# Efficient Graphlet Counting for Large Networks

Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi
Department of Computer Science
Purdue University
{nkahmed, neville, rrossi}@purdue.edu

Nick Duffield
Department of Electrical & Computer Engineering
Texas A&M University
duffieldng@tamu.edu

***Abstract*—From social science to biology, numerous applications often rely on graphlets for intuitive and meaningful characterization of networks at both the global macro-level as well as the local micro-level. While graphlets have witnessed a tremendous success and impact in a variety of domains, there has yet to be a fast and efficient approach for computing the frequencies of these subgraph patterns. However, existing methods are not scalable to large networks with millions of nodes and edges, which impedes the application of graphlets to new problems that require large-scale network analysis. To address these problems, we propose a fast, efficient, and parallel algorithm for counting graphlets of size $k = \{3, 4\}$-nodes that take only a fraction of the time to compute when compared with the current methods used. The proposed graphlet counting algorithms leverages a number of proven combinatorial arguments for different graphlets. For each edge, we count a few graphlets, and with these counts along with the combinatorial arguments, we obtain the exact counts of others in constant time. On a large collection of $300$+ networks from a variety of domains, our graphlet counting strategies are on average $460$x faster than current methods. This brings new opportunities to investigate the use of graphlets on much larger networks and newer applications as we show in the experiments. To the best of our knowledge, this paper provides the largest graphlet computations to date as well as the largest systematic investigation on over $300$+ networks from a variety of domains.**

***Keywords*—*graphlet; motif counting; graph kernel; parallel method; graph classification; visual analytics***

## I. Introduction

Recursive decomposition of networks is a widely used approach in network analysis to factorize the complex structure of real-world networks into small subgraph patterns of size $k$ nodes. These patterns are called *graphlets* [1]. Graphlets (also known as motifs [2]) are defined as subgraph patterns recurring in real-world networks at frequencies that are statistically significant from those in random networks. Given a network, we can count the number of embedding of each graphlet in the network, creating a profile of sufficient statistics that characterizes the network structure [3]. While knowing the graphlet frequencies does not uniquely define the network structure, it has been shown that graphlet frequencies often carry significant information about the local network structure in a variety of domains [4]–[6]. This is in contrast to global topological properties (e.g., diameter, degree distribution), where networks with similar/exact global topological properties can exhibit significantly different local structures.

### A. Graphlets, Scalability, & Applications

From social science to biology, graphlets have found numerous applications and were used as the building blocks of network analysis [2]. In social science, graphlet analysis (typically known as $k$-subgraph census) is widely adopted in sociometric studies [4], [6]. Much of the work in this vein focused on analyzing triadic tendencies as important structural features of social networks (e.g., transitivity or triadic closure) as well as analyzing triadic configurations as the basis for various social network theories (e.g., social balance, strength of weak ties, stability of ties, or trust [7]). In biology [1], [8], graphlets were widely used for protein function prediction [3], network alignment [9], and phylogeny [10] to name a few. More recently, there has been an increased interest in exploring the role of graphlet analysis in computer networking [11]–[13] (e.g., for web spam detection, analysis of peer-to-peer protocols and Internet AS graphs), chemoinformatics [14], [15], image segmentation [16], among others [17].

While graphlet counting and discovery have witnessed a tremendous success and impact in a variety of domains from social science to biology, there has yet to be a fast and efficient approach for computing the frequencies of these patterns. For instance, Shervashidze et al. [3] takes hours to count graphlets on relatively small biological networks (i.e., few hundreds/thousands of nodes/edges) and uses such counts as features for graph classification [18]. Previous work showed that graphlet counting is computationally intensive since the number of possible $k$-subgraphs in a graph $G$ increases exponentially with $k$ in $\mathcal{O}(|V|^k)$ and can be computed in $\mathcal{O}(|V|.\Delta^{k-1})$ for any bounded degree graph, where $\Delta$ is the maximum degree of the graph [3].

To address these problems, we propose a fast, efficient, and parallel algorithm for counting graphlets of size $k = \{3, 4\}$-nodes that take only a fraction of the time to compute when compared with the current methods used. The proposed graphlet counting algorithm leverages a number of proven combinatorial arguments for different graphlets. For each edge, we count a few graphlets, and with these counts along with the combinatorial arguments, we obtain the exact counts of others in constant time. On a large collection of 300+ networks from a variety of domains, our graphlet counting strategies are on average $460$x faster than current methods. This brings new opportunities to investigate the use of graphlets on much larger networks and newer applications as we show in our experiments. To the best of our knowledge, this paper provides the largest graphlet computations to date as well as the largest systematic investigation on over 300+ networks.

Furthermore, a number of important machine learning tasks are likely to benefit from such an approach, including graph anomaly detection [19], as well as using graphlets as features for improving community detection [20], role discovery [21], graph classification [18], and relational learning [22].

We test the scalability of our proposed approach experimentally on 300+ networks from a variety of domains, such as biological, social, and technological domains. We compare our approach to the state-of-the-art exact counting methods such as RAGE [23], FANMOD [24], and Orca [25]. We found that RAGE [23] took 2400 seconds to count graphlets on a small 26k node graph, whereas our proposed method is 460x faster, taking only 0.01 seconds. We also note that FANMOD [24], another recent approach, takes 172800 seconds, and Orca [25] takes 2.5 seconds for the same small graph. Our exact graphlet analysis is well-suited for shared-memory multi-core architectures (CPU and GPU), distributed architectures (MPI), and hybrid implementations that leverage the advantages of both.

### B. Contributions

- **Algorithms.** A fast, efficient, and parallel graphlet counting algorithm that leverages a number of combinatorial arguments that we show for different graphlets. The combinatorial arguments we show in this paper enable us to obtain significant improvement on the scalability of graphlet counting.
- **Scalability.** The proposed graphlet counting algorithm achieves on average 460x runtime improvement over the state-of-the-art methods. In addition, we analyze graphlet counts on graphs of sizes that are beyond the scope of the state-of-the-art (e.g., on graphs with hundred million nodes and billion edges).
- **Effectiveness.** Largest graphlet computations to date and largest systematic evaluation on over 300+ large-scale networks from a variety of domains.
- **Applications.** We systematically investigate a variety of existing and new applications for graphlet counting, such as finding unique patterns in graphs, as well as graph similarity and classification.

## II. Background

Graphlets are subgraph patterns recurring in real-world networks at frequencies that are significantly higher than those in random networks [1], [2]. Previous work showed that graphlets can be used to define universal classes of networks [2]. Moreover, graphlets are at the heart and foundation of many network analysis tasks (e.g., network classification, network alignment, etc.) [1], [8], [26]. In this paper, we introduce an efficient algorithm to compute the number of embedding of each graphlet of size $k = \{2, 3, 4\}$ nodes in the network.

### A. Notation and Definitions

Given an undirected simple input graph $G = (V, E)$, a graphlet of size $k$ nodes is defined as any subgraph $G_k \subset G$ which consists of a subset of $k$ nodes of the graph $G$. In this paper, we mainly focus on computing the frequencies of induced graphlets. An *induced* graphlet is an induced subgraph that consists of *all* edges between its nodes that are present in the input graph (as described in Definition 1). In addition, we distinguish between *connected* and *disconnected* graphlets (see Table I). A graphlet is connected if there is a path from any node to any other node in the graphlet (see Definition 2).

TABLE I. Summary of graphlet notation

Summary of the notation and properties for the graphlets of size $k = \{2, 3, 4\}$. Note that $\rho$ denotes density, $\Delta$ and $\bar{d}$ denote the max and mean degree, whereas assortativity is denoted by $r$. Also, $|T|$ denotes the total number of triangles, $K$ is the max k-core number, $\chi$ denotes the Chromatic number, whereas D denotes the diameter, B denotes the max betweenness, and $|C|$ denotes the number of components. Note that if $|C| > 1$, then $r$, D, and B are from the largest component.

| | **Graphlet** | Description | Complement | $\rho$ | $\Delta$ | $\bar{d}$ | $r$ | $\lvert T\rvert$ | $K$ | $\chi$ | D | B | $\lvert C\rvert$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(k = 4)$—Graphlets | | | | | | | | | | | | | |
| Connected | $g_{4_1}$ | 4-clique | | 1.00 | 3 | 3.0 | 1.00 | 4 | 3 | 4 | 1 | 0 | 1 |
| | $g_{4_2}$ | 4-chordalcycle | | 0.83 | 3 | 2.5 | -0.66 | 2 | 2 | 3 | 2 | 1 | 1 |
| | $g_{4_3}$ | 4-tailedtriangle | | 0.67 | 3 | 2.0 | -0.71 | 1 | 2 | 3 | 2 | 2 | 1 |
| | $g_{4_4}$ | 4-cycle | | 0.67 | 2 | 2.0 | 1.00 | 0 | 2 | 2 | 2 | 1 | 1 |
| | $g_{4_5}$ | 3-star | | 0.50 | 3 | 1.5 | -1.00 | 0 | 1 | 2 | 2 | 3 | 1 |
| | $g_{4_6}$ | 4-path | | 0.50 | 2 | 1.5 | -0.50 | 0 | 1 | 2 | 3 | 2 | 1 |
| Disconnected | $g_{4_7}$ | 4-node-1-triangle | | 0.50 | 2 | 1.5 | 1.00 | 1 | 2 | 3 | 1 | 0 | 2 |
| | $g_{4_8}$ | 4-node-2-star | | 0.33 | 2 | 1.0 | -1.00 | 0 | 1 | 2 | 2 | 1 | 2 |
| | $g_{4_9}$ | 4-node-2-edge | | 0.33 | 1 | 1.0 | 1.00 | 0 | 1 | 2 | 1 | 0 | 2 |
| | $g_{4_{10}}$ | 4-node-1-edge | | 0.17 | 1 | 0.5 | 1.00 | 0 | 1 | 2 | 1 | 0 | 3 |
| | $g_{4_{11}}$ | 4-node-independent | | 0.00 | 0 | 0.0 | 0.00 | 0 | 0 | 1 | $\infty$ | 0 | 4 |
| $(k = 3)$—Graphlets | | | | | | | | | | | | | |
| | $g_{3_1}$ | triangle | | 1.00 | 2 | 2.0 | 1.00 | 1 | 2 | 3 | 1 | 0 | 1 |
| | $g_{3_2}$ | 2-star | | 0.67 | 2 | 1.33 | -1.00 | 0 | 1 | 2 | 2 | 1 | 1 |
| | $g_{3_3}$ | 3-node-1-edge | | 0.33 | 1 | 0.67 | 1.00 | 0 | 1 | 2 | 1 | 0 | 2 |
| | $g_{3_4}$ | 3-node-independent | | 0.00 | 0 | 0.00 | 0.00 | 0 | 0 | 1 | $\infty$ | 0 | 3 |
| $(k = 2)$—Graphlets | | | | | | | | | | | | | |
| | $g_{2_1}$ | edge | | 1.00 | 1 | 1.0 | 1.00 | 0 | 1 | 2 | 1 | 0 | 1 |
| | $g_{2_2}$ | 2-node-independent | | 0.00 | 0 | 0.0 | 0.00 | 0 | 0 | 1 | $\infty$ | 0 | 2 |

Table I provides a summary of the notation and properties of all possible induced graphlets of size $k = \{2, 3, 4\}$.

