Exact and Estimation of Local Edge-centric Graphlet Counts

Nesreen K. Ahmed  
Parallel Computing Lab, Intel Corporation  
nesreen.k.ahmed@intel.com

Theodore L. Willke  
Parallel Computing Lab, Intel Corporation  
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 graphlet 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 a flexible, 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 \( k \in \{3,4\} \) for each edge. The framework gives rise to efficient, parallel, and accurate unbiased estimation methods as well as exact graphlet algorithms for counting graphlets locally. Experiments demonstrate the effectiveness of the proposed exact and estimation methods.

Keywords: Graphlets, local graphlet counts, edge graphlet counts, edge features, statistical estimation, induced subgraphs, motifs, network analysis, statistical relational learning, link classification, parallel algorithms.

1. Introduction

Graphlets are small induced subgraphs and are important for many predictive and descriptive modeling tasks (Pržulj et al., 2004; Milenkoviæ and Pržulj, 2008; Hayes et al., 2013) in a variety of disciplines including bioinformatics (Vishwanathan et al., 2010; Shervashidze et al., 2009), cheminformatics (Rupp and Schneider, 2010; Kashima et al., 2010), and image processing and computer vision (Zhang et al., 2016, 2013). Given a network \( G \), our approach counts the frequency of each \( k \in \{3,4\} \)-vertex induced subgraph patterns (See Table 2). These counts represent powerful features that succinctly characterize the fundamental network structure (Shervashidze et al., 2009). Indeed, it has been shown that such features accurately capture the local network structure in a variety of domains (Holland and Leinhardt, 1976; Faust, 2010; Frank, 1988). 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 (Shervashidze et al., 2009; Ahmed et al., 2015), in this paper however, we focus on local micro-level graphlet
statistics. Micro-level graphlet statistics $x_i$ of an individual edge $e_i \in E$ 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 $\{x_1, x_2, \ldots, x_M\}$ for improving statistical relational learning (SRL) tasks (Rossi et al., 2012) such as relational classification (Getoor and Taskar, 2007), link prediction and weighting tasks (e.g., recommending items, friends, web sites, music, events, etc.) (Liu et al., 2013), detecting anomalies in graphs (e.g., detecting fraud, or attacks/malicious behavior in computer networks) (Noble and Cook, 2003; Akoglu et al., 2014), among many others (Bhattacharya and Getoor, 2006; Schaeffer, 2007; Rossi and Ahmed, 2015b). 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 use cases and applications.

In this paper, we propose an exact, fast, efficient, and parallel algorithm for computing local graphlet statistics. Moreover, we combine our proposed algorithm with a statistical unbiased estimation approach 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. First, Section 2 provides background (notation, key definitions) along with a formulation of the problem. Next, Section 3 derives a flexible computational framework (for counting graphlets locally for each edge in $G$) that serves as a basis for the proposed methods. Section 4 provides a variety of experiments demonstrating the effectiveness of the methods, and Section 5 reviews related work. Finally, Section 6 concludes.

2. Local Graphlet Counting

This section formulates the local (micro\textsuperscript{1} graphlet estimation problem, then derives a flexible computational framework. Preliminaries are given in Section 2.1 and the problem formulation is provided in Section 2.2.

2.1. 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., $V = \{v_1, v_2, \ldots, v_i, \ldots, v_n\}$ such that $v_i-1$ appears before $v_i$ and so forth. Similarly, the ordered edges are denoted $E = \{e_1, e_2, \ldots, e_i, \ldots, e_m\}$. 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 1 for a summary of the key notation).

