & \cdots & \widehat{x}_{15} \end{bmatrix}$$

where $\widehat{\mathbf{x}}_j$ is an approximation of the exact local graphlet statistics denoted $\mathbf{x}_j$ for edge e_j such that $\widehat{\mathbf{x}}_j \approx \mathbf{x}_j$ and thus the distance $\mathbb{D}\left(\, \widehat{\mathbf{x}}_j \parallel \mathbf{x}_j \,\right)$ is minimized as well as the computational cost associated with the estimation. Moreover, $\widehat{\mathbf{x}}_j$ is a provably unbiased estimate of $\mathbf{x}_j$ with precise error bounds (as we show in the next sections). Note that $\mathbb{D}\left(\, \widehat{\mathbf{x}}_j \parallel \mathbf{x}_j \,\right)$ can be any loss/distance function. The aim of the *local edge-centric graphlet estimation problem* is to compute a fast approximation of the graphlet statistics centered at (or incident to) an individual edge[2]. See Figure 1 for further intuition on the problem and potential use cases.

III. LOCAL GRAPHLET FRAMEWORK

This section derives a flexible computational framework for the local graphlet counting problem, and serves as a basis for the proposed exact and estimation methods. In particular, Section III-A discusses the initial preprocessing steps that significantly improve the efficiency of our method. Section III-B introduces the exact approach whereas the unbiased estimation method for local edge graphlet counts is proposed in Section IV.

[2]Graphlets are estimated locally (at the micro-level), that is, per edge as opposed to globally.

Algorithm 1 Family of Local Edge-centric Graphlet Decomposition Algorithms

```
Input:
    Graph $G = (V, E)$
    Ordered set of edges $J = \{e_1, \cdots, e_K\} \subseteq E$
    Error rate $\epsilon$
    Confidence level $\delta$
1: Compute preprocessing steps from Section III-A
2: parallel for each $e_j \in J$ in order do
3:     Obtain exact local graphlet counts $\mathbf{x}_j$ for $e_j$ via Alg. 2 or
       use Alg. 3 (with parameters $\epsilon$ and $\delta$) to obtain a fast and
       accurate unbiased estimation of the local graphlet counts
       $\widehat{\mathbf{x}}_j$ for edge $e_j$
4:     Set $\mathbf{X}_{j:}$ to be $\mathbf{x}_j^\top$
5: end parallel
6: return $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_j & \cdots & \mathbf{x}_K \end{bmatrix}^\top$ consisting of the
   local graphlet counts $\mathbf{x}_j$ for each edge $e_j \in J$
```

A. Preprocessing Steps

Our approach benefits from the preprocessing steps below and the useful computational properties that arise.

P1 The vertices $V = \{v_1, \ldots, v_N\}$ are sorted from smallest to largest degree and relabeled such that $d(v_1) \leq d(v_2) \leq d(v_i) \leq d(v_N)$.

P2 For each $\Gamma(v_i) \in \{\Gamma(v_1), \ldots, \Gamma(v_N)\}$, the vertex neighbors in $\Gamma(v_i) = \{\ldots, w_j, \ldots, w_k, \ldots\}$ are ordered by an arbitrary graph property $f(\cdot)$ s.t. $j < k$ if $f(w_j) \geq f(w_k)$. Thus, the set of neighbors $\Gamma(v_i)$ are ordered from largest to smallest degree and ties are broken by vertex id.

P3 Given an edge $(v, u) \in E$, we ensure that $d_v \geq d_u$ (*i.e.*, v is always the vertex with larger or equal degree).

P4 Let π be an ordering of the set of edges such that $k < j$ for e_k and e_j if $f(e_k) > f(e_j)$, and ties are broken arbitrarily.

Clearly, the order that the preprocessing steps are performed is important. The above preprocessing steps ($\mathbf{P1} - \mathbf{P4}$) give rise to many useful properties and leads to significant reduction in runtime. For finding the 4-cycles centered at an

Algorithm 2 A generalized framework is described below for the local graphlet problem. Given a graph $G = (V, E)$, the algorithm returns the graphlet feature vector $\mathbf{x}_j = \mathbf{x}$ for edge $e_j \in E$.

```
1:  procedure LocalGraphlet($G$, $e_j = (v, u)$)
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 $\mathbf{\Psi}(w) = \lambda_1$
5:      parallel for each $w \in \Gamma(u)$ and $w \neq v$ do
6:          if $\mathbf{\Psi}(w) = \lambda_1$ then
7:              $T_e \leftarrow T_e \cup \{w\}$ and set $\mathbf{\Psi}(w) = \lambda_3$        ▷ triangle
8:              $S_v \leftarrow S_v \setminus \{w\}$
9:          else $S_u \leftarrow S_u \cup \{w\}$ and set $\mathbf{\Psi}(w) = \lambda_2$        ▷ wedge
10:     parallel for each $w \in T_e$ do
11:         for $r \in \Gamma(w)$ do
12:             if $\mathbf{\Psi}(r) = \lambda_3$ then Set $x_5 \leftarrow x_5 + 1$      ▷ local 4-clique
13:         Set $\mathbf{\Psi}(w)$ to $\lambda_4$
14:         Set $\sigma' \leftarrow \sigma' + |\Gamma(w)| - 2$
15:     parallel for each $w \in S_u$ do
16:         for $r \in \Gamma(w)$ do
17:             if $\mathbf{\Psi}(r) = \lambda_1$ then set $x_8 \leftarrow x_8 + 1$      ▷ local 4-cycle
18:             if $\mathbf{\Psi}(r) = \lambda_2$ then set $x_7 \leftarrow x_7 + 1$      ▷ local tailed-tri
19:             if $\mathbf{\Psi}(r) = \lambda_4$ then set $\sigma \leftarrow \sigma + 1$
20:         Set $\mathbf{\Psi}(w)$ to 0
21:         Set $\sigma' \leftarrow \sigma' + |\Gamma(w)| - 1$
22:     parallel for each $w \in S_v$ do
23:         for $r \in \Gamma(w)$ do
24:             if $\mathbf{\Psi}(r) = \lambda_1$ then set $x_7 \leftarrow x_7 + 1$      ▷ local tailed-tri
25:             if $\mathbf{\Psi}(r) = \lambda_4$ then set $\sigma \leftarrow \sigma + 1$
26:         Set $\mathbf{\Psi}(w)$ to 0
27:         Set $\sigma' \leftarrow \sigma' + |\Gamma(w)| - 1$
28:     Derive local 3-graphlets for $e_j$ via Eq.1-4
29:     Use Eq.7–9 to derive the local connected 4-graphlets for $e_j$ and
        disconnected 4-graphlets via Eq.10–14.
30:     return $\mathbf{x}_j = \mathbf{x}$, where $x_i$ is the count of graphlet $G_i$ for $e_j$
```

