Heterogeneous Graphlets

Ryan A. Rossi
Adobe Research

Nesreen K. Ahmed
Intel Labs

Aldo Carranza
Stanford University

David Arbour
Adobe Research

Anup Rao
Adobe Research

Sungchul Kim
Adobe Research

Eunyee Koh
Adobe Research

ABSTRACT

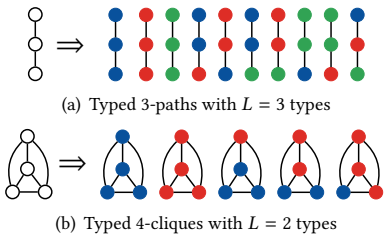
In this work, we generalize the notion of network motifs (graphlets) to heterogeneous networks by introducing the notion of a small induced typed subgraph called *typed graphlet*. Typed graphlets generalize graphlets to rich heterogeneous networks as they explicitly capture the higher-order typed connectivity patterns in such networks. To address this problem, we describe a general framework for counting the occurrences of such typed graphlets. The proposed algorithms leverage a number of combinatorial relationships for different typed graphlets. For each edge, we count a few typed graphlets, and with these counts along with the combinatorial relationships, we obtain the exact counts of the other typed graphlets in $o(1)$ constant time. Notably, the worst-case time complexity of the proposed approach matches the best known untyped algorithm. In addition, the approach lends itself to an efficient lock-free and asynchronous parallelization. The experiments confirm the approach is orders of magnitude faster *and* more space-efficient than existing methods. Unlike existing methods that take hours on small networks, the proposed approach takes only seconds on large networks with millions of edges. This gives rise to new opportunities and applications for typed graphlets on large real-world networks.

KEYWORDS

Heterogeneous graphlets, network motifs, colored motifs, heterogeneous networks, labeled graphs

1 INTRODUCTION

Higher-order connectivity patterns such as small induced subgraphs called graphlets (network motifs)[1] are known to be the fundamental building blocks of simple homogeneous networks [13] and are essential for modeling and understanding the fundamental components of these networks [4, 7]. Furthermore, graphlets are also important for many predictive and descriptive modeling tasks [2, 5, 10, 12, 13, 15, 19–21, 23]. However, such (untyped) graphlets are *unable* to capture the rich (typed) connectivity patterns in more complex networks such as those that are heterogeneous (which includes signed, labeled, bipartite, k-partite, and attributed graphs as special cases). In heterogeneous networks, nodes and edges can be of different types and explicitly modeling such types is crucial [1, 6, 8, 9].



(a) Typed 3-paths with $L = 3$ types

(b) Typed 4-cliques with $L = 2$ types

Figure 1: Examples of heterogeneous graphlets

In this work, we introduce the notion of a *typed graphlet* that naturally generalizes the notion of network motif to heterogeneous networks.[2] Typed graphlets generalize the notion of graphlets to rich heterogeneous networks as they capture both the induced subgraph of interest and the types associated with the nodes in the induced subgraph (Figure 1). These small induced typed subgraphs are the fundamental *building blocks of rich heterogeneous networks*. Typed graphlets naturally capture the higher-order typed connectivity patterns in bipartite, k-partite, signed, labeled, k-star, attributed graphs, and more generally heterogeneous networks. As such, typed graphlets are useful for a wide variety of predictive and descriptive modeling applications in these rich complex networks.

Despite their fundamental and practical importance, counting typed graphlets in large graphs remains a challenging and unsolved problem. To address this problem, we propose a fast, parallel, and space-efficient framework for counting typed graphlets in large networks. The time complexity is provably optimal as it matches the best untyped graphlet counting algorithm. Using non-trivial combinatorial relationships between lower-order $(k-1)$-node typed graphlets, we derive equations that allow us to compute many of the k-node typed graphlet counts in $o(1)$ constant time. Thus, we avoid explicit enumeration of many typed graphlets by simply computing the exact count directly in constant time using the discovered combinatorial relationships. For every edge, we count a few typed graphlets and obtain the exact counts of the remaining typed graphlets in $o(1)$ constant time. Furthermore, we store only the nonzero typed graphlet counts for every edge.

This work generalizes the notion of network motif to heterogeneous networks and describes a framework for finding all such *heterogeneous network motifs*. The proposed framework has the following desired properties:

- **Fast**: The approach is fast for large graphs by leveraging novel *non-trivial combinatorial relationships* to derive many of the typed

[1] The terms graphlet, network motif, and induced subgraph are used interchangeably.

[2] The terms typed, colored, labeled, and heterogeneous graphlet/network motif are used interchangeably.

MLG KDD '19, August 4–8, 2019, Anchorage, AK, USA

graphlets in $o(1)$ constant time. Theoretically, the worst-case time complexity is shown to match the best untyped graphlet algorithm (Section 5.1). As shown in Table 3, the approach is orders of magnitude faster than existing methods.

- **Space-Efficient**: The approach is space-efficient by hashing and storing only the typed motif counts that appear on a given edge. Compared to existing methods, the proposed approach is orders of magnitude more space-efficient (Section 6.2).
- **Scalable for Large Networks**: Unlike existing methods, the proposed approach is scalable for *large* heterogeneous networks.
- **Parallel**: The typed graphlet approach lends itself to an efficient lock-free & asynchronous parallel implementation.
- **Effectiveness**: We demonstrate the utility of typed motifs for graph mining/exploratory analysis (Section 6.4).

2 HETEROGENEOUS GRAPHLETS

This section introduces a generalization of graphlets (network motifs) called *heterogeneous graphlets* (or simply *typed graphlets*).[3]

Definition 1 (Heterogeneous network). *A heterogeneous network is defined as $G = (V, E)$ consisting of a set of node objects V and a set of edges E connecting the nodes in V. A heterogeneous network also has a* node type mapping function *$\phi : V \to \mathcal{T}_V$ and an* edge type mapping function *defined as $\xi : E \to \mathcal{T}_E$ where $\mathcal{T}_V$ and $\mathcal{T}_E$ denote the set of node object types and edge types, respectively. The type of node i is denoted as ϕ_i whereas the type of edge $e = (i, j) \in E$ is denoted as $\xi_{ij} = \xi_e$.*

Figure 2 shows a few special cases of our formulation of heterogeneous networks.

2.1 Graphlet Generalization

In this section, we introduce a more general notion of graphlet called *typed graphlet* that naturally extends to both homogeneous and general heterogeneous networks. We use G to represent a graph