**Definition 1.** Induced Graphlet: an induced graphlet $G_k = (V_k, E_k)$ is a subgraph that consists of a subset of $k$ vertices of the graph $G = (V, E)$ (i.e., $V_k \subset V$) together with all the edges whose endpoints are both in this subset (i.e., $E_k = \{\forall e \in E \,|\, e = (u, v) \wedge u, v \in V_k\}$).

**Definition 2.** Connected Graphlet: a graphlet $G_k = (V_k, E_k)$ is connected when there is a path from any node to any other node in the graphlet (i.e., $\forall u, v \in V_k, \exists P_{u-v} : u, ..., w, ..., v$, such that $d(u, v) \geq 0 \wedge d(u, v) \neq \infty$). By definition, there exist one and only one connected component in a graphlet $G_k$ (i.e., $|C| = 1$) if and only if $G_k$ is connected.

**Problem Definition.** Given a family of graphlets of size $k$ nodes $\mathcal{G}_k = \{g_{k_1}, g_{k_2}, ..., g_{k_m}\}$, our goal is to count the number of embeddings (appearances) of each graphlet $g_{k_i} \in \mathcal{G}_k$ in the input graph $G$. In other words, we need to count the number of induced graphlets $G_k$ in $G$ that are isomorphic to each graphlet $g_{k_i} \in \mathcal{G}_k$ in the family, such a number is denoted by $\binom{G}{g_{k_i}}$ [27]. A graphlet $g_{k_i} \in \mathcal{G}_k$ is embedded in the graph $G$, if and only if there is an injective mapping $\sigma : V_{g_{k_i}} \rightarrow V$, with $e = (u, v) \in E_{g_{k_i}}$ if and only if $e' = (\sigma(u), \sigma(v)) \in E$. Table I shows that $|\mathcal{G}_k| = \{2, 4, 11\}$ when $k = \{2, 3, 4\}$ respectively. Further, given a family $\mathcal{G}_k = \{g_{k_1}, g_{k_2}, ..., g_{k_m}\}$ of graphlets of size $k$ nodes, we define $f(g_{k_i}, G)$ as the relative frequency of any graphlet $g_{k_i} \in \mathcal{G}_k$ in the input graph $G$.

### B. Relationship to Graph Complement

The complement of a graph $G$, denoted by $\bar{G}$, is the graph defined on the same vertices as $G$ such that two vertices are connected in $\bar{G}$ if and only if they are not connected in $G$. Therefore, the graph sum $G + \bar{G}$ gives the complete graph on the set of vertices of $G$. There are direct relationships between the frequencies of graphlets and the frequencies of their complement. For each graphlet $g_{k_i}$, there exists a non-isomorphic complementary graphlet pattern $\bar{g}_{k_i}$, such that two vertices are connected in $\bar{g}_{k_i}$ if and only if they are not connected in $g_{k_i}$ [27]. For example, cliques and independent sets of size $k$ nodes are pairs of complementary graphlets. Similarly, chordal cycles of size 4 nodes are complementary to the 4-node-1edge graphlet (see Table I). It is also worth noting that the 4-path graphlet is a self-complementary pattern, which means the 4-path is isomorphic to itself. From this discussion, it is clear that the number of embeddings of each graphlet $g_{k_i} \in \mathcal{G}_k$ in the input graph $G$ is equivalent to the number of embeddings of its complementary graphlet $\bar{g}_{k_i}$ in the complement graph $\bar{G}$. In other words, $f(g_{k_i}, G) = f(\bar{g}_{k_i}, \bar{G})$ [27].

### C. Relationship to Graph/Matrix Reconstruction Theorems

The graph reconstruction conjecture [27], states that an undirected graph $G$ can be uniquely determined up to an isomorphism, from the set of all possible vertex-deleted subgraphs of $G$ (i.e., $\{G_v\}_{v \in V}$) [28]. Verification of this conjecture for all possible graphs up to 6 vertices was carried by Kelly [29], and later was extended to up to 11 vertices by McKay [28]. Clearly, if two graphs are isomorphic (i.e., $G \cong G'$), then their graphlet frequencies would be the same (i.e., $f_k(G) = f_k(G')$), but the reverse remains a conjecture for the general case of graphs. In contrast, the matrix reconstruction theorem has been resolved [30], which states that any $N \times N$ matrix can be reconstructed from its list of all possible principal minors obtained by the deletion of the $k$-th row and the $k$-th column [30], which is the foundation of a class of graph kernels called the *graphlet kernel* [3].

### D. Related Work

In this section, we briefly discuss some of the related work, highlighting various graph mining and machine learning tasks that would benefit from our approach. Much of the previous work focused on counting certain types of graphlets (e.g., only connected graphlets such as cliques and cycles) [24], [25], [31]. However, a number of graph mining and machine learning tasks rely on counting all graphlets of a certain size.

For example, some previous work used the full spectrum of graphlet frequencies to define a domain-independent coordinate system in which collections of graphs can be compactly represented and analyzed within a common space [32]. Moreover, a variety of graph kernels have been proposed in machine learning (e.g., graphlet, subtree, and random walk kernels) [3], [18], [33] to bridge the gap between graph learning and kernel methods. And some types of the graph kernels, in particular the graphlet kernel, rely on counting all graphlets. However, a general limitation of most graph kernels (including the graphlet kernel) is that they scale poorly to large graphs with more than few hundreds/thousands of nodes [18]. Thus, our fast algorithms would speedup the computations of these methods and their related applications in graph modeling, similarity, and comparisons.

Recently, there is an increased interest in sampling and other heuristic approaches for obtaining approximate counts of various graphlets [34], [35]. However, our approach focuses on exact graphlet counting and thus sampling methods are outside the scope of this paper. Nevertheless, the analysis and combinatorial arguments we show in this paper can be used along with efficient sampling methods to provide more accurate and efficient approximations.

In addition, the aim and scope of this paper is different from the aforementioned problem of graph reconstruction. While graph reconstruction tries to test for the notion of isomorphism and structure equivalence between graphs, our goal is to relax the notion of equivalence to some form of *structural similarity* between graphs, such that the graph similarity is measured using the feature representation of graphlets.

## III. Framework

In this section, we describe our approach for graphlet counting that takes only a fraction of the time to compute when compared with the current methods used. We introduce a number of combinatorial arguments that we show for different graphlets. The proposed graphlet counting algorithm leverages these combinatorial arguments to obtain significant improvement on the scalability of graphlet counting. For each edge, we count only a few graphlets, and with these counts along with the combinatorial arguments, we derive the exact counts of the others in constant time.

### A. Searching Edge Neighborhoods

Our proposed algorithm iterates over all the edges of the input graph $G = (V, E)$. For each edge $e = (u, v) \in E$, we define the *neighborhood* of an edge $e$, denoted by $\mathcal{N}(e)$, as the set of all nodes that are connected to the endpoints of $e$ — i.e., $\mathcal{N}(e) = \{\mathcal{N}(u) \backslash \{v\}\} \cup \{\mathcal{N}(v) \backslash \{u\}\}$, where $\mathcal{N}(u)$ and $\mathcal{N}(v)$ are the set of neighbors of $u$ and $v$ respectively. Given a single edge $e = (u, v) \in E$, we explore the subgraph surrounding this edge — i.e., the subgraph induced by both its endpoints and the nodes in its neighborhood. We call this subgraph the *egonet* of the edge $e$, where $e$ is the center (ego) of the subgraph.

We search for possible graphlet patterns of size $k = \{3, 4\}$ in the egonets of all edges in the graph. By searching egonets of edges, we first map the problem to the local (lower-dimensional) space induced by the neighborhood of each edge, and then merge the search results for all edges. Searching over a local low-dimensional space of edge neighborhoods is clearly more efficient than searching over the global high-dimensional space of the whole graph. Moreover, searching over a local low-dimensional space of edge neighborhoods is amenable to parallel implementation, which offers additional speedup over iterative methods. Note that exhaustive search of the egonet of any edge $e \in E$ yields at least $\mathcal{O}(\Delta^{k-1})$ asymptotically, where $\Delta$ is the maximum degree in $G$. Clearly, exhaustive search is computationally intensive for large graphs, and our approach is more efficient as we will show next.

### B. Counting Graphlets of Size $(k = 3)$ Nodes

Algorithm 1 (TriadCensus) shows how to count graphlets of size $k = 3$ for each edge. There are four possible graphlets of size $k = 3$ nodes, where only $g_{3_1}$ (i.e., triangle patterns) and $g_{3_2}$ (i.e., 2-star patterns) are connected graphlets (see Table I).