edge $e_j = (v, u) \in E$, we avoid searching S_v completely, and instead search only S_u, which due to $\mathbf{P3}$ is likely to be much less expensive than searching S_v.

B. Exact Algorithm

Given a set of edges $J \subseteq E$ where $K = |J|$, Alg. 1 computes $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_k & \cdots & \mathbf{x}_K\end{bmatrix}^\top$ consisting of the local graphlet counts $\mathbf{x}_j$ for each edge $e_j \in J$. The exact method for deriving local graphlet counts $\mathbf{x}_j$ centered at an individual edge e_j is given in Alg. 2.

Local 3-graphlets: The proposed approach derives all k-graphlets for $k \in \{3, 4\}$ using only the local edge-based counts of triangles, cliques, and cycles, along with a few other constant time graph and vertex parameters such as number of vertices $N = |V|$, edges $M = |E|$, as well as vertex degree $d_v = |\Gamma(v)|$. Given an edge $e_j = (v, u) \in E$ from G, let x_1, x_5, and x_8 be the frequency of triangles, cliques, cycles, and tailed-triangles centered at (or incident to) the edge $e_j \in E$ in the graph G, respectively. Note x_i (or x_{ji}, X_{ji}) is the count of the induced subgraph G_i for an arbitrary edge e_j (See Table II for graphlets definition). Note that this framework is based on the state-of-the-art methods in [10], [12]. The local (micro-level) 3-graphlets for edge e_j are as follows:

$$x_1 = |T_e| \tag{1}$$
$$x_2 = (d_u + d_v - 2) - 2|T_e| \tag{2}$$
$$x_3 = N - x_2 - |T_e| - 2 \tag{3}$$
$$x_4 = \binom{N}{3} - (x_1 + x_2 + x_3) \tag{4}$$

Further, notice that given $x_1 = |T_e|$ for $e_j = (v, u) \in E$, we can derive $|S_u|$ and $|S_v|$ (that is, the number of 2-star patterns centered at u and v of e_j, respectively) as:

$$|S_u| = d_u - |T_e| - 1 \tag{5}$$
$$|S_v| = d_v - |T_e| - 1 \tag{6}$$

Therefore, the number of two-stars centered at e_j denoted x_4 can be rewritten simply as $x_2 = |S_u| + |S_v|$. These 3-vertex induced subgraph statistics are then used as a basis to derive the induced subgraphs of size $k + 1$ (*i.e.*, 4-vertex graphlets).

Local Connected 4-graphlets: Recall that S_u and S_v are the number of 2-star patterns centered at u and v for edge e_j and can easily be derived in $o(1)$ time using only d_u, d_v, and the triangle count x_1. Given the frequency of 3-cliques (triangles) x_1 and 4-cliques x_5 centered at e_j, the local chordal cycles x_6 centered at e_j are as follows (note that e_j in this case is the chord edge):

$$x_6 = \binom{|T_e|}{2} - x_5 \tag{7}$$

Similarly, given the local 4-cycle count x_8, we derive the local 4-path count x_{10} for edge e_j as follows:

$$x_{10} = \big(|S_v| \cdot |S_u|\big) - x_8 \tag{8}$$

Finally, given the local tailed-triangle count x_7, we derive the local 3-star count x_9 for edge e_j as follows (note that e_j in this case is the tail edge):

$$x_9 = \binom{|S_v|}{2} + \binom{|S_u|}{2} - x_7 \tag{9}$$

Local Disconnected 4-graphlets: Disconnected graphlets are derived as follows:

$$x_{11} = |T_e| \cdot \big[N - (|T_e| + |S_u| + |S_v| + 2)\big] \tag{10}$$
$$x_{12} = M - (d_u + d_v - 1) - (\sigma' - \sigma - x_5 - x_8 - x_7) \tag{11}$$
$$x_{13} = (|S_u| + |S_v|) \cdot \big[N - (|T_e| + |S_u| + |S_v| + 2)\big] \tag{12}$$
$$x_{14} = \binom{N - \big[|T_e| + |S_u| + |S_v| + 2\big]}{2} - x_{12} \tag{13}$$
$$x_{15} = \binom{N}{4} - \sum_{i=5}^{14} x_i \tag{14}$$

Algorithm Discussion: Alg. 2 counts only a few graphlets and achieves all the remaining counts in constant time by using equations 1–14. Alg. 2 starts by finding the set of triangles T_e and 2-stars S_u, S_v incident to an input edge $e_j = (u, v)$ (see Lines 3–9). Note that in Alg. 2, we use a categorization scheme to divide the search space (similar to [10], [12]). More specifically, we use a hash table $\mathbf{\Psi}(w)$ to identify the connectivity pattern of any node w in the neighborhood of e_j (*i.e.*, $w \in \Gamma(e_j)$). We set $\mathbf{\Psi}(w)$ to one of the variables $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$. The variables $\{\lambda_1, \lambda_2, \lambda_3\}$ are used to distinguish between star nodes incident to v, star nodes incident to u, and triangle nodes incident to both u and v respectively. Note that the variables $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ are unique variables (*i.e.*, $\lambda_1 \neq \lambda_2 \neq \lambda_3 \neq \lambda_4$). In Lines 10–14, Alg. 2 finds the 4-cliques, and set $\mathbf{\Psi}(w) = \lambda_4$ for all nodes $w \in T_e$ (to avoid. Then, Lines 15–21 searches for 4-cycles and tailed-triangles (centered around vertex u). Finally, Lines 22–27 continues the search for tailed-triangles (centered around vertex v).