Given a graph $G$ and an edge $(v, u) \in E$, the edge-induced subgraph is simply $H = (W, E[W])$ where $W = \Gamma(v) \cup \Gamma(u)$ 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 $G = \text{\textsuperscript{1}}$ 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.
Exact and Estimation of Local Edge-centric Graphlet Counts

Table 1: Summary of notation. All sets are ordered. Whenever possible, we use standard terminology in the literature.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N)</td>
<td>Number of nodes in the graph (G)</td>
</tr>
<tr>
<td>(M)</td>
<td>Number of edges in the graph (G)</td>
</tr>
<tr>
<td>(J)</td>
<td>Set of selected edges via an arbitrary mechanism, e.g. a sampling procedure</td>
</tr>
<tr>
<td>(K)</td>
<td>Number of selected edges</td>
</tr>
<tr>
<td>(p_j)</td>
<td>Sampling probability for edge (e_j)</td>
</tr>
<tr>
<td>(d_v)</td>
<td>Degree of vertex (v) where (d_v =</td>
</tr>
<tr>
<td>(\Delta)</td>
<td>Maximum degree of the graph (G)</td>
</tr>
<tr>
<td>(T_e)</td>
<td>Set of vertices that form a triangle with edge ((v,u))</td>
</tr>
<tr>
<td>(S_u)</td>
<td>Set of vertices forming a 2-star (centered at vertex (v)) with edge ((v,u))</td>
</tr>
<tr>
<td>(\Gamma(e_j))</td>
<td>Edge neighborhood of (e_j = (v, u)), also denoted as (\Gamma(v, u)) for convenience. Let (\Gamma_k(e_j)) be the set of neighbors within (h)-hops of the edge (e_j). Note (\Gamma_k(e_j) = \Gamma_{h=1}(e_j)).</td>
</tr>
<tr>
<td>(\Psi(\cdot))</td>
<td>Fast lookup table for checking edge existence in constant time (i.e., (o(1)))</td>
</tr>
<tr>
<td>(G_i)</td>
<td>The (i^{th}) induced subgraph, see Table 2. For convenience, we also use (</td>
</tr>
<tr>
<td>(X)</td>
<td>Let (X \in \mathbb{R}^{M \times k}) be a matrix representing the local graphlet counts for all edges (e_j \in E).</td>
</tr>
<tr>
<td>(X_{ij})</td>
<td>Count of graphlet (G_i) for edge (e_j \in E)</td>
</tr>
<tr>
<td>(x_j)</td>
<td>A vector of local graphlet counts for edge (e_j \in E). For convenience, we also denote the local graphlet counts for (e_j) as (X_{i,j}).</td>
</tr>
</tbody>
</table>

\((V,E)\) together with all edges whose endpoints are both in this subset \(E_k = \{e \in E : e = (u,v) \land u,v \in V_k\} \). Let \(G^{(k)}\) denote the set of all possible \(k\)-vertex induced subgraphs and \(G = G^{(2)} \cup \ldots \cup G^{(k)}\) 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 \(G_i \in G^{(k)}\).

It is important to distinguish between the two fundamental classes of graphlets, namely, connected and disconnected graphlets (see Table 2). 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 \(G_i\), \(\forall u,v \in V_k, \exists P_{u-v} : u, \ldots, v, w, \ldots, v\), such that \(d(u,v) \geq 0 \land d(u,v) \neq \infty\) and \(d(u,v)\) is the distance (number of hops) between \(u\) and \(v\). 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.

2.2. 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)
Table 2: Summary of the graphlets of size \( k \in \{3, 4\} \). Graphlets are arranged into sets of \( k \)-graphlets and categorized into connected (left) and disconnected graphlets (right). Connected graphlets are ordered from most dense to least, whereas disconnected graphlets are arranged next to their corresponding complement connected graphlet (on the left).