**Connected graphlets of size $k = 3$.** Lines 5—13 of Algorithm 1 show how to find and count triangles incident to an edge. For any edge $e = (u, v)$, a triangle $(u, v, w)$ exists, if and only if $w$ is connected to *both* $u$ and $v$. Let $\text{Tri}_e$ be the set of all nodes that form a triangle with $e = (u, v)$, and $|\text{Tri}_e|$ be the number of such triangles. Then, $\text{Tri}_e$ is the set of overlapping nodes in the neighborhoods of $u$ and $v$ — $\text{Tri}_e = \mathcal{N}(u) \cap \mathcal{N}(v)$. Note that Algorithm 1 counts each triangle three times (one time for each edge in the triangle), and therefore we divide the total count by 3 as in Equation (1),

$$f(g_{3_1}, G) = \frac{1}{3}. \sum_{e=(u,v)\in E} |\text{Tri}_e| \quad (1)$$

Now we need to count 2-star patterns (i.e., $g_{3_2}$). For any edge $e = (u, v)$, let $\text{Star}_e$ be the set of all nodes that form a 2-star with $e$, and $|\text{Star}_e|$ be the number of such star patterns. A 2-star pattern $(u, v, w)$ exists, if and only if $w$ is connected to *either* $u$ or $v$ but not both. Accordingly, $\text{Star}_e = \text{Star}_u \cup \text{Star}_v$, where $\text{Star}_u$ and $\text{Star}_v$ are the set of nodes that form a 2-star with $e$ centered at $u$ and $v$ respectively. More formally, $\text{Star}_u$ can be defined as $\text{Star}_u = \{w \in \mathcal{N}(u) \setminus \{v\} | w \notin \mathcal{N}(v)\}$, and $\text{Star}_v$ can be defined as $\text{Star}_v = \{w \in \mathcal{N}(v) \setminus \{u\} | w \notin \mathcal{N}(u)\}$.

Similar to counting triangles, Algorithm 1 counts each 2-star pattern two times (one time for each edge in the 2-star). Thus, we divide the sum for all edges by 2 as follows,

$$f(g_{3_2}, G) = \frac{1}{2}. \sum_{e=(u,v)\in E} |\text{Star}_u| + |\text{Star}_v| \quad (2)$$

**Disconnected graphlets of size $k = 3$.** There are two disconnected graphlets of size $k = 3$ nodes, $g_{3_3}$ (i.e., the 3-node-1-edge pattern) and $g_{3_4}$ (i.e., the independent set defined on 3 nodes) (see Table I). Lines 16 and 21 show how to count these patterns.

Equation (3) shows that the number of 3-node-1-edge graphlets per edge $e$ is equivalent to the number of all nodes that are not in the neighborhood subgraph (egonet) of edge $e$ (i.e., $V \setminus \{\mathcal{N}(u) \cup \mathcal{N}(v)\}$),

$$f(g_{3_3}, G) = \sum_{e=(u,v)\in E} |V| - |\mathcal{N}(u) \cup \mathcal{N}(v)| \quad (3)$$

where $|\mathcal{N}(u) \cup \mathcal{N}(v)| = |\text{Tri}_e| + |\text{Star}_\text{e}| + |\{u, v\}|$. Note that the number of 3-node-1-edge graphlets can be computed in $o(1)$ for each edge.

Given that the total number of graphlets of size 3 nodes is $\binom{N}{3}$, Equation (4) shows how to compute the frequency of $g_{3_4}$, which clearly can be done in $o(1)$,

$$f(g_{3_4}, G) = \binom{|V|}{3} - \big(f(g_{3_1}, G) + f(g_{3_2}, G) + f(g_{3_3}, G)\big) \quad (4)$$

The complexity of counting all graphlets of size $k = 3$ is $\mathcal{O}(|E|.\Delta)$ asymptotically (proof omitted for brevity).

**Algorithm 1** Our exact triad census algorithm for counting all 3-node graphlets. The algorithm takes an undirected graph as input and returns the frequencies of all 3-node graphlets $f(\mathcal{G}_3, G)$.

```
1:  procedure TRIADCENSUS($G = (V, E)$)
2:      Initialize Array $X$
3:      parallel for $e = (u, v) \in E$ do
4:          $\text{Star}_u = \emptyset, \text{Star}_v = \emptyset, \text{Tri}_e = \emptyset$
5:          for $w \in \mathcal{N}(u)$ do
6:              if $w = v$ then continue
7:              Add $w$ to $\text{Star}_u$ and set $X(w) = 1$
8:          for $w \in \mathcal{N}(v)$ do
9:              if $w = u$ then continue
10:             if $X(w) = 1$ then                          ▷ found triangle
11:                 Add $w$ to $\text{Tri}_e$
12:                 Remove $w$ from $\text{Star}_u$
13:             else Add $w$ to $\text{Star}_v$
14:         $f(g_{3_1}, G)$ += $|\text{Tri}_e|$
15:         $f(g_{3_2}, G)$ += $|\text{Star}_u| + |\text{Star}_v|$
16:         $f(g_{3_3}, G)$ += $|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|$
17:         for $w \in \mathcal{N}(u)$ do $X(w) = 0$
18:     end parallel
19:     $f(g_{3_1}, G) = 1/3.f(g_{3_1}, G)$
20:     $f(g_{3_2}, G) = 1/2.f(g_{3_2}, G)$
21:     $f(g_{3_4}, G) = \binom{|V|}{3} - f(g_{3_1}, G) - f(g_{3_2}, G) - f(g_{3_3}, G)$
22:     return $f(\mathcal{G}_3, G)$
```

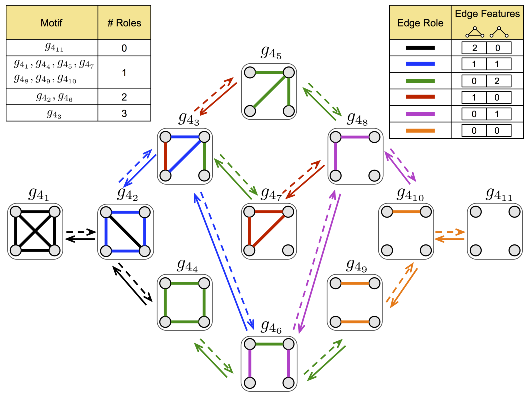
## IV. COUNTING GRAPHLETS OF SIZE $(k = 4)$ NODES

An exhaustive search of the egonet of any edge to count all 4-node graphlets independently yields $\mathcal{O}(\Delta^3)$ asymptotically, where $\Delta$ is the maximum degree in $G$. Clearly, exhaustive search is computationally intensive for large graphs. On the other hand, our approach is hierarchical and more efficient as we show next.

For each edge $e = (u, v)$, we start by finding triangles and 2-star patterns. Our central principle is that any 4-node graphlet $g_{4_i}$ can be decomposed into four 3-node graphlets [27], obtained by deleting one node from $g_{4_i}$ each time. Thus, we jointly count all possible 4-node graphlets by leveraging the knowledge obtained from finding 3-node graphlets and some combinatorial arguments that describe the relationships between pairs of graphlets. We summarize this procedure in the following steps:

1) For each edge $e$, find all neighborhood nodes forming triangle and 2-star patterns with $e$.
2) For each edge $e$, use the knowledge from step 1 to count only 4-cliques and 4-cycles.
3) For each edge $e$, use the knowledge from step 1 and some combinatorial arguments to compute unrestricted counts for all 4-node graphlets in constant time.
4) Merge the counts from all edges in the graph, and use combinatorial arguments involving unrestricted counts to obtain the counts of all other graphlets.

Note that we refer to the unrestricted counts as the counts that can be computed in constant time and using only the knowledge obtained from step 1. Next, we discuss the details of our approach. We start by discussing the graphlet transition diagram to show the pairwise relationships between different 4-node graphlets. Then, we discuss a general principle for counting 4-node graphlets, which leverages the graphlet transition



| Motif | # Roles |
|---|---|
| $g_{4_{11}}$ | 0 |
| $g_{4_1}, g_{4_4}, g_{4_5}, g_{4_7}$ $g_{4_8}, g_{4_9}, g_{4_{10}}$ | 1 |
| $g_{4_2}, g_{4_6}$ | 2 |
| $g_{4_3}$ | 3 |

| Edge Role | Edge Features | |
|---|---|---|
| ■ (black) | 2 | 0 |
| ■ (blue) | 1 | 1 |
| ■ (green) | 0 | 2 |
| ■ (red) | 1 | 0 |
| ■ (purple) | 0 | 1 |
| ■ (orange) | 0 | 0 |

Fig. 1. **4–node graphlet transition diagram:** Figure shows all possible $\pm 1$ *edge* transitions between the set of all 4-node graphlets. Dashed right arrows denote the deletion of one edge to transition from one graphlet to another. Solid left arrows denote the addition of one edge to transition from one graphlet to another. Edges are colored by their feature-based roles, where the set of feature are defined by the number of triangles and 2-stars incident to an edge (see Table in the top-right corner). We define six different classes of edge roles colored from black to orange (see Table in the top-right corner). Dashed/solid arrows are colored similar to the edge roles to denote which edge would be deleted/added to transition from one graphlet to another. The table in the top-left corner shows the number of edge roles per each graphlet.

diagram and some combinatorial arguments to improve the performance of graphlet counting.

## A. Graphlet Transition Diagram

Assume that each graphlet is a state, Fig. 1 shows all possible $\pm 1$ *edge* transitions between the states of all 4-node graphlets. We can transition from one graphlet to another by the deletion (denoted by dashed right arrows) or addition (denoted by solid left arrows) of a single edge. We define six different classes of possible edge roles denoted by the colors from black to orange (see Table in the top-right corner in Fig. 1). An *edge role* is an edge-level connectivity pattern (e.g., a chord edge), where two edges belong to the same role (i.e., class) if they are similar in their topological features. For each edge, we define a topological feature vector that consists of the number of triangles and 2-stars incident to this edge. Then, we classify edges to one of the six roles based on their feature vectors. Thus, all edges that appear in 4-node graphlets are colored by their roles. In addition, the transition arrows are colored similar to the edge roles to denote which edge type should be deleted/added to transition from one graphlet to another. Note that a single edge deletion/addition changes the role (class) of other edges in the graphlet. The table in the top-left corner of Fig. 1 shows the number of edge roles per each graphlet.