C. Sampling Algorithm

A generalized and flexible framework for the *local graphlet estimation problem* is given in Alg 3. In particular, Alg. 3 takes as input an edge e_j, a graph G, an error rate ϵ, a confidence level δ, and it returns the unbiased estimates for the graphlet feature vector $\widehat{\mathbf{x}}_j$ for edge $e_j \in E$. Given $w \in T_e$ (or S_u, S_v), we propose selecting $r \in \Gamma(w)$ (a neighbor of w) with probability π_r according to an arbitrary weighted/uniform distribution F. First, we compute T_e, S_u, and S_v in Lines 3-10. Afterwards, Eq. 1–4 compute all graphlets of size $k = 3$ exactly. Next, we compute 4-cliques in Lines 11-16. Line 11 searches each vertex $w \in T_e$ in parallel. Given $w \in T_e$, we draw a vertex r randomly with replacement from the neighbors of w, where π_r is the inclusion/sampling probability according to some distribution function F. Then, we check if r is of type λ_3 (from Line 8), as this indicates that r also participates in a triangle with $e_j = (v, u)$, and since $r \in \Gamma(w)$, then $\{v, u, w, r\}$ is a 4-clique. Finally, we use the estimators and error bounds in Section IV to obtain the unbiased estimate of 4-clique counts for edge e_j. In addition, Line 16 ensures that the same 4-clique is not counted twice. Further, 4-cycles are computed in Lines 18-25 as well as the tailed-triangles centered around u. The remaining tailed-triangles are computed in Lines 27-33. As an aside, σ, σ' are also computed and used for estimating x_{12} for graphlet G_{12} (Eq. 11). Finally, the remaining graphlets $\{x_6, \ldots, x_{15}\}$ are estimated in $o(1)$ time (Eq. 7–14) using knowledge from the previous steps. Notably, Alg. 3 gives rise to an efficient exact method, *e.g.*, if $\pi_r = 1$ and selection is performed without replacement (in this case Alg. 3 is equivalent to Alg. 2).

Algorithm 3 Unbiased Local Graphlet Estimation Framework. Given a graph G, error rate ϵ, and confidence level δ, the algorithm returns the estimated graphlet feature vector $\widehat{\mathbf{x}}_j$ for $e_j \in E$.

```
1  procedure LocalGraphletEstimation($G$, $e_j = (v, u)$, $\epsilon$, $\delta$)
2      Initialize variables
3      parallel for each $w \in \Gamma(v)$ do
4          if $w \neq u$ then
5              $S_v \leftarrow S_v \cup \{w\}$ and $\mathbf{\Psi}(w) = \lambda_1$
6      parallel for each $w \in \Gamma(u)$ and $w \neq v$ do
7          if $\mathbf{\Psi}(w) = \lambda_1$ then
8              $T_e \leftarrow T_e \cup \{w\}$ and set $\mathbf{\Psi}(w) = \lambda_3$        ▷ triangle
9              $S_v \leftarrow S_v \setminus \{w\}$
10         else $S_u \leftarrow S_u \cup \{w\}$ and set $\mathbf{\Psi}(w) = \lambda_2$        ▷ wedge
11     parallel for each $w \in T_e$ do
12         Set $d'_w = \lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$
13         for $i = 1, ..., d'_w$ do
14             Select a vertex $r \in \Gamma(w)$ via an arbitrary distribution $\mathsf{F}$
15             if $\mathbf{\Psi}(r) = \lambda_3$ then Set $x_5 \leftarrow x_5 + \left({}^{d_w}/{}_{d'_w}\right)$   ▷ 4-clique
16         Set $\mathbf{\Psi}(w)$ to $\lambda_4$
17         Set $\sigma' \leftarrow \sigma' + |\Gamma(w)| - 2$
18     parallel for each $w \in S_u$ do
19         Set $d'_w = \lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$
20         for $i = 1, ..., d'_w$ do
21             Select a vertex $r \in \Gamma(w)$ via an arbitrary distribution $\mathsf{F}$
22             if $\mathbf{\Psi}(r) = \lambda_1$ then set $x_8 \leftarrow x_8 + \left({}^{d_w}/{}_{d'_w}\right)$     ▷ 4-cycle
23             if $\mathbf{\Psi}(r) = \lambda_2$ then set $x_7 \leftarrow x_7 + \left({}^{d_w}/{}_{d'_w}\right)$    ▷ tailed-tri
24             if $\mathbf{\Psi}(r) = \lambda_4$ then set $\sigma \leftarrow \sigma + \left({}^{d_w}/{}_{d'_w}\right)$
25         Set $\mathbf{\Psi}(w)$ to 0
26         Set $\sigma' \leftarrow \sigma' + |\Gamma(w)| - 1$
27     parallel for each $w \in S_v$ do
28         Set $d'_w = \lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$
29         for $i = 1, ..., d'_w$ do
30             Select a vertex $r \in \Gamma(w)$ via an arbitrary distribution $\mathsf{F}$
31             if $\mathbf{\Psi}(r) = \lambda_1$ then set $x_7 \leftarrow x_7 + \left({}^{d_w}/{}_{d'_w}\right)$     ▷ tailed-tri
32             if $\mathbf{\Psi}(r) = \lambda_4$ then set $\sigma \leftarrow \sigma + \left({}^{d_w}/{}_{d'_w}\right)$
33         Set $\mathbf{\Psi}(w)$ to 0
34         Set $\sigma' \leftarrow \sigma' + |\Gamma(w)| - 1$
35     Derive local 3-graphlets for $e_j$ via Eq.1-4
36     Use Eq.7–9 to derive the local connected 4-graphlets for $e_j$ and
       disconnected 4-graphlets via Eq.10–14.
37     return $\widehat{\mathbf{x}}_j = \widehat{\mathbf{x}}$, where $\widehat{x}_i$ is the estimate of graphlet $G_i$ for $e_j$
```