<table>
<thead>
<tr>
<th>( k = 3 )</th>
<th>Connected</th>
<th>Disconnected</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_1 ) triangle</td>
<td>( G_4 ) 3-node-independent</td>
<td></td>
</tr>
<tr>
<td>( G_2 ) 2-star</td>
<td>( G_3 ) 3-node-1-edge</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( k = 4 )</th>
<th>Connected</th>
<th>Disconnected</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_5 ) 4-clique</td>
<td>( G_{15} ) 4-node-independent</td>
<td></td>
</tr>
<tr>
<td>( G_6 ) chordal-cycle(^\dagger)</td>
<td>( G_{14} ) 4-node-1-edge</td>
<td></td>
</tr>
<tr>
<td>( G_7 ) tailed-triangle(^\dagger)</td>
<td>( G_{13} ) 4-node-2-star</td>
<td></td>
</tr>
<tr>
<td>( G_8 ) 4-cycle</td>
<td>( G_{12} ) 4-node-2-edge</td>
<td></td>
</tr>
<tr>
<td>( G_9 ) 3-star</td>
<td>( G_{11} ) 4-node-1-triangle</td>
<td></td>
</tr>
<tr>
<td>( G_{10} ) 4-path</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^\dagger\) Diamond and chordal-cycle are interchangeable.
\(^\dagger\) Paw and tailed-triangle are interchangeable.

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 2 for all graphlet patterns of size \( k = \{3, 4\} \) nodes). For example, \(|G_1(e_j)|\) represents the number of triangles incident to edge \( e_j \in \mathcal{E} \). Note that counting \( k \)-vertex graphlets incident to any edge \( e_j \) can be performed naively in \( O(\Delta^{k-1}) \) asymptotically, where \( \Delta \) is the maximum vertex degree in the graph. Clearly, the time complexity of the algorithm is expensive for larger values of \( k \) and \( \Delta \). Therefore, we also provide an unbiased estimation for the counts of graphlets incident to an edge \( e_j \).

**Problem.** (Local Edge-Centric Graphlet Estimation) Given a graph \( G = (V, E) \) and an edge \( e_j = (v, u) \in \mathcal{E} \), the local edge-centric graphlet estimation problem is to find

\[
\mathbf{x}_j = [x_1 \ldots x_4 \ x_5 \ldots x_{10} \ldots x_{15}]
\]

where \( \mathbf{x}_j \) is an approximation of the exact local graphlet statistics denoted \( \mathbf{y}_j \) for edge \( e_j \) such that \( \mathbf{x}_j \approx \mathbf{y}_j \) and thus \( \mathbb{D}(\mathbf{x}_j \parallel \mathbf{y}_j) \) is minimized as well as the computational cost associated with the estimation. Moreover, \( \mathbf{x}_j \) is an unbiased estimate of \( \mathbf{y}_j \). Note that \( \mathbb{D}(\mathbf{x}_j \parallel \mathbf{y}_j) \) can be any loss/distance function (such as kolmogorov-smirnov distance (Ahmed et al., 2014b)). 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.

\(^2\) Graphlets are estimated locally (at the micro-level), that is, per edge as opposed to globally.
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 local graphlet decomposition method can easily be transformed into node features using techniques by Rossi et al. (2012) or used directly by SRL algorithms (Getoor and Taskar, 2007).

3. 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 3.1 discusses the initial preprocessing steps that significantly improve the efficiency of our method. Section 3.2 introduces the exact approach whereas the unbiased estimation method for local edge graphlet counts is proposed in Section 3.3.

3.1. 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 $d(v_1) \leq d(v_2) \leq \cdots \leq d(v_N)$.

**P2** For each $\Gamma(v_i) \in \{\Gamma(v_1), \ldots, \Gamma(v_N)\}$, the vertex neighbors in $\Gamma(v_i) = \{w_j, \ldots, w_k, \ldots\}$ are ordered 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 $\pi$ be an ordering of the set of edges by an arbitrary graph property $f(\cdot)$ 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 (**P1 – P4**) give rise to many useful properties and leads to significant reduction in runtime. For finding the 4-cycles centered at an edge $e_j = (v, u) \in E$, we avoid searching $S_v$ completely, and instead search only $S_u$, which due to **P3** is likely to be much less expensive than searching $S_v$. All the work above takes either $O(N)$ or $O(M)$ time and is computed in parallel.
Algorithm 1 Edge-centric family of Algorithms for local graphlet counts

Input:
- a graph $G = (V, E)$
- an ordered set of edges $J = \{e_1, \ldots, e_K\} \subseteq E$ where $K = |J|$ and $\phi = \lceil K/M \rceil$
- an edge sampling probability vector $p = [p_1 \ldots p_K]$ for each $e_j \in J$. Note that if $p$ is the vector of all ones, then graphlets are computed exactly for each edge in $J$.