For example, consider the 4-clique graphlet ($g_{4_1}$), where each edge participates exactly in two triangles. Therefore, all the edges in a 4-clique graphlet ($g_{4_1}$) belong to the first role (denoted by the black color). Similarly, consider the 4-chordalcycle ($g_{4_2}$), where each edge (except the chord edge) participates exactly in one triangle and one 2-star. Therefore, all edges in a 4-chordalcycle "$g_{4_2}$" belong to the second role (denoted by the blue color) except for the chord edge which belongs to the first role (denoted by the black color). Fig. 1 shows how to transition from the 4-clique to the 4-chordalcycle "$g_{4_2}$" by deleting one (any) edge from the 4-clique.

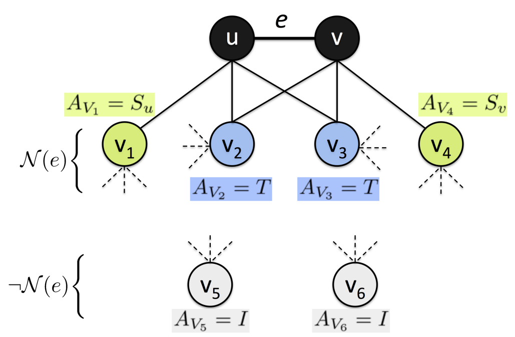


Fig. 2. Let $T$ denote the nodes forming triangles with edge $(u, v)$ (i.e., $V_2, V_3$), whereas $S_u$ and $S_v$ denote the nodes forming 2-stars centered at $u$ and $v$ respectively (i.e., $V_1, V_4$), and let $I$ denote the nodes that are not connected to edge $e$ (i.e., $V_5, V_6$). Further, the dotted lines represent edges incident to these nodes.

## B. General Principle for Counting Graphlets of size $k = 4$

Generally speaking, suppose we have $N^{(e)}$ distinct 4-node subgraphs that contains an edge $e = (u, v)$,

$$N^{(e)} = \Big|\{\{u, v, w, r\} \mid w, r \in V \setminus \{u, v\} \wedge w \neq r\}\Big| \tag{5}$$

Each subgraph $\{u, v, w, r\}$ in this collection may satisfy one or two properties $a_i, a_j \in A = \{T, S_u, S_v, I\}$. These properties describe the topological properties of nodes $w$ and $r$ with respect to edge $e$, such that $A_w = a_i$ if $\{u, v, w\}$ forms subgraph pattern $a_i$, and $A_r = a_j$ if $\{u, v, r\}$ forms subgraph pattern $a_j$. For example, $A_w = T$ if $w$ forms a triangle with $e$, and $A_w = S_u$ or $S_v$ if $w$ forms a 2-star with $e$ centered around $u$ or $v$ respectively. Also, $A_w = I$ if $w$ is independent (disconnected) from $e$. We clarify these properties by example in Fig. 2.

Let $N^{(e)}_{a_i,a_j}$ denote the number having properties $a_i, a_j \in A$,

$$N^{(e)}_{a_i,a_j} = \left|\left\{\{u, v, w, r\} \,\middle|\, \begin{smallmatrix} w,r \in V\setminus\{u,v\} \\ \wedge w \neq r \\ \wedge A_w = a_i, A_r = a_j \end{smallmatrix}\right\}\right| \tag{6}$$

Now that we defined the topological properties of nodes $w$ and $r$ relative to edge $e$, we need to define whether nodes $w$ and $r$ are connected themselves. Let $e'_{wr}$ represent whether $w$ and $r$ are connected or not, such that $e'_{wr} = 1$ if $(w, r) \in E$ and $e'_{wr} = 0$ otherwise. Accordingly, let $N^{(e)}_{a_i,a_j,e'_{wr}}$ denotes the number of 4-node graphlets $\{u, v, w, r\}$, where $w, r$ satisfy property $a_i, a_j \in A$ and $e'_{wr} \in \{0, 1\}$,

$$N^{(e)}_{a_i,a_j,e'_{wr}} = \left|\left\{\{u, v, w, r\} \,\middle|\, \begin{smallmatrix} w,r \in V\setminus\{u,v\} \\ \wedge w \neq r \\ \wedge A_w = a_i, A_r = a_j \\ \wedge e'_{wr} \in \{0,1\} \end{smallmatrix}\right\}\right| \tag{7}$$

For example, $N^{(e)}_{T,T,1}$ is the number of all graphlets $\{u, v, w, r\}$ containing edge $e$, where both $w$ and $r$ are forming triangles with $e$ and there exist an edge between $w$ and $r$. Using Equations (6) and (7), we provide a general principle for graphlet counting in the following theorem.

**Theorem 1.** *General Principle for Graphlet Counting: Given a graph $G$, for any edge $e = (u, v)$ in $G$, and for any properties*

$a_i, a_j \in A$, *the number of* 4*-node graphlets* $\{u, v, w, r\}$ *satisfies the following rule,*

$$N^{(e)}_{a_i,a_j,0} = N^{(e)}_{a_i,a_j} - N^{(e)}_{a_i,a_j,1} \tag{8}$$

*Proof:* Suppose there is a subgraph $\{u, v, w, r\}$ containing edge $e$, where nodes $w$ and $r$ satisfy $a_i, a_j$ properties respectively, and $(w, r) \in E$. Then the expression on the right side counts this subgraph once in the $N^{(e)}_{a_i,a_j}$ term, and once in the $N^{(e)}_{a_i,a_j,1}$. By the principle of inclusion-exclusion [36], the total contribution of the subgraph $\{u, v, w, r\}$ in $N^{(e)}_{a_i,a_j,0}$ is zero. Thus, $N^{(e)}_{a_i,a_j,0}$ is the number of graphlets having properties $a_i, a_j$, but $(w, r) \notin E$. ■

Clearly, it is sufficient to compute $N^{(e)}_{a_i,a_j}$ and $N^{(e)}_{a_i,a_j,1}$ only, and use Theorem 1 to compute $N^{(e)}_{a_i,a_j,0}$ in constant time. Note that $N^{(e)}_{a_i,a_j}$ is an unrestricted count and can be computed in constant time using the knowledge we have from finding 3-node graphlets.

To simplify the discussion in the following sections, we precisely show how to compute $N^{(e)}_{a_i,a_j}$, the number of 4-node graphlets $\{u, v, w, r\}$ such that $w, r$ satisfy property $a_i, a_j \in A$ respectively. Let $\mathcal{W}_{a_i}$ be the set of nodes with property $a_i \in A$ (i.e., $\mathcal{W}_{a_i} = \{w \in V \setminus \{u, v\} \mid A_w = a_i, \forall a_i \in A\}$), and similarly $\mathcal{R}_{a_j}$ be the set of nodes with property $a_j \in A$ (i.e., $\mathcal{R}_{a_j} = \{r \in V \setminus \{u, v\} \mid A_r = a_j, \forall a_j \in A\}$). If $a_i = a_j$, then $\mathcal{W}_{a_i} = \mathcal{R}_{a_j}$. Thus,

$$N^{(e)}_{a_i,a_i} = \binom{|\mathcal{W}_{a_i}|}{2} = \frac{1}{2}.(|\mathcal{W}_{a_i}| - 1).|\mathcal{W}_{a_i}| \tag{9}$$

However, if $a_i \neq a_j$, then $\mathcal{W}_{a_i}$ and $\mathcal{R}_{a_j}$ are mutually exclusive (i.e., $\mathcal{W}_{a_i} \cap \mathcal{R}_{a_j} = \emptyset$).

Thus, we get the following,

$$N^{(e)}_{a_i,a_j} = |\mathcal{W}_{a_i}|.|\mathcal{R}_{a_j}| \tag{10}$$

### C. Analysis & Combinatorial Arguments

In this part, we discuss combinatorial arguments involving unrestricted counts that can be computed computed directly from our knowledge of 3-node graphlets. These combinatorial arguments capture the relationships between the counts of pairs of 4-node graphlets. The proofs of these relationships are based on Theorem 1 and the transition diagram in Fig. 1. For each pair of graphlets $g_{4_i}$ and $g_{4_j}$, we show the relationship for each edge in the graph (in Corollary 1–14), then we show a generalization for the whole graph (in Lemma 1–7). We only show some of the proofs for brevity, all other related materials are available online.

*Relationship between* 4*-Cliques &* 4*-ChordalCycles*

**Corollary 1.** *For any edge* $e = (u, v)$ *in the graph, the number of* 4*-cliques containing* $e$ *is* $N^{(e)}_{T,T,1}$.

**Corollary 2.** *For any edge* $e = (u, v)$ *in the graph, the number of* 4*-chordalcycles, where* $e$ *is the chord edge of the cycle (denoted by the black color in Fig. 1), is* $N^{(e)}_{T,T,0}$.

**Lemma 1.** *For any graph* $G$*, the relationship between the counts of* 4*-cliques (i.e.,* $f(g_{4_1}, G)$*) and* 4*-chordalcycles (i.e.,* $f(g_{4_2}, G)$*) is,*

$$f(g_{4_2}, G) = \sum_{e \in E} \binom{|\text{Tri}_e|}{2} - 6.f(g_{4_1}, G)$$

*Proof:* From Theorem 1 and the addition principle [36], the total count for all edges in $G$ is,

$$\sum_{e \in E} N^{(e)}_{T,T,0} = \sum_{e \in E} N^{(e)}_{T,T} - \sum_{e \in E} N^{(e)}_{T,T,1} \tag{11}$$

Given that $N^{(e)}_{T,T}$ is the number of 4-node subgraphs $\{u, v, w, r\}$ containing $e$, such that $A_w = T, A_r = T$. Thus, from Eq. (9), $N^{(e)}_{T,T} = \binom{|\text{Tri}_e|}{2}$. From Corollary 1, each 4-clique will be counted 6 times (once for each edge in the clique). Thus, the total count of 4-cliques in $G$ is $f(g_{4_1}, G) = \frac{1}{6}. \sum_{e \in E} N^{(e)}_{T,T,1}$. Similarly, from Corollary 2, each 4-chordalcycle is counted only once for each chord edge. Thus, the total count of 4-chordalcycles in $G$ is $f(g_{4_2}, G) = \sum_{e \in E} N^{(e)}_{T,T,0}$. By direct substitution in Eq. (11), this lemma is true. ■

*Relationship between* 4*-Cycles &* 4*-Paths*

**Corollary 3.** *For any edge* $e = (u, v)$ *in the graph, the number of* 4*-cycles containing* $e$ *is* $N^{(e)}_{S_u,S_v,1}$.