IV. Unbiased Estimation and Error Bounds

Assume we are given an input edge $e_j = (v, u)$ and an arbitrary neighbor vertex w where $w \in \Gamma(e_j)$ with degree $d_w = |\Gamma(w)|$ and $\mathbf{\Psi}(w) = \lambda$. For each vertex r in $\Gamma(w)$, we check if $\mathbf{\Psi}(r) = \lambda'$. Note that λ and λ' are two identifiers used to define the connectivity patterns of w and r with respect to e_j. Hence, $\lambda \equiv \lambda'$ when the two vertices w and r have the same connectivity pattern with respect to e_j. For example, if $\mathbf{\Psi}(w) = \lambda_3$ and $\mathbf{\Psi}(r) = \lambda_3$ then both w and r form triangles with e_j (*i.e.*, $w, r \in T_e(e_j)$).

Let $H = (\{v, u, w, r\}, E[\{v, u, w, r\}])$ be the 4-vertex induced subgraph incident to e_j and isomorphic to some graphlet $G_i \in \mathcal{G}^{(4)}$ where $\mathbf{\Psi}(w) = \lambda$ and $\mathbf{\Psi}(r) = \lambda'$. Observe that there are x_{ji} 4-vertex induced subgraphs incident to e_j and isomorphic to some graphlet $G_i \in \mathcal{G}^{(4)}$, where x_{ji} defined as follows,

$$x_{ji} = \sum_{\substack{\forall w \in \Gamma(e_j) \\ \mathbb{I}[\mathbf{\Psi}(w)=\lambda]=1}} \sum_{\forall r \in \Gamma(w)} \mathbb{I}[\mathbf{\Psi}(r) = \lambda']$$

Observe that $\mathbb{I}[\mathbf{\Psi}(w) = \lambda]$ and $\mathbb{I}[\mathbf{\Psi}(r) = \lambda']$ are indicator functions that take the value 1 if $\mathbf{\Psi}(w) = \lambda$ and $\mathbf{\Psi}(r) = \lambda'$ respectively, and 0 otherwise.

Suppose we draw a random sample $q = \{q_1, ..., q_n\}$ of size $n < |\Gamma(w)|$ with replacement from the set of neighbors of w (*i.e.*, $\Gamma(w)$). Because we are sampling with replacement, the sample may contain the same vertex more than once. Let π_r be the sampling probability of vertex $r \in \Gamma(w)$, such that $\pi_r = 1/|\Gamma(w)|$ if r is sampled uniformly. Let $Y = \{Y_1, ..., Y_n\}$ be a set of random variables such that $Y_s = 1$ if $\mathbf{\Psi}(q_s) = \lambda'$ and $Y_s = 0$ otherwise. Observe that $H = (\{v, u, w, q_s\}, E[\{v, u, w, q_s\}])$ is a *sampled* 4-vertex induced subgraph incident to e_j and isomorphic to some graphlet $G_i \in \mathcal{G}^{(4)}$. We can give a reasonably good estimate of x_{ji} by computing the count of all 4-vertex induced subgraphs that are observed in the sample (*i.e.*, $Y_{tot} = \sum_{s=1}^{n} Y_s$), and also isomorphic to the graphlet $G_i \in \mathcal{G}^{(4)}$ (*i.e.*, H). Then, we scale the observed count by $|\Gamma(w)|/n$ (using Horvitz-Thompson estimation to fix the sampling bias [21], [22]). In other words, if $\widehat{x}_{ji}$ is our estimate of the count of graphlet G_i incident e_j, then

$$\widehat{x}_{ji} = \sum_{\substack{\forall w \in \Gamma(e_j) \\ \mathbb{I}[\mathbf{\Psi}(w)=\lambda]=1}} \sum_{s=1}^{n} Y_s \cdot \frac{|\Gamma(w)|}{n}$$

Note that we calculate this estimate given only the values of Y for each sampled vertex in q. Next, we prove in Lemma 1 that the local estimated count for any graphlet incident to any edge e_j is unbiased.

Lemma 1. *Given a graph G and any graphlet pattern $G_i \in \mathcal{G}^{(k)}$, for any edge $e_j = (v, u)$ and using Algorithm 3, $\widehat{x}_{ji}$ is an unbiased estimator of x_{ji} – where x_{ji} is the count of all occurrences of G_i incident to e_j.*

Proof: For each $w \in \Gamma(e_j)$, assume we select a random sample $q = \{q_1, ..., q_n\}$ with replacement from the set of neighbors $\Gamma(w)$, such that $\pi_r = 1/|\Gamma(w)|$ is the sampling probability for any vertex $r \in \Gamma(w)$ and $n < |\Gamma(w)|$ is the sample size. Let z_r be a Binomial random variable that indicate the number of times that vertex $r \in \Gamma(w)$ appeared in the sample q. Note that $z_r = 0$ if vertex r did not appear in the sample. We can rewrite $\widehat{x}_{ji}$ as follows,

$$\widehat{x}_{ji} = \sum_{\substack{\forall w \in \Gamma(e_j) \\ \mathbb{I}[\mathbf{\Psi}(w)=\lambda]=1}} \sum_{\forall r \in \Gamma(w)} z_r \cdot \mathbb{I}[\mathbf{\Psi}(r) = \lambda'] \cdot \frac{1}{n \cdot \pi_r}$$