1: Compute preprocessing steps from Section 3.1
2: parallel for each $e_j \in J$ in order do
3: Obtain exact local graphlet counts $x_j$ for $e_j$ via Alg. 2 or use Alg. 3 (with $p_j$) to obtain a fast and accurate unbiased estimation of the local graphlet counts $x_j$ for edge $e_j$
4: Set $X_j$ to be $x_j^\top$
5: end parallel
6: return $X = [x_1 \ldots x_j \ldots x_K]^\top$ consisting of the local graphlet counts $x_j$ for each edge $e_j \in J$

3.2. Exact Algorithm

Given a set of edges $J \subseteq E$ where $K = |J|$, Alg. 1 computes $X = [x_1 \ldots x_k \ldots x_K]^\top$ consisting of the local graphlet counts $x_j$ for each edge $e_j \in J$. The exact method for deriving local graphlet counts $x_j$ centered at an individual edge $e_j$ is given in Alg. 2.

3.2.1. 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 2). The local (micro-level) 3-graphlets for edge $e_j$ are as follows:

\begin{align*}
    x_1 &= |T_e| && (1) \\
    x_2 &= (d_u + d_v - 2) - 2|T_e| && (2) \\
    x_3 &= N - x_2 - |T_e| - 2 && (3) \\
    x_4 &= \left( \frac{N}{3} \right) - (x_1 + x_2 + x_3) && (4)
\end{align*}

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:

\begin{align*}
    |S_u| &= d_u - |T_e| - 1 && (5) \\
    |S_v| &= d_v - |T_e| - 1 && (6)
\end{align*}

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).
3.2.2. 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:

$$x_6 = \left( \frac{|T_e|}{2} \right) - x_5$$  \hspace{1cm} (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:

$$x_{10} = (|S_v| \cdot |S_u|) - x_8$$  \hspace{1cm} (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:

$$x_9 = \left( \frac{|S_v|}{2} \right) + \left( \frac{|S_u|}{2} \right) - x_7$$  \hspace{1cm} (9)

3.2.3. Local Disconnected 4-graphlets

Disconnected graphlets are derived as follows:

$$x_{11} = |T_e| \cdot \left[ N - (|T_e| + |S_u| + |S_v| + 2) \right]$$  \hspace{1cm} (10)

$$x_{12} = (|S_u| + |S_v|) \cdot \left[ N - (|T_e| + |S_u| + |S_v| + 2) \right]$$  \hspace{1cm} (11)

$$x_{13} = M - (|T_e| + d_u + d_v + 1) - \sigma$$  \hspace{1cm} (12)

$$x_{14} = \left( N - \left[ |T_e| + |S_u| + |S_v| + 2 \right] \right)$$  \hspace{1cm} (13)

$$x_{15} = \left( \frac{N}{5} \right) - \sum_{i=1}^{14} x_i$$  \hspace{1cm} (14)

3.3. Statistical Estimation

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 $e_i$, a graph $G$, a sampling probability $p_j$, and it returns the graphlet feature vector $x_j$ for edge $e_j \in E$. 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 local graphlet approximation methods have many attractive properties such as unbiasedness, consistency, among others.

The algorithm estimates local graphlet properties including local single-valued statistics and multi-valued distributions (for a given edge or set of edges).