**Corollary 4.** *For any edge* $e = (u, v)$ *in the graph, the number of* 4*-paths containing* $e$*, where* $e$ *is the middle edge in the path (denoted by the green color in Fig. 1), is* $N^{(e)}_{S_u,S_v,0}$.

**Lemma 2.** *For any graph* $G$*, the relationship between the counts of* 4*-cycles (i.e.,* $f(g_{4_4}, G)$*) and* 4*-paths (i.e.,* $f(g_{4_6}, G)$*) is,*

$$f(g_{4_6}, G) = \sum_{e \in E} |\text{Star}_u|.|\text{Star}_v| - 4.f(g_{4_4}, G)$$

*Proof:* From Theorem 1 and the addition principle [36], the total count for all edges in $G$ is,

$$\sum_{e \in E} N^{(e)}_{S_u,S_v,0} = \sum_{e \in E} N^{(e)}_{S_u,S_v} - \sum_{e \in E} N^{(e)}_{S_u,S_v,1} \tag{12}$$

Given that $N^{(e)}_{S_u,S_v}$ is the number of 4-node subgraphs $\{u, v, w, r\}$ containing $e$, such that $w, r$ $A_w = S_u, A_r = S_v$. Thus, from Eq. (10), $N^{(e)}_{S_u,S_v} = |\text{Star}_u|.|\text{Star}_v|$. From Corollary 3, each 4-cycle will be counted 4 times (once for each edge in the cycle). Thus, the total count of 4-cycles in $G$ is $f(g_{4_4}, G) = \frac{1}{4}. \sum_{e \in E} N^{(e)}_{S_u,S_v,1}$. Similarly, from Corollary 4, each 4-path is counted only once for each middle edge in the path. Thus, the total count of 4-paths in $G$ is $f(g_{4_6}, G) = \sum_{e \in E} N^{(e)}_{S_u,S_v,0}$. By direct substitution in Eq. (12), this lemma is true. ■

*Relationship between* 4*-TailedTriangles &* 4*-ChordalCycles*

**Corollary 5.** *For any edge* $e = (u, v)$ *in the graph, the number of* 4*-tailedtriangles where* $e$ *is part of both the triangle* *and 2-star patterns (denoted by the blue color in Fig. 1), is* $N^{(e)}_{T,S_u \vee S_v,0}$.

**Corollary 6.** *For any edge $e = (u, v)$ in the graph, the number of 4-chordalcycles where $e$ is a cycle edge (denoted by the blue color in Fig. 1), is* $N^{(e)}_{T,S_u \vee S_v,1}$.

**Lemma 3.** *For any graph $G$, the relationship between the counts of 4-chordalcycles (i.e., $f(g_{4_2}, G)$) and 4-tailedtriangles (i.e., $f(g_{4_3}, G)$) is,*

$$2.f(g_{4_3}, G) = \sum_{e \in E} |\text{Tri}_e|.(|\text{Star}_u| + |\text{Star}_v|) - 4.f(g_{4_2}, G)$$

*Relationship between 4-TailedTriangles & 3-Stars*

**Corollary 7.** *For any edge $e = (u, v)$ in the graph, the number of 4-tailedtriangles with $e$ as the tail edge (denoted by the green color in Fig. 1) and $u$ is part of the triangle, is* $N^{(e)}_{S_u,S_u,1}$.

In a similar fashion, the number of 4-tailedtriangles with $e$ as the tail edge and $v$ is part of the triangle is $N^{(e)}_{S_v,S_v,1}$. Thus, the total number of 4-tailedtriangles with $e$ as the tail edge and $u \vee v$ is part of the triangle is $N^{(e)}_{S_.,S_.,1} = N^{(e)}_{S_u,S_u,1} + N^{(e)}_{S_v,S_v,1}$.

**Corollary 8.** *For any edge $e = (u, v)$ in the graph, the number of 3-star centered around $u$ is* $N^{(e)}_{S_u,S_u,0}$.

Again, the number of 3-stars centered around $v$ is $N^{(e)}_{S_v,S_v,0}$. Thus, the total number of 3-stars centered around $u$ or $v$ is $N^{(e)}_{S_.,S_.,0} = N^{(e)}_{S_u,S_u,0} + N^{(e)}_{S_v,S_v,0}$.

**Lemma 4.** *For any graph $G$, the relationship between the counts of 3-stars (i.e., $f(g_{4_5}, G)$) and 4-tailedtriangles (i.e., $f(g_{4_3}, G)$) is,*

$$3.f(g_{4_5}, G) = \sum_{e \in E} \binom{|\text{Star}_u|}{2} + \binom{|\text{Star}_v|}{2} - f(g_{4_3}, G)$$

*Relationship between 4-TailedTriangles & 4-Node-1-Triangles*

**Corollary 9.** *For any edge $e = (u, v)$ in the graph, the number of 4-node-1-triangle is* $N^{(e)}_{T,I,0}$.

**Corollary 10.** *For any edge $e = (u, v)$ in the graph, the number of 4-tailedtriangles with $e$ participating in the triangle but not connected to the tail edge (denoted by the red color in Fig. 1), is* $N^{(e)}_{T,I,1}$.

**Lemma 5.** *For any graph $G$, the relationship between the counts of 4-tailedtriangles (i.e., $f(g_{4_3}, G)$) and 4-node-1-triangles (i.e., $f(g_{4_7}, G)$) is,*

$$3.f(g_{4_7}, G) = \sum_{e \in E} \Big(\text{Tri}_e.\,(|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|)\Big) - f(g_{4_3}, G)$$

*Relationship between 4-Paths & 4-node-2-Stars*

**Corollary 11.** *For any edge $e = (u, v)$ in the graph, the number of 4-paths where $e$ is the start or end of the path (denoted by the purple color in Fig. 1), is* $N^{(e)}_{S_u \vee S_v,I,1}$.

**Corollary 12.** *For any edge $e = (u, v)$ in the graph, the number of 4-node-2-stars where $e$ is one of the star edges (denoted by the purple color in Fig. 1), is* $N^{(e)}_{S_u \vee S_v,I,0}$.

**Lemma 6.** *For any graph $G$, the relationship between the counts of 4-paths (i.e., $f(g_{4_6}, G)$) and 4-node-2-stars (i.e., $f(g_{4_8}, G)$) is,*

$$2.f(g_{4_8}, G) = \sum_{e \in E} |\text{Star}_e|.(|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|) - 2.f(g_{4_6}, G)$$

*Relationship between 4-node-2-edges & 4-node-1-edge*

**Corollary 13.** *For any edge $e = (u, v)$ in the graph, the number of 4-node-2-edges where $e$ is any of the two independent edges in the graphlet (denoted by the orange color in Fig. 1), is* $N^{(e)}_{I,I,1}$.

**Corollary 14.** *For any edge $e = (u, v)$ in the graph, the number of 4-node-1-edge where $e$ is an isolated/single edge in the graphlet (denoted by the orange color in Fig. 1), is* $N^{(e)}_{I,I,0}$.

**Lemma 7.** *For any graph $G$, the relationship between the counts of 4-node-2-edge graphlets (i.e., $f(g_{4_9}, G)$) and 4-node-1-edge graphlets (i.e., $f(g_{4_{10}}, G)$) is,*

$$f(g_{4_{10}}, G) = \sum_{e \in E} \binom{|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|}{2} - 2.f(g_{4_9}, G)$$

While it is straightforward to compute $N^{(e)}_{I,I}$ for each edge $e$, this is not the case for $N^{(e)}_{I,I,1}$ or $N^{(e)}_{I,I,0}$, as they require searching outside the local edge neighborhood. However, since $N^{(e)}_{I,I,1}$ is the number of edges outside the egonet of $e$, it can be computed as,

$$\begin{aligned} N^{(e)}_{I,I,1} = {} & |E| - |\mathcal{N}(u) \setminus \{v\}| - |\mathcal{N}(v) \setminus \{u\}| - |\{e\}| \\ & - [N^{(e)}_{T,T,1} + N^{(e)}_{T,S_u \vee S_v,1} + N^{(e)}_{T,I,1}] \\ & - [N^{(e)}_{S_.,S_.,1} + N^{(e)}_{S_u,S_v,1} + N^{(e)}_{S_.,I,1}] \end{aligned}$$

Thus, the total number of 4-node-2-edges is,

$$\begin{aligned} 2.f(g_{4_9}, G) &= \sum_{e \in E} N^{(e)}_{I,I,1} \\ &= \sum_{e \in E} |E| - |\mathcal{N}(u) \setminus \{v\}| - |\mathcal{N}(v) \setminus \{u\}| - |\{e\}| \\ &\quad - [6.f(g_{4_1}, G) + 4.f(g_{4_2}, G) + 2.f(g_{4_3}, G)] \\ &\quad - [4.f(g_{4_4}, G) + 2.f(g_{4_6}, G)] \end{aligned} \tag{13}$$

Finally, the number of 4-node-independent graphlets ($g_{4_{11}}$) is,

$$f(g_{4_{11}}, G) = \binom{|V|}{4} - \sum_{i=1}^{10} f(g_{4_i}, G) \tag{14}$$

## D. Algorithm

Algorithm 2 shows how to count all graphlets of size $k = \{3, 4\}$ nodes efficiently (using Lemma 1— 7). As discussed previously, we start by finding all triangle and 2-star patterns in Lines 7–15 (i.e., step 1). Then, in Lines 18—19 we only count 4-cliques and 4-cycles (i.e., step 2). Then, Lines 21—32 compute unrestricted counts for all 4-node graphlets in constant time (using knowledge from steps 1 and 2, as discussed in step 3), and finally Lines 35—37 compute the final counts (using the lemma proved in Section IV-C) (i.e., step 4). Our approach counts all 4-cliques and 4-cycles in $\mathcal{O}(m.\Delta.T_{max})$ and $\mathcal{O}(m.\Delta.S_{max})$ respectively, where $T_{max}$ is the maximum number of triangles incident to an edge and $T_{max} \ll \Delta$ for sparse graphs, and $S_{max}$ is the maximum number of stars incident to an edge and $S_{max} \leq \Delta$. This is more efficient than $\mathcal{O}(|V|.\Delta^3)$ given by [3], and $\mathcal{O}(\Delta.|E| + |E|^2)$ given by [23].