Given that $\sum_{r \in \Gamma(w)} z_r = n$ and $\mathbb{E}(z_r) = n \cdot \pi_r$ for any $r \in \Gamma(w)$, we have the expected value of $\widehat{x}_{ji}$ as follows,

$$\begin{aligned}
\mathbb{E}(\widehat{x}_{ji}) &= \sum_{\substack{\forall w \in \Gamma(e_j) \\ \mathbb{I}[\mathbf{\Psi}(w)=\lambda]=1}} \sum_{\forall r \in \Gamma(w)} \mathbb{E}(z_r) \cdot \mathbb{I}[\mathbf{\Psi}(r) = \lambda'] \cdot \frac{1}{n \cdot \pi_r} \\
&= \sum_{\substack{\forall w \in \Gamma(e_j) \\ \mathbb{I}[\mathbf{\Psi}(w)=\lambda]=1}} \sum_{\forall r \in \Gamma(w)} n \cdot \pi_r \cdot \mathbb{I}[\mathbf{\Psi}(r) = \lambda'] \cdot \frac{1}{n \cdot \pi_r} \\
&= \sum_{\substack{\forall w \in \Gamma(e_j) \\ \mathbb{I}[\mathbf{\Psi}(w)=\lambda]=1}} \sum_{\forall r \in \Gamma(w)} \mathbb{I}[\mathbf{\Psi}(r) = \lambda']
\end{aligned}$$

Thus, $\widehat{x}_{ji}$ is an unbiased estimator for x_{ji} because,

$$\mathbb{E}(\widehat{x}_{ji}) = x_{ji}$$

■

Next, we show that the local estimated count has provable error bounds by using Hoeffding's inequality [23].

Theorem 1. *[Hoeffding's Inequality [23]] Let $Y = \{Y_1, Y_2, ..., Y_n\}$ be n independent 0-1 random variables, not necessarily identically distributed. Then for $Y_{tot} = \sum_{s=1}^{n} Y_s$, $\mu = \mathbb{E}(Y_{tot})$, and $b > 0$,*

$$\mathbf{Pr}[|Y_{tot} - \mu| \geq b] \leq e^{-2b^2/n}$$

Lemma 2. *With probability $1 - \delta$, the estimated local graphlet count is bounded as follows,*

$$x_{ji}(w) - \epsilon|\Gamma(w)| \leq \widehat{x}_{ji}(w) \leq x_{ji}(w) + \epsilon|\Gamma(w)| \quad (15)$$

for any given edge $e_j \in E$, vertex $w \in \Gamma(e_j)$, a random sample with replacement $q = \{q_1, ..., q_n\}$ selected uniformly from the neighbors of w, and any graphlet pattern G_i.

Proof: Let Y_s be a random variable for vertex $q_s \in q$, if $\mathbf{\Psi}(q_s) = \lambda'$ then $Y_s = 1$ and $Y_s = 0$ otherwise. Note that the sampling probability that any vertex $r \in \Gamma(w)$ is $\pi_r = 1/|\Gamma(w)|$. Also, let $x_{ji}(w)$ is the count of all occurrences of G_i incident to e_j and w. Then, the expected value of Y_s is

$$\begin{aligned}
\mathbb{E}(Y_s) &= \sum_{\forall r \in \Gamma(w)} \pi_r \cdot \mathbb{I}[\mathbf{\Psi}(r) = \lambda'] \\
&= \pi_r \cdot x_{ji}(w)
\end{aligned}$$

Thus, if $Y_{tot} = \sum_{s=1}^{n} Y_s$, we have $\mu = \mathbb{E}(Y_{tot}) = n \cdot \pi_r \cdot x_{ji}(w)$. We have $\widehat{x}_{ji}(w) = \frac{|\Gamma(w)|}{n} \cdot Y_{tot}$ from Lemma 1. If we now apply the Hoeffding's inequality with $b = \epsilon n$, we have that

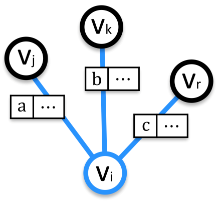


Figure 2. Edge to node graphlet frequency. Given graphlet counts a, b, and c for graphlet G, we compute the graphlet count x_i for node v_i as $(a+b+c)/\alpha$ where α is the normalization constant for graphlet G.

$$\mathbf{Pr}[|Y_{tot} - \mu| \geq \epsilon n] \leq 2e^{-2\epsilon^2 n} = \delta$$

So that we have

$$\frac{n}{|\Gamma(w)|} \cdot x_{ji}(w) - \epsilon n \leq Y_{tot} \leq \frac{n}{|\Gamma(w)|} \cdot x_{ji}(w) + \epsilon n$$

Now if we multiply the inequalities by $|\Gamma(w)|/n$, we get the desired result with probability at least $1-\delta$ as follows,

$$x_{ji}(w) - \epsilon|\Gamma(w)| \leq \widehat{x}_{ji}(w) \leq x_{ji}(w) + \epsilon|\Gamma(w)|$$

If we set $n = \lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$ then with confidence at least $1-\delta$ the error rate in our estimate is at most ϵ.

■

Note that Lemma 2 naturally generalizes for the total estimated counts for any edge e_j and graphlet pattern G_i.