Given $w \in T_e$ (or $S_u, S_v$), we propose selecting $r \in \Gamma(w)$ (a neighbor of $w$) with probability $p_j$ accordingly to an arbitrary weighted/uniform distribution $F$. Alg. 3 shows how to efficiently obtain an unbiased estimate of the graphlet counts of size $k \in \{3, 4\}$ for an edge $e_j \in E$. 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. In particular, Line 11 searches each vertex $w \in T_e$ in parallel. Given $w \in T_e$, we select a neighbor $r \in \Gamma(w)$
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 \( x_j \) 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. \( \text{if } w \neq u \text{ then } S_v \leftarrow S_v \cup \{w\} \text{ and } \Psi(w) = \lambda_1 \) 
5. parallel for each \( w \in \Gamma(u) \text{ and } w \neq v \) do 
6. \( \text{if } \Psi(w) = \lambda_1 \text{ then } \) 
7. \( T_u \leftarrow T_u \cup \{w\} \text{ and set } \Psi(w) = \lambda_3 \) \( \triangleright \) triangle 
8. \( S_v \leftarrow S_v \setminus \{w\} \) 
9. \( \text{else } S_u \leftarrow S_u \cup \{w\} \text{ and set } \Psi(w) = \lambda_2 \) \( \triangleright \) wedge 
10. parallel for each \( w \in T_v \) do 
11. \( \text{for } r \in \Gamma(w) \text{ do } \) 
12. \( \text{if } \Psi(r) = \lambda_3 \text{ then } \) Set \( x_5 \leftarrow x_5 + 1 \) \( \triangleright \) local 4-clique 
13. Set \( \Psi(w) \) to \( \lambda_4 \) 
14. parallel for each \( w \in S_u \) do 
15. \( \text{for } r \in \Gamma(w) \text{ do } \) 
16. \( \text{if } \Psi(r) = \lambda_1 \text{ then } \) Set \( x_8 \leftarrow x_8 + 1 \) \( \triangleright \) local 4-cycle 
17. \( \text{if } \Psi(r) = \lambda_2 \text{ then } \) Set \( x_7 \leftarrow x_7 + 1 \) \( \triangleright \) local tailed-tri 
18. \( \text{if } \Psi(r) = \lambda_4 \text{ then } \) Set \( \sigma \leftarrow \sigma + 1 \) 
19. Set \( \Psi(w) \) to \( \lambda_4 \) 
20. parallel for each \( w \in S_v \) do 
21. \( \text{for } r \in \Gamma(w) \text{ do } \) 
22. \( \text{if } \Psi(r) = \lambda_1 \text{ then } \) Set \( x_7 \leftarrow x_7 + 1 \) \( \triangleright \) local tailed-tri 
23. \( \text{if } \Psi(r) = \lambda_4 \text{ then } \) Set \( \sigma \leftarrow \sigma + 1 \) 
24. Set \( \Psi(w) \) to \( \lambda_4 \) 
25. Derive local 3-graphlets for \( e_j \) via Eq.1-4 
26. Use Eq.7–9 to derive the local connected 4-graphlets for \( e_j \) and disconnected 4-graphlets via Eq.10–14. 
27. return \( x \), where \( x_i \) is the estimate of graphlet \( G_i \) for \( e_j \)

with probability \( p_j \) according to an arbitrary weighted/uniform distribution function \( F \). In this paper, we select \( r \) uniformly at random. Then, we check if \( r \) is of type \( \lambda_3 \) (from Line 8), 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. We use Horvitz-Thompson estimator to obtain the unbiased estimate of 4-clique counts for edge \( e_j \) (Ahmed et al., 2014a). Thus, the count of 4-cliques for edge \( e_j \) is weighted by its sampling probability. Line 16 ensures that the same 4-clique is not counted twice. Further, 4-cycles are computed in Lines 17-24 as well as a fraction of the tailed-triangles. The remaining tailed-triangles are computed in Lines 25-31. As an aside, \( \sigma \) is also computed (Lines 17-31) and used for estimating \( x_{13} \) for graphlet \( G_{13} \) (Eq. 12). Finally, the remaining graphlets \( \{x_{9}, \ldots, x_{15}\} \) 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 \( p_j = 1 \) and selection is performed without replacement.