## V. Experiments & Applications

We proceed by first demonstrating how fast our algorithm (Algorithm 2) counts all graphlets of size $k = \{3, 4\}$ (both connected and disconnected graphlets) on various networks. We make all our implementations, further experiments, and proofs available in an online appendix[1]. In this paper, we show detailed results for 55 networks categorized in 8 broad classes from social, facebook [37], biological, web, technological, co-authorship, infrastructure, among other domains (see the links[2] for data download). And, in the online appendix, we present a more extensive collection of 300+ networks, including both large sparse networks as well as dense networks from the DIMACs challenge[3]. Note that for all of the networks, we discard edge weights, self-loops, and edge direction. To the best of our knowledge, this is the largest study for graphlet counting, and these are the largest graphlet computations published to date. Our own implementation of Algorithm. 2 uses shared memory, but the algorithm is well-suited for other architectures. We used a two processor, Intel Xeon system with 6 cores and 48GB of memory.

### A. *Scalability & Runtime*

Table II describes the properties of the 55 networks considered here. It also shows the counts of graphlets of size $k = \{3, 4\}$ and states the time (seconds) taken to count all graphlets. We only show counts of connected graphlets due to space limitations, however all counts are available in the online appendix. Notably, Algorithm 2 takes only few seconds to count all graphlets for large social, web, and technological graphs (among others). For example, for a large road network (i.e., inf-road-usa) with 24M nodes and 29M edges, Algorithm 2 takes only 4 seconds to count all graphlets. Also as shown in Table II, for large facebook networks with nearly 2M edges, Algorithm 2 takes only 15 seconds, and for large web graphs with nearly 8M edges, Algorithm 2 takes only 25 seconds.

We compare the empirical runtime of Algorithm 2 to the state-of-the-art baseline method RAGE [23]. For social and facebook networks, we observed that Algorithm 2 is on average 460x faster than RAGE. For all other networks, we observed that Algorithm 2 is on average 600x faster than RAGE. Notably, Algorithm 2 takes only 7 seconds to count graphlets of facebook networks with 1.3M edges, while RAGE takes almost an hour for the same networks. For larger networks with millions of nodes/edges, RAGE was timed out (as it did not finish within 30 hours of runtime). Moreover, for dense graphs from the DIMACS challenge, RAGE takes almost 17 minutes, while Algorithm 2 takes less than a second. We

[1]http://nesreenahmed.com/graphlets

[2]http://networkrepository.com/

[3]http://dimacs.rutgers.edu/Challenges/

**Algorithm 2** Our exact graphlet census algorithm for counting all 3, 4-node graphlets. The algorithm takes an undirected graph as input and returns the frequencies of all 3, 4-node graphlets

```
 1: procedure GraphletCounting($G = (V, E)$)
 2:   Initialize Array $X$
 3:   $N_{T,T} = 0$, $N_{S_u,S_v} = 0$, $N_{T,S_u \vee S_v} = 0$, $N_{S.,S.} = 0$
 4:   $N_{T,I} = 0$, $N_{S_u \vee S_v,I} = 0$, $N_{I,I} = 0$, $N_{I,I,1} = 0$
 5:   parallel for $e = (u, v) \in E$ do
 6:     $\mathrm{Star}_u = \emptyset, \mathrm{Star}_v = \emptyset, \mathrm{Tri}_e = \emptyset$
 7:     for $w \in \mathcal{N}(u)$ do
 8:       if $w = v$ then continue
 9:       Add $w$ to $\mathrm{Star}_u$ and set $X(w) = 1$
10:     for $w \in \mathcal{N}(v)$ do
11:       if $w = u$ then continue
12:       if $X(w) = 1$ then                          ▷ found triangle
13:         Add $w$ to $\mathrm{Tri}_e$ and set $X(w) = 2$
14:         Remove $w$ from $\mathrm{Star}_u$
15:       else Add $w$ to $\mathrm{Star}_v$ and set $X(w) = 3$
16:     Compute $f(\mathcal{G}_3, G)$ as in Lines 14—16 of Alg. 1
17:     // Get Counts of 4-Cliques & 4-Cycles
18:     $f(g_{4_1}, G)$ += CliqueCount($X, \mathrm{Tri}_e$)
19:     $f(g_{4_4}, G)$ += CycleCount($X, \mathrm{Star}_u$)
20:     // Get Unrestricted Counts for 4-Node Connected Graphlets
21:     $N_{T,T} \mathrel{+}= \binom{|\mathrm{Tri}_e|}{2}$
22:     $N_{S_u,S_v} \mathrel{+}= |\mathrm{Star}_u|.|\mathrm{Star}_v|$
23:     $N_{T,S_u \vee S_v} \mathrel{+}= |\mathrm{Tri}_e|.(|\mathrm{Star}_u| + |\mathrm{Star}_v|)$
24:     $N_{S_u,S_u} = \binom{|\mathrm{Star}_u|}{2}$ and $N_{S_v,S_v} = \binom{|\mathrm{Star}_v|}{2}$
25:     $N_{S.,S.} \mathrel{+}= N_{S_u,S_u} + N_{S_v,S_v}$
26:     // Get Unrestricted Counts for 4-Node Disconnected Graphlets
27:     $N_{T,I} \mathrel{+}= \mathrm{Tri}_e.(|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|)$
28:     $N_{S_u,I} = |\mathrm{Star}_u|.(|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|)$
29:     $N_{S_v,I} = |\mathrm{Star}_v|.(|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|)$
30:     $N_{S_u \vee S_v,I} \mathrel{+}= N_{S_u,I} + N_{S_v,I}$
31:     $N_{I,I} \mathrel{+}= \binom{|V| - |\mathcal{N}(u) \cup \mathcal{N}(v)|}{2}$
32:     $N_{I,I,1} \mathrel{+}= |E| - |\mathcal{N}(u) \setminus \{v\}| - |\mathcal{N}(v) \setminus \{u\}| - 1$
33:     for $w \in \mathcal{N}(v)$ do $X(w) = 0$
34:   end parallel
35:   Use Lemma 1—5 to compute $f(g_{4_i}, G)$ for $i = 1:8$
36:   Use Eq. (13) to compute $f(g_{4_9}, G)$ and Lemma 7 for $f(g_{4_{10}}, G)$
37:   Use Eq. (14) to compute $f(g_{4_{11}}, G)$
38:   return $f(\mathcal{G}_3, G), f(\mathcal{G}_4, G)$
39: procedure CliqueCount($X, \mathrm{Tri}_e$)
40:   $\mathrm{cliq}_e = 0$
41:   for each node $w \in \mathrm{Tri}_e$ do
42:     for $r \in \mathcal{N}(w)$ do
43:       if $X(r) = 2$ then $\mathrm{cliq}_e \mathrel{+}= 1$            ▷ found 4-Clique
44:     $X(w) = 0$
45:   return $\mathrm{cliq}_e$
46: procedure CycleCount($X, \mathrm{Star}_u$)
47:   $\mathrm{cyc}_e = 0$
48:   for each node $w \in \mathrm{Star}_u$ do
49:     for $r \in \mathcal{N}(w)$ do
50:       if $X(r) = 3$ then $\mathrm{cyc}_e \mathrel{+}= 1$             ▷ found 4-Cycle
51:     $X(w) = 0$
52:   return $\mathrm{cyc}_e$
```

also compared to the baseline method FANMOD [24] and Orca [25], we found that for a facebook network with 250k edges, FANMOD takes roughly 2.5 hours for counting all graphlets, RAGE takes almost 7 minutes for the same network, and Orca takes almost 10 seconds, while Algorithm 2 takes less than a second. We omit the results of FANMOD and

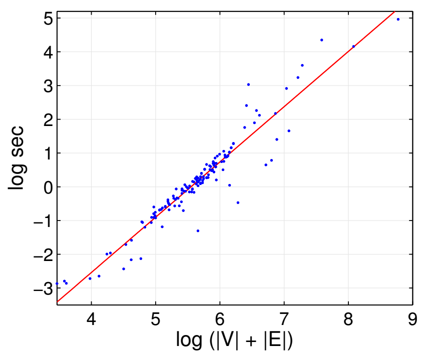


Fig. 3. The empirical runtime of our exact graphlet counting (Alg.2) in social and information networks scales almost linearly with the network dimension.

Orca for brevity. Note that both RAGE and Orca count only connected graphlets, while our algorithm and FANMOD count both connected and disconnected graphlets. In Figure 3, we plot the runtime of Algorithm 2 for a representative subset of 150 social and information networks. The figure shows that our algorithm exhibits nearly linear-time scaling over networks ranging from 1K to 100M nodes.