V. Edge to Node Graphlet Frequency

Besides the straightforward parallel performance and load-balancing advantages that arise from our edge-centric graphlet counting approach, the edge-centric graphlet counts can be used to efficiently derive the node-centric graphlet counts. However, the reverse is not possible, that is, given a set of vectors $\{\mathbf{x}_i \in \mathbb{R}^\kappa\}_{i=1}^N$ where $\mathbf{x}_i$ is the κ-dimensional vector for node $v_i \in V$, it is not possible to efficiently derive the counts $\{\mathbf{x}_k \in \mathbb{R}^\kappa\}_{k=1}^M$ where $\mathbf{x}_k$ is a vector of graphlet counts for an edge $e_k \in E$. Moreover, it is impractical and essentially amounts to recomputing the counts per edge.

Now, we demonstrate how to derive node graphlet counts efficiently using only edge counts from the edge-centric graphlet decomposition. For simplicity, let ω_{ij} be the count of an arbitrary graphlet pattern $\mathcal{G}$ for an edge $e_k = (v_i, v_j) \in E$, thus all $\omega_{ij} \geq 0$. Further, let $\mathbf{A} = [A_{ij}]$ be a sparse adjacency matrix where $A_{ij} = \omega_{ij}$ if there exists an edge $e_k = (v_i, v_j) \in E$, and $A_{ij} = 0$ otherwise. Let $\mathbf{D} = \text{diag}(\mathbf{Ae})$ be an $N \times N$ square diagonal matrix with the (weighted) degree of each vertex on the diagonal, and $\mathbf{d} = \mathbf{De}$ is the vector of degrees where $\mathbf{e}$ is the vector of all ones. Thus, $\mathbf{d} = [d_1 \cdots d_i \cdots d_n]$ where $d_i = 1/\alpha \sum_{v_j \in \Gamma(v_i)} A_{ij}$; d_i is the sum of graphlet counts over all he edges incident to vertex $v_i \in V$, and α is some normalization constant to account for counting the same pattern more than once. Suppose the graphlet pattern is 4-cliques, then $\mathbf{x} = {}^{\mathbf{d}}/_{3}$ where $\mathbf{x} \in \mathbb{R}^N$ is the vector of 4-clique counts for the n nodes in V (see Fig. 2 for an example). Notice the above transformation is extremely efficient. For a single node v_i, it is linear in the number of neighbors.

VI. Experiments

In this section, we investigate the effectiveness of the proposed *exact* and *estimation methods* from Section III for the local graphlet counting problem. We have also released all codes[3] and graph data sets[4] are also available for download [24].

Table III
Runtime results comparing exact variants.

	Class of graphlets (sec.)				
Graph	**3-graphlets**	**4-clique**	**4-cycle**	**tailed-tri**	**all $k \in \{3,4\}$ graphlets**
soc-flickr	0.11	6.75	18.23	73.93	80.67
soc-gowalla	0.01	0.2	0.47	3.85	4.05
socfb-MIT	0.003	0.06	0.27	0.75	0.81
socfb-Texas84	0.02	0.44	2.2	9.01	9.45
web-wikipedia09	0.03	0.11	0.51	2.49	2.61
bio-dmela	0.001	0.002	0.01	0.03	0.03
brain-mouse-ret1	0.002	0.13	0.14	0.62	0.75
tech-RL-caida	0.003	0.01	0.06	0.2	0.21

A. Exact methods

This section investigates the runtime performance of the following exact variants derived from Alg. 2 including:

(a) **3-graphlets**: a method that derives all the *3-graphlets* from a single quantity representing the triangles (3-cliques) centered at edge $e_j \in E$.

(b) **clique-based graphlets**: a method that finds only *4-cliques* and the other 4-graphlets that are directly derived from that quantity.

(c) **cycle-based graphlets**: a method that finds only *4-cycles* and the other 4-graphlets that are directly derived from that quantity.

(d) **tailed-triangles**: a method for finding *tailed-triangles* for each edge in G directly.

(e) **all $\{3,4\}$-graphlets**: a method that finds all graphlet counts from Table II for each edge in G.

We denote the graphlets derived from (b) and (c) as the class of clique-based and cycle-based graphlets. All the exact variants find the frequency of the connected and disconnected 3-graphlets, which are then used as a basis for deriving the others extremely efficiently.

See Table III for results. Note that (b)-(e) (last four columns of Table III) all include the time it takes to compute all 3-graphlets (first column in Table III), as these are used to derive the others efficiently. Observe that in most cases, the class of 4-clique graphlets are orders of magnitude faster than the others including graphlets based on 4-cycles. The only exception appears to be brain-mouse-ret1 as the

[3]www.github.com/nkahmed/pgd
[4]www.networkrepository.com

runtime of 4-cliques is very close to that of 4-cycle-based graphlets. Nevertheless, in all cases, the runtime of 4-cliques and 4-cycles is orders of magnitude faster than tailed-triangles (third column in Table III).

Unfortunately, there is no direct method for comparison, since existing local exact methods are limited to counting graphlets for each node (*i.e.*, are node graphlet counting methods) [25], [26], whereas our approach reveals edge graphlet features and counts for individual edges in the graph. Nevertheless, we found that our approach is significantly faster than existing local node graphlet methods such as FANMOD [11], ORCA [26], and RAGE [25]. Moreover, in many large network problems, these methods failed to finish after running for 12 hours. However, in the case of smaller networks (that these methods could handle), our approach was found to be more than 100 times faster, even despite the fact that these methods count graphlets for each node, as opposed to each edge — a fundamentally more challenging problem. For instance, our approach leads to a 498× improvement in runtime over RAGE [25] and a 19324× improvement over FANMOD [11] for an email communication network (ia-email-EU).

Table IV
LOCAL GRAPHLET ESTIMATION EXPERIMENTS. THE AVERAGE RELATIVE ERROR FOR THE TOP-1000 EDGES WITH LARGEST DEGREE $(d_v + d_u)$ ARE REPORTED BELOW USING $\pi_r = 0.01$. RESULTS SHOWN FOR COUNTS AND GRAPHLET FREQUENCY DIST.