Observe that vertices can also be selected directly from \( T_v \), \( S_u \), and \( S_v \). However, that approach does not share the same guarantees \( \text{w.r.t. time and space. For instance, suppose} \)
In this section, we investigate the effectiveness of the proposed exact methods from Section 3 for the local graphlet counting problem. In particular, Section 4.1 demonstrates the efficiency of the exact method and a few different exact variants. Finally,

$$w \in S_u$$ is sampled and $$\Gamma(w)$$ is searched, thus, $$|\Gamma(w)| = \Delta$$ in the worst case (which is fairly likely), and thus $$O(|T_e| + |S_u| + |S_v|)$$, which is no different than the exact method.

4. Experiments

In this section, we investigate the effectiveness of the proposed exact and estimation methods from Section 3 for the local graphlet counting problem. In particular, Section 4.1 demonstrates the efficiency of the exact method and a few different exact variants. Finally,
Section 4.3 provides the space and time complexity of the methods. We have also released all codes\(^3\) and graph data sets\(^4\) are also available for download (Rossi and Ahmed, 2015a).

### 4.1. 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 connected and disconnected 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 2 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. This result is largely inspired by the recent state-of-the-art global graphlet counting algorithm proposed by Ahmed et al. (2015, 2016).

See Table 3 for results. Note that (b)-(e) (last four columns of Table 3) all include the time it takes to compute all 3-graphlets (first column in Table 3), 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 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 3).

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) (Marcus and Shavitt, 2012; Hočevar and Demšar, 2014), whereas our approach reveals edge graphlet features and counts for individual edges in the graph. Nevertheless, our approach was found to be significantly faster than existing local node graphlet methods such as FANMOD (Wernicke and Rasche, 2006) and RAGE (Marcus and Shavitt, 2012). 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 10-1000 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 (Marcus and Shavitt, 2012) and a 19324× improvement over FANMOD (Wernicke and Rasche, 2006) for an email communication network (ia-email-EU).

---

\[3. \text{www.github.com/nkahmed/pgd}\]

\[4. \text{www.networkrepository.com}\]
Table 3: Results comparing different exact variants.

<table>
<thead>
<tr>
<th>GRAPH</th>
<th>3-graphlets</th>
<th>4-clique</th>
<th>4-cycle</th>
<th>tailed-tri</th>
<th>all k ∈ {3, 4}</th>
</tr>
</thead>
<tbody>
<tr>
<td>soc-flickr</td>
<td>0.11</td>
<td>6.75</td>
<td>18.23</td>
<td>73.93</td>
<td>80.67</td>
</tr>
<tr>
<td>soc-gowalla</td>
<td>0.01</td>
<td>0.2</td>
<td>0.47</td>
<td>3.85</td>
<td>4.05</td>
</tr>
<tr>
<td>socfb-MIT</td>
<td>0.003</td>
<td>0.06</td>
<td>0.27</td>
<td>0.75</td>
<td>0.81</td>
</tr>
<tr>
<td>socfb-Texas84</td>
<td>0.02</td>
<td>0.44</td>
<td>2.2</td>
<td>9.01</td>
<td>9.45</td>
</tr>
</tbody>
</table>