TABLE II. RUNTIME & STATISTICS FOR A SUBSET OF 55 NETWORKS

| | | | | | | | | | | | Seconds | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **graph** | $\lvert V\rvert$ | $\lvert E\rvert$ | $\lvert g_{3_1}\rvert$ | $\lvert g_{3_2}\rvert$ | $\lvert g_{4_1}\rvert$ | $\lvert g_{4_2}\rvert$ | $\lvert g_{4_4}\rvert$ | $\lvert g_{4_6}\rvert$ | $\lvert g_{4_5}\rvert$ | $\lvert g_{4_3}\rvert$ | Alg.2 | RAGE |
| soc-brightkite | 57k | 213k | 494k | 12M | 2.9M | 12M | 2.7M | 533M | 1.3B | 114M | 0.2 | 273.03 |
| socfb-Berkeley13 | 23k | 852k | 5.4M | 125M | 27M | 153M | 87M | 17B | 25B | 2.7B | 4.94 | 2514.59 |
| socfb-Wisconsin87 | 24k | 836k | 4.9M | 107M | 23M | 121M | 59M | 12B | 21B | 1.9B | 3.93 | 1450.31 |
| socfb-FSU53 | 28k | 1.0M | 7.9M | 130M | 63M | 242M | 95M | 16B | 10B | 2.9B | 5.55 | 2192.94 |
| socfb-MSU24 | 32k | 1.1M | 6.5M | 139M | 33M | 183M | 106M | 16B | 32B | 2.6B | 5.67 | 1904.09 |
| socfb-Texas80 | 32k | 1.2M | 9.6M | 160M | 68M | 316M | 122M | 21B | 11B | 3.9B | 7.53 | 2967.01 |
| socfb-Michigan23 | 30k | 1.2M | 8.3M | 162M | 49M | 277M | 146M | 23B | 13B | 3.5B | 7.57 | 2995.83 |
| socfb-Indiana69 | 30k | 1.3M | 9.4M | 181M | 60M | 269M | 141M | 25B | 13B | 3.8B | 8.44 | 3212.10 |
| socfb-UIllinois20 | 31k | 1.3M | 9.4M | 172M | 64M | 273M | 130M | 23B | 27B | 3.8B | 7.88 | 3088.77 |
| socfb-UF21 | 35k | 1.5M | 12M | 266M | 98M | 433M | 186M | 40B | 150B | 7.2B | 14.49 | N/A |
| soc-flickr | 514k | 3.2M | 59M | 963M | 1.7B | 14B | 6.7B | 244B | 326B | 90B | 182.57 | N/A |
| soc-orkut | 3.1M | 117M | 628M | 44B | 3.2B | 48B | 70B | 19T | 98T | 1.5T | 14448.6 | N/A |
| bio-celegans | 453 | 2.0k | 3.3k | 69k | 3.0k | 37k | 4.5k | 495k | 2.9M | 363k | <0.001 | 1.7 |
| bio-diseasome | 516 | 1.2k | 1.4k | 5.4k | 1.4k | 923 | 42 | 18k | 27k | 19k | <0.001 | 0.44 |
| bio-dmela | 7.4k | 26k | 2.9k | 572k | 393 | 13k | 107k | 11M | 9.2M | 312k | 0.01 | 2.47 |
| bio-yeast-protein-inter | 1.8k | 2.2k | 222 | 11k | 41 | 198 | 140 | 31k | 72k | 2.6k | <0.001 | 0.53 |
| bio-yeast | 1.5k | 1.9k | 206 | 11k | 39 | 195 | 139 | 31k | 72k | 2.5k | <0.001 | 0.43 |
| bio-human-gene2 | 14k | 9.0M | 4.9B | 10B | 2.3T | 3.7T | 90B | 4.4T | 5.3T | 8.4T | 8023.84 | N/A |
| bio-mouse-gene | 43k | 14M | 3.6B | 15B | 670B | 2.1T | 223B | 9.0T | 6.7T | 7.7T | 5515.6 | N/A |
| ca-CSphd | 1.9k | 1.7k | 8 | 6.6k | 0 | 5 | 8 | 9.4k | 32k | 93 | <0.001 | 1.25 |
| ca-GrQc | 4.2k | 13k | 48k | 85k | 329k | 66k | 1.1k | 553k | 406k | 628k | <0.001 | 5.99 |
| ca-dblp-2012 | 317k | 1.0M | 2.2M | 15M | 17M | 4.8M | 203k | 252M | 259M | 97M | 0.48 | 227.79 |
| ca-cit-HepTh | 23k | 2.4M | 191M | 1.6B | 13B | 47B | 7.3B | 538B | 976B | 385B | 132.66 | N/A |
| ca-cit-HepPh | 28k | 3.1M | 196M | 1.5B | 9.8B | 34B | 6.1B | 536B | 479B | 276B | 125.49 | N/A |
| ca-coauthors-dblp | 540k | 15M | 444M | 698M | 15B | 3.4B | 31M | 42B | 27B | 67B | 40.26 | N/A |
| ca-hollywood-2009 | 1.1M | 56M | 4.9B | 33B | 1.4T | 635B | 168B | 21T | 17T | 8.9T | 13799.6 | N/A |
| tech-as-caida2007 | 26k | 53k | 36k | 15M | 54k | 1.7M | 407k | 285M | 7.8B | 47M | 0.19 | 36.83 |
| tech-p2p-gnutella | 63k | 148k | 2.0k | 1.6M | 16 | 826 | 42k | 15M | 8.1M | 71k | 0.02 | 7.44 |
| tech-RL-caida | 191k | 608k | 455k | 21M | 423k | 7.4M | 40M | 583M | 1.7B | 77M | 0.39 | 71.74 |
| tech-WHOIS | 7.5k | 57k | 782k | 5.3M | 12M | 31M | 2.9M | 229M | 566M | 194M | 0.14 | 44.52 |
| tech-as-skitter | 1.7M | 11M | 29M | 16B | 149M | 20B | 43B | 819B | 96T | 162B | 476.06 | N/A |
| web-BerkStan-dir | 685k | 6.6M | 65M | 28B | 1.1B | 99B | 25B | 49B | 382T | 476B | 149.17 | N/A |
| web-edu | 3.0k | 6.5k | 10k | 81k | 40k | 4.6k | 18 | 435k | 1.3M | 186k | <0.001 | 0.52 |
| web-google-dir | 876k | 4.3M | 13M | 687M | 40M | 382M | 38M | 4.1B | 650B | 6.7B | 4.45 | N/A |
| web-indochina-2004 | 11k | 48k | 210k | 481k | 1.2M | 88k | 9.2k | 5.5M | 12M | 4.9M | 0.01 | 24.36 |
| web-it-2004 | 509k | 7.2M | 339M | 56M | 29B | 815M | 175M | 1.1B | 1.4B | 527M | 25.26 | N/A |
| web-baidu-baike | 2.1M | 17M | 25M | 31B | 28M | 4.5B | 9.2B | 3.3T | 571T | 327B | 3975.81 | N/A |
| web-wikipedia-growth | 1.9M | 37M | 127M | 123B | 288M | 38B | 68B | 29T | 3.1P | 3.2T | 22389.2 | N/A |
| web-ClueWeb09-50m | 148M | 447M | 1.2B | 494B | 5.6B | 243B | 774B | 34T | 24P | 3.4T | 91697.4 | N/A |
| inf-italy-osm | 6.7M | 7.0M | 7.4k | 8.2M | 0 | 244 | 47k | 9.9M | 992k | 27k | 0.85 | N/A |
| inf-openflights | 2.9k | 16k | 73k | 639k | 286k | 1.5M | 319k | 17M | 17M | 9.0M | 0.01 | 2.46 |
| inf-power | 4.9k | 6.6k | 651 | 17k | 90 | 385 | 324 | 38k | 20k | 5.1k | <0.001 | 0.58 |
| inf-roadNet-CA | 2.0M | 2.8M | 120k | 5.6M | 40 | 13k | 249k | 11M | 2.4M | 521k | 0.35 | N/A |
| inf-roadNet-PA | 1.1M | 1.5M | 67k | 3.2M | 16 | 5.7k | 152k | 6.2M | 1.4M | 295k | 0.19 | N/A |
| inf-road-usa | 24M | 29M | 439k | 50M | 90 | 21k | 1.6M | 81M | 18M | 1.5M | 4.05 | N/A |
| ia-email-EU-dir | 265k | 364k | 267k | 194M | 581k | 10M | 6.7M | 4.4B | 221B | 341M | 1.52 | 887.18 |
| ia-enron-only | 143 | 623 | 889 | 4.8k | 779 | 2.7k | 648 | 29k | 17k | 14k | <0.001 | 0.12 |
| ia-reality | 6.8k | 7.7k | 400 | 497k | 63 | 1.7k | 2.8k | 1.6M | 26M | 93k | <0.001 | 1.39 |
| ia-wiki-Talk-dir | 2.4M | 4.7M | 9.2M | 13B | 65M | 1.0B | 924M | 1.2T | 192T | 64B | 281.33 | N/A |
| ia-wikiquote-user-edits | 93k | 238k | 279k | 636M | 411k | 70M | 44M | 8.9B | 2.4T | 2.5B | 2.41 | 691.28 |
| ia-wiki-user-edits-page | 2.1M | 5.6M | 6.7M | 550B | 10M | 70B | 44B | 4.8T | 88P | 2.0T | 5691.92 | N/A |
| brock200-3 | 200 | 12k | 291k | 570k | 3.2M | 12M | 4.1M | 11M | 3.5M | 16M | 0.02 | 22.96 |
| brock200-4 | 200 | 13k | 373k | 584k | 5.2M | 16M | 4.3M | 8.9M | 3.0M | 17M | 0.02 | 21.85 |
| brock400-3 | 400 | 60k | 4.4M | 4.5M | 184M | 372M | 63M | 84M | 28M | 251M | 0.4 | 997.15 |
| brock400-4 | 400 | 60k | 4.4M | 4.5M | 185M | 373M | 63M | 84M | 28M | 250M | 0.4 | 1010.26 |

N/A: RAGE was timed out after 30 hours of runtime

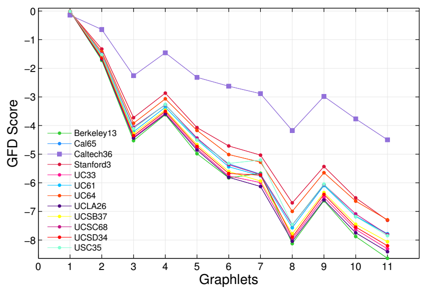


Fig. 4. Facebook social networks of California Universities. Using the space of graphlets of size $k = 4$, Caltech is noticeably different than others.