		RELATIVE ERROR					
graph							
soc-gowalla	GFD	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$
	COUNT	0.008	0.008	0.001	0.006	$<10^{-4}$	0.0003
ca-dblp12	GFD	0.0001	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$
	COUNT	0.003	$<10^{-4}$	0.0003	$<10^{-4}$	0.0004	$<10^{-4}$
socfb-Texas84	GFD	0.001	0.001	0.001	0.001	0.001	0.001
	COUNT	0.031	0.075	0.013	0.042	0.002	0.002

Table V
KS-STATISTIC AND L1 DISTANCE RESULTS FOR LOCAL GRAPHLET ESTIMATION.

graph	**KS**	**L1**
soc-gowalla	0.0002	$<10^{-4}$
ca-dblp12	0.0002	$<10^{-4}$
socfb-Texas84	0.002	0.001

B. Estimation methods

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 count $\widehat{x}_{ji}$ of an arbitrary graphlet $G_i \in \mathcal{G}$ for edge $e_j \in E$, we consider the relative error: $\mathbb{D}\big(\widehat{x}_{ji} \,\|\, x_{ji}\big) = \frac{|\widehat{x}_{ji} - x_{ji}|}{x_{ji}}$ where x_{ji} is the actual count of G_i. The relative error indicates the quality of an estimated graphlet statistic relative to the magnitude of the actual statistic. Table IV shows the relative error for the top 1000 edges with highest cumulative degree $(d_v + d_u)$ for each edge $e_j = (v, u)$. In addition, we use Kolmogorov-Smirnov (KS) statistic and Normalized L_1 distance [27] to quantify the distance between the actual and estimated normalized frequency (graphlet frequency distribution GFD computed on the full spectrum of connected and disconnected graphlets). The KS and L_1 errors for a variety of graph problems are shown in Table V, and found to be extremely small. Thus, the estimation methods have excellent accuracy and the difference between the estimated and actual statistic is small and in most cases the difference is insignificant.

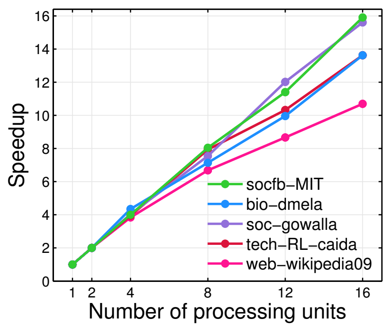


Figure 3. Strong scaling results for a variety of networks.

C. Scaling

This section investigates the parallel performance of the proposed method. The parallel algorithms for exact and approximate local graphlet counting have the following properties: lock-free updates due to the partitioning of edges across the processing units and the fact that each edge is guaranteed to be processed by a single worker, decentralized, and finally completely asynchronous due to the intrinsic lock-free property of our algorithm. We have also optimized memory to be closely aligned with cache lines. For these experiments[5], we used a machine with two Intel Xeon[6] E5-2687W v4 platform with 3.1GHz CPUs. Each processor has 8 cores with 20MB of L3 cache and 256KB of L2 cache. The machine has 128GB of memory, however, the method never came close to using all of it. The proposed method scales well as the number of processing units increase. In particular, *strong scaling* is observed in Fig. 3 for a variety

[5]Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance

[6]Intel, Xeon, and Intel Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

of graphs with different characteristics. Significant speedups are obtained from the different graphs. For all graph types we obtain linear or nearly linear speedups.

Table VI
COMPUTATIONAL COMPLEXITY

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

D. Complexity

Time Complexity: The computational complexity of our proposed algorithms is summarized in Table VI. Recall that we only need to compute a few graphlets and can directly obtain the others in constant time. To compute all local graphlets for a given edge, it takes: $\mathcal{O}\Big(\Delta_{\text{ub}}\big(|S_u| + |S_v| + |T_e|\big)\Big)$ where $\Delta_{\text{ub}} \leq \Delta$ is the maximum sampled degree from any vertex in S_v, S_u, and T_e. We place this upper bound Δ_{ub} on the number of neighbors searched from any vertex in S_v, S_u, and T_e by using sampling and estimation (as we show in Alg. 3). This allows us to reduce the time quite significantly[7]. 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.

Space Complexity: Given an edge e_j, our approach requires the frequency of triangles x_3, 4-cycles x_8, tailed-triangles x_7, and 4-cliques x_5, and from these we can derive all other graphlets of size $k \in \{3, 4\}$ directly in $o(1)$ time. Alg. 1 takes $\mathcal{O}(4M)$ only space to store the graphlets, and thus it is *space-efficient*, since the counts of all other graphlets can be derived from the few ones stored.

VII. APPLICATIONS

A. Relational Classification

Given a partially labeled (attributed) network G, a known set of node labels $Y^\ell = \{y_i | v_i \in V^\ell\}$ for nodes $V^\ell \subset V$, the within-network classification task is to infer Y^u for the set $V^u = V \setminus V^\ell$ of remaining vertices with unknown labels.

We compared exact and estimated local graphlet features from Alg. 2 and Alg. 3, respectively. We used 5-fold cross-validation with a label density of 10% (*e.g.*, the fraction of nodes with known labels is 10%.). For this experiment, we used the TerroristRel network. All self attributes and all other features besides the local graphlet-based features were discarded. This was done since we are interested in understanding the impact in predictive performance when the graphlet features for each edge (and node) are estimated as opposed to computed exactly. For simplicity, we used a recent approach called relational similarity machines (RSM), see [28] for more details. Though, we have also observed similar behavior using other relational and collective classification approaches. Using only the *exact* graphlet features gave 92.29% accuracy, whereas the *estimated* graphlet features resulted in 92.16% accuracy. The difference in predictive accuracy is not significantly different. Furthermore, the accuracy did not change for most other networks tested, including Cora, Citeseer, a Gene protein interaction network, and many smaller biological and chemical/molecular networks. We posit that in some cases, the estimated graphlet features could even lead to an increase in accuracy due to the possible benefits of generalization and reducing overfitting.