4.2. 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 $x_i$ of an arbitrary graphlet $G_i \in G$ for edge $e_j \in E$, we consider the relative error: $D(x_i \parallel y_i) = \frac{|x_i - y_i|}{y_i}$ where $y_i$ 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. Results are shown in Table 4. We compute the relative error between the actual GFD (i.e., graphlet frequency distribution) (and count) from the exact method and the estimated GFD (and count); and find in all cases that the difference is extremely small (Table 4). In addition, Kolmogorov-Smirnov (KS) statistic and Normalized $L_1$ are used to quantify the the average relative error across the full spectrum of connected and disconnected graphlets. KS-statistic quantifies the maximum vertical distance between the actual cumulative distribution functions (CDF) $F$ and the estimated $\hat{F}$ as $D_{KS}(F \parallel \hat{F}) = \max_x ||F(x) - \hat{F}(x)||$ where $x$ represents the range of the random variable. Normalized $L_1$ is computed over the counts directly (as opposed to the various distributions) and measures the average relative error between $y$ and $x$ as $D_{L_1}(y \parallel x) = \frac{1}{|x|} \sum_i |y_i - x_i|$. The KS and $L_1$ error for a variety of graph problems are shown in Table 5, 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. In addition to the excellent accuracy, the estimation methods are between 900-1000K times faster with probability $p_j = 0.001$, and thus extremely fast for large networks.

4.3. Complexity

Time and space complexity of Alg. 1 is given below.

4.3.1. Time

The computational complexity is summarized in Table 6. Note that just as before, we only need to compute a few graphlets and can directly obtain the others in constant time. To compute all local graphlets for a given edge, it takes: $O\left(\Delta_{ub}(|S_v| + |S_u| + |T_e|)\right)$ where $\Delta_{ub}$ is the maximum degree from any vertex in $S_v$, $S_u$, and $T_e$. Alternatively, we can place an upper bound $\Delta_{ub}$ on the number of neighbors searched from any vertex in $S_v$, $S_u$, and $T_e$. We achieve this by using sampling and estimation (as we show in Alg. 3). This allows us to reduce the time quite significantly. The intuition is that for vertices with large
Table 4: Local graphlet estimation experiments. The average relative error for the top-1000 edges with largest degree \((d_v + d_u)\) are reported below using \(p_e = 0.01\). We also used a lower bound of \(\beta = 100\). This implies that if a search vertex (e.g., \(w \in S_u\)) has less neighbors than the lower bound \(\beta\), for a particular class of graphlets (e.g. 4-cycles), then estimation is not used, as the benefit is minimal. We also corrected for over- and under- estimation whenever possible using local lower and upper bounds for the expected frequency of a particular graphlet. The results below are for networks with significantly different structural characteristics.

<table>
<thead>
<tr>
<th>graph</th>
<th>RELATIVE ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GFD</td>
</tr>
<tr>
<td>soc-gowalla</td>
<td>count</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>graph</th>
<th>KS</th>
<th>L1</th>
</tr>
</thead>
<tbody>
<tr>
<td>soc-gowalla</td>
<td>0.0002</td>
<td>&lt;10^{-4}</td>
</tr>
<tr>
<td>socfb-Texas84</td>
<td>0.002</td>
<td>0.001</td>
</tr>
</tbody>
</table>

neighborhoods we only need to observe a relatively small (but representative) fraction of it to accurately extrapolate to the unobserved neighbors and their structure.

4.3.2. Space

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. Thus, Alg. 1 takes \(O(4M)\) only space to store the graphlets. Therefore, our algorithm is space-efficient, since the counts of all other graphlets can be derived from the few ones stored. Moreover, since the algorithm is parallelized via \(M\) independent edge neighborhood computations, then the graphlet counts for each edge are computed by exactly one worker, and therefore no extra space is required as the worker stores the counts for \(e_j\) in the \(k^{th}\) position. However, each worker requires a hash table of size \(N\). Nevertheless, if memory is limited, one can leverage cuckoo filters (Fan et al., 2014), bloom filters, or any other space-efficient probabilistic data structure. Furthermore, instead of the hash table, one can use binary search over the set of triangle vertices \(T_e\) and the set of 2-star vertices \(S_u\) (and \(S_v\)).
Table 6: Computational complexity