TABLE III. GRAPH CLASSIFICATION ACCURACY

| graph | Type | No. Graphs | Accuracy(%) | Feature Computation |
|---|---|---|---|---|
| D&D | Protein | 1178 | $76.13 \pm 0.03$ | 1.05 secs |
| MUTAG | Chemicals | 188 | $86.4 \pm 0.21$ | 0.14 secs |

## B. *Large-Scale Graph Comparison & Classification*

Graphlets are also useful for large-scale comparison and classification of graphs. In this case, we relax the notion of equivalence and isomorphism to some form of *structural similarity* between graphs, such that the graph similarity is measured using feature-based graphlet counts. We study the full data set of Facebook100, which contains 100 Facebook networks for a variety of US schools [37]. We plot the GFD (i.e., graphlet frequency distribution) score pictorially in Figure 4 for all California schools. The GFD score is simply the normalized frequencies of graphlets of size $k$ [1]. In our case, we use $k = 4$. The figure shows Caltech noticeably different than others, consistent with the results in [37] which shows how Caltech is well-known to be organized almost exclusively according to its undergraduate "Housing" residence system, in contrast to other schools that follow the "dormitory" residence system.

We use counts of graphlets of size $k = \{2, 3, 4\}$-nodes as features, from which we learn a model to predict the label of the unlabeled graphs (e.g., the function of proteins). We test our approach on protein graphs (D&D collection of 1178 protein graphs) and chemical compound graphs (MUTAG collection of 188 chemical compound graphs) [18]. We extract the graphlet features using Algorithm 2. Then, we learn a model using SVM (RBF kernel), and we use 10-fold validation for evaluation. Table III shows the accuracy of this approach is 76% for protein function prediction, and 86% for mutagenic effect prediction. Note that by using all graphlet-based features up to size 4 nodes, we were able to obtain better accuracy than previous work (which achieved maximum 75% and 83% accuracy for D&D and MUTAG respectively [3]). Moreover, Algorithm 2 extracts all the features (graphlet counts) in almost one second. This yields a significant improvement over the graphlet extraction method in [3], which takes 2.45 hours to extract features from the D&D collection.

## C. *Finding Large Stars, Cliques, and other Patterns Fast*

How can we quickly and efficiently find large cliques, stars, and other unique patterns? Further, how can we identify the top-k largest cliques, stars, etc? Note that many of these problems are NP-hard, e.g., finding the clique of maximum size is a well-known NP-hard problem [27]. To answer these and other related queries, we leverage the proposed parallel graphlet counting method in Algorithm 2. For brevity, many details and results have been removed. However, the idea is clearly shown in Figure 5. Figure 5 provides a visualization of the human diseasome network [38], where we used Alg. 2 to rank (weight) all the edges in the network by the number of star patterns of size 4 nodes. The intuition behind the method is that if an edge (or node) has a (relatively) large number of stars of 4 nodes (cliques, or another graphlet of interest), then it is also likely to be part of a star of a large size. Recall that removing a node from a $k$-star or $k$-clique forms a star or clique of size $k-1$ [27]. Accordingly, edges with large weights are likely to be members of large stars. Thus, as shown in Figure 5, a visualization based on our fast graphlet counting method can help to quickly highlight such large stars by using the counts (of stars of size 4 nodes) as edge weights or colors.

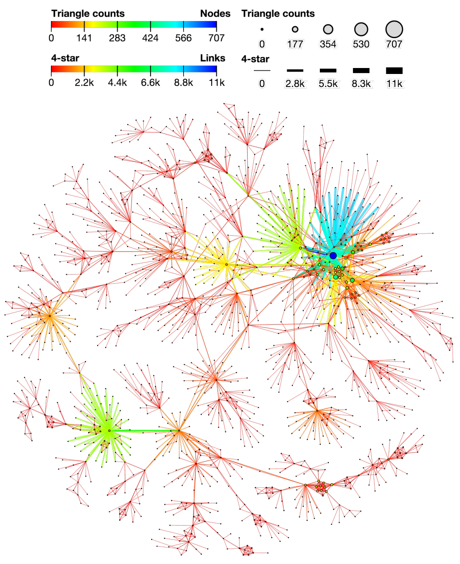


Fig. 5. **Visualization of the human diseasome network**: A network of disorders and disease genes linked by known disorder-gene associations [38]. Edges are weighted/colored by their number of incident star graphlets of size 4 nodes, nodes are weighted/colored by their triangle counts. The large star on the right denoted by light blue color corresponds to colon cancer; the large star on the lower left denoted by lime green color corresponds to deafness; and the large star on the right denoted by lime green color corresponds to leukemia. Notably this figure highlights the few phenotypes (such as colon cancer, leukemia, and deafness) correspond to hubs (large stars) that are connected to a large number of distinct disorders, which is consistent with [38]

## VI. Conclusion & Future Work

In this paper, we proposed a fast, efficient, and parallel algorithm for counting graphlets of size $k = \{3, 4\}$-nodes that take only a fraction of the time to compute when compared with the current methods used. The proposed graphlet counting algorithm leverages a number of proven combinatorial arguments for different graphlets. For each edge, we count a few graphlets, and with these counts along with the combinatorial arguments, we obtain the exact counts of others in constant time. We systematically investigate the scalability of our algorithm on a large collection of 300+ networks from a variety of domains.In future work, we aim to extend our proposed algorithm to higher-order graphlets.

## Acknowledgment

This research is supported by NSF under contract number(s) IIS-1149789 and CCF-0939370. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright notation hereon.

## References

[1] N. Pržulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?" *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, 2004.
[2] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
[3] N. Shervashidze, T. Petri, K. Mehlhorn, K. M. Borgwardt, and S. Vishwanathan, "Efficient graphlet kernels for large graph comparison," in *AISTATS*, 2009.
[4] P. W. Holland and S. Leinhardt, "Local structure in social networks," *Sociological Methodology*, vol. 7, pp. pp. 1–45, 1976.
[5] 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.
[6] O. Frank, "Triad count statistics," *Annals of Disc. Math.*, pp. 141–149, 1988.
[7] M. Granovetter, "The strength of weak ties: A network theory revisited," *Sociological theory*, vol. 1, no. 1, pp. 201–233, 1983.
[8] T. Milenkoviæ and N. Pržulj, "Uncovering biological network function via graphlet degree signatures," *Cancer informatics*, vol. 6, p. 257, 2008.
[9] T. Milenković, W. L. Ng, W. Hayes, and N. Pržulj, "Optimal network alignment with graphlet degree vectors," *Cancer informatics*, vol. 9, p. 121, 2010.
[10] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj, "Topological network alignment uncovers biological function and phylogeny," *Journal of the Royal Society Interface*, vol. 7, no. 50, pp. 1341–1354, 2010.
[11] D. Feldman and Y. Shavitt, "Automatic large scale generation of internet pop level maps," in *IEEE GLOBECOM*, 2008.
[12] D. Hales and S. Arteconi, "Motifs in evolving cooperative networks look like protein structure networks," *Journal of Networks and Heterogeneous Media*, vol. 3, no. 2, 2008.
[13] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *SIGKDD*, 2008.
[14] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, "Graph kernels for chemical informatics," *Neural Networks*, vol. 18, no. 8, pp. 1093–1110, 2005.
[15] H. Kashima, H. Saigo, M. Hattori, and K. Tsuda, "Graph kernels for chemoinformatics," *Chemoinformatics and Advanced Machine Learning Perspectives: Complex Computational Methods and Collaborative Techniques*, p. 1, 2010.
[16] 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.
[17] L. Zhang, Y. Han, Y. Yang, M. Song, S. Yan, and Q. Tian, "Discovering discriminative graphlets for aerial image categories recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5071–5084, 2013.
[18] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, vol. 11, pp. 1201–1242, 2010.
[19] C. C. Noble and D. J. Cook, "Graph-based anomaly detection," in *SIGKDD*, 2003.
[20] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, 2007.
[21] R. Rossi and N. Ahmed, "Role discovery in networks," *TKDE*, 2015.
[22] L. Getoor and B. Taskar, *Introduction to statistical relational learning*. MIT press, 2007.
[23] D. Marcus and Y. Shavitt, "Rage–a rapid graphlet enumerator for large networks," *Computer Networks*, vol. 56, no. 2, pp. 810–819, 2012.
[24] S. Wernicke and F. Rasche, "Fanmod: a tool for fast network motif detection," *Bioinformatics*, vol. 22, no. 9, pp. 1152–1153, 2006.
[25] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, vol. 30, no. 4, pp. 559–565, 2014.
[26] W. Hayes, K. Sun, and N. Pržulj, "Graphlet-based measures are suitable for biological network comparison," *Bioinformatics*, vol. 29, no. 4, pp. 483–491, 2013.
[27] J. L. Gross, J. Yellen, and P. Zhang, *Handbook of Graph Theory, Second Edition*, 2nd ed. Chapman & Hall/CRC, 2013.
[28] B. D. McKay, "Small graphs are reconstructible," *Australasian Journal of Combinatorics*, vol. 15, pp. 123–126, 1997.
[29] P. J. Kelly, "A congruence theorem for trees." *Pacific Journal of Mathematics*, vol. 7, no. 1, pp. 961–968, 1957.
[30] B. Manvel and P. K. Stockmeyer, "On reconstruction of matrices," *Mathematics Magazine*, pp. 218–221, 1971.
[31] T. Kloks, D. Kratsch, and H. Müller, "Finding and counting small induced subgraphs efficiently," *Info. Proc. Letters*, vol. 74, no. 3, pp. 115–121, 2000.
[32] J. Ugander, L. Backstrom, and J. Kleinberg, "Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections," in *WWW*, 2013.
[33] F. Costa and K. De Grave, "Fast neighborhood subgraph pairwise distance kernel," in *ICML*, 2010.
[34] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *ICDM*, 2012.
[35] M. Gonen and Y. Shavitt, "Approximating the number of network motifs," *Internet Mathematics*, vol. 6, no. 3, pp. 349–372, 2009.
[36] R. P. Stanley, *What Is Enumerative Combinatorics?* Springer, 1986.
[37] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of facebook networks," *Physica A: Statistical Mechanics and its Applications*, 2012.
[38] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabási, "The human disease network," *PNAS*, vol. 104, no. 21, pp. 8685–8690, 2007.