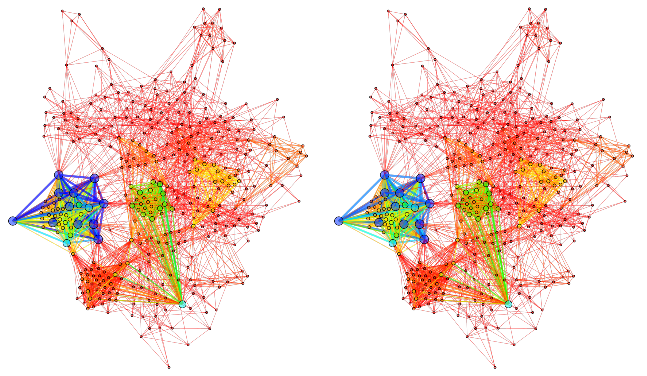


Figure 4. Edges and nodes from tech-routers are colored and weighted by the local 4-cliques. On the left, the exact local 4-cliques are used whereas the right is from our graphlet estimation framework with $\epsilon = 0.05$ and $\delta = 0.05$. Note that nodes with degree less than 15 were filtered (along with their links). See text for discussion.

B. Estimation Impact on Revealing Higher-order Structures

Recently, Ahmed et al. [10], [12] observed that by encoding the color and size of nodes and edges using the counts of specific network motifs/graphlets immediately reveal large structures that are otherwise difficult to observe and understand. Moreover, in that work, they use graphlets to find and rank large stars, cliques, and other larger higher-order structures, which are of fundamental importance in many types of networks such as technological/communication networks (*e.g.*, the Internet AS) as well as social networks [29]. More recently, Benson *et al.* [30] proposed a spectral clustering algorithm based on that key observation. We test the impact of our proposed local graphlet unbiased estimation on revealing higher-order structures. We observe that the visualizations in Fig 4(a) and Fig 4(b) using the exact and estimated graphlet features (to encode the color and size of the edges and nodes) are strikingly similar, with trivial differences. This finding justifies the local graphlet

[7]proof is omitted for brevity

estimation framework and its efficiency in revealing higher-order structure and organization in large networks.

VIII. Related Work

While there have been a number of exact/approximation methods for the *global graphlet counting problem*, *i.e.*, counting all graphlets in the graph G [11], [25], [26], [31], there are significantly fewer methods for the local graphlet problem [11], even despite its fundamental importance for a variety of machine learning tasks. Moreover, existing methods for local graphlet counting focus on counting graphlets centered around a vertex and/or counting/sampling a few particular graphlet patterns (*e.g.*, triangles and wedges) [32], [33]. In contrast, our approach counts *all* the local graphlets that surround an *edge*. Moreover, nearly all of the existing methods focus only on connected graphlets, whereas this work derives the full spectrum of graphlet patterns for each edge (both connected and disconnected)[8]. Moreover, most existing methods for counting local graphlets are sequential [11], [25], [26]. To the best of our knowledge, this work is the first to demonstrate the significant speedups that can be achieved using the power of sampling combined with an effective parallelism strategy for counting all local graphlets.

IX. Conclusion

This work proposed efficient parallel exact and estimation methods for the local graphlet counting problem. The methods were shown to be extremely effective across a wide variety of networks with fundamentally different structural properties. In particular, the estimation methods are strikingly accurate and fast with little noticeable difference between the exact and estimated graphlet counts.

References

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

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

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

[4] H. Kashima, H. Saigo, M. Hattori, and K. Tsuda, "Graph kernels for chemoinformatics," *Chemoinformatics and Advanced Machine Learning Perspectives*, 2010.

[5] 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.

[6] 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.

[7] P. W. Holland and S. Leinhardt, "Local structure in social networks," *Soc. Meth.*, vol. 7, pp. pp. 1–45, 1976.

[8] K. Faust, "A puzzle concerning triads in social networks: Graph constraints and the triad census," *Social Networks*, vol. 32, no. 3, pp. 221–233, 2010.

[9] O. Frank, "Triad count statistics," *Annals of Disc. Math.*, pp. 141–149, 1988.

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

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

[12] N. K. Ahmed, J. Neville, R. A. Rossi, N. Duffield, and T. L. Willke, "Graphlet decomposition: Framework, algorithms, and applications," in *KAIS*, 2016, pp. 1–32.

[13] 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.

[14] L. Getoor and B. Taskar, *Intro. to SRL.* MIT press, 2007.

[15] 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.

[16] C. C. Noble and D. J. Cook, "Graph-based anomaly detection," in *SIGKDD*, 2003.

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

[18] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, 2007.

[19] R. A. Rossi and N. K. Ahmed, "Role discovery in networks," *in TKDE*, 2015.

[20] ——, "Coloring large complex networks," *SNAM*, vol. 4, no. 1, pp. 1–37, 2014.

[21] 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.

[22] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *SIGKDD*, 2014.

[23] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *JASA*, vol. 58, no. 301, pp. 13–30, 1963.

[24] R. A. Rossi and N. K. Ahmed, "The Network Data Repository with Interactive Graph Analytics and Visualization," in *AAAI*, 2015.

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

[27] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *TKDD*, vol. 8, no. 2, pp. 1–56, 2014.

[28] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Relational Similarity Machines," in *MLG KDD*, 2016, pp. 1–9.

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

[30] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[31] 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.

[32] C. Seshadhri, A. Pinar, and T. G. Kolda, "Triadic measures on graphs: The power of wedge sampling," in *SDM*, 2013.

[33] Y. Lim and U. Kang, "Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams," in *SIGKDD*. ACM, 2015, pp. 685–694.

[8]Disconnected graphlets are known to be important – *e.g.*, [3] found that disconnected graphlets are *essential* for correct classification.