<table>
<thead>
<tr>
<th></th>
<th>Global</th>
<th>Local</th>
<th>Graphlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(K \Delta T_{\text{max}}))</td>
<td>(O(\Delta_{ub} \cdot</td>
<td>T_e</td>
<td>))</td>
</tr>
<tr>
<td>(O(K \Delta S_{\text{max}}))</td>
<td>(O(\Delta_{ub} \cdot</td>
<td>S_u</td>
<td>))</td>
</tr>
<tr>
<td>(O(K \Delta(S_{\text{max}} + T_{\text{max}})))</td>
<td>(O(\Delta_{ub}(</td>
<td>S_u</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all</td>
<td></td>
</tr>
</tbody>
</table>

5. Related Work

**Exact algorithms:** While there have been a number of exact methods for the *global graphlet counting problem*, i.e., counting all graphlets in the graph \(G\) (Wernicke and Rasche, 2006; Marcus and Shavitt, 2012), there are significantly fewer methods for the local graphlet problem (Wernicke and Rasche, 2006), even despite its fundamental importance. 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 (Ahmed et al., 2015, 2016). However, all such methods for the local graphlet counting problem focus on counting graphlets centered around a vertex. In contrast, our approach counts the local graphlets that surround an edge. Edge graphlet features are by definition more descriptive and powerful than those counted on the vertices. This property arises when there are far more edges than vertices, and thus, each edge captures more information. An alternative view that may be useful for intuition is that edges and vertices can be thought of as parameters in a model, which in our case the model can be viewed as the graph. Therefore, a model with more parameters is obviously able to represent the data better, and in the same manner, counting graphlets for each edge as opposed to vertex represents the graph more closely. We also posit that the vertex counts for an arbitrary graphlet \(G_i\) can be derived from the edge counts of the same graphlet pattern \(G_i\), however, the converse is not true by applying the above result. Moreover, nearly all of the existing vertex methods focus only on connected graphlets, whereas this work derives the full spectrum of graphlet patterns for each edge consisting of both connected and disconnected graphlets\(^5\).

**Estimation techniques:** There have been a number of recent approximation methods for counting graphlets globally over the entire graph (Rahman et al., 2014b,a). However, this work is the first to formulate the local graphlet estimation problem, along with extremely accurate, efficient, and unbiased local graphlet estimation methods. The methods are largely inspired by the work of Ahmed et al. (2014a).

**Parallel algorithms:** Unfortunately, existing methods for counting local graphlets are all sequential. This work is the first to demonstrate the significant speedups that can be achieved using an effective parallelization strategy. We expect that future research will further expand on the capabilities of the proposed parallel algorithm, as well as adapt existing methods for computing graphlets in parallel.

---

5. Disconnected graphlets are known to be important — e.g., Shervashidze et al. (2009) found that disconnected graphlets are essential for correct classification.
6. Conclusion

This work proposed efficient 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. Additionally, all methods were also parallelized and shown to scale well as the number of processing units increases.

References


Hisashi Kashima, Hirotosai Ogo, Masahiro Hattori, and Koji Tsuda. Graph kernels for
chemoinformatics. Chemoinformatics and Advanced Machine Learning Perspectives:
Complex Computational Methods and Collaborative Techniques, page 1, 2010.

Nathan N Liu, Luheng He, and Min Zhao. Social temporal collaborative ranking for context
aware movie recommendation. ACM Transactions on Intelligent Systems and Technology,


Tijana Milenkovic and Nataša Pržulj. Uncovering biological network function via graphlet


N Pržulj, Derek G Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric?

Mahmudur Rahman, Mansurul Alam Bhuiyan, Mahmuda Rahman, and Mohammad


Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph
analytics and visualization. In Proceedings of the Twenty-Ninth AAAI Conference on
Artificial Intelligence, 2015a.


graph data for statistical relational learning. Journal of Artificial Intelligence Research,

Matthias Rupp and Gisbert Schneider. Graph kernels for molecular similarity. Molecular


Nino Shervashidze, Tobias Petri, Kurt Mehlhorn, Karsten M Borgwardt, and Svn Vish-

S.V.N Vishwanathan, Nicol Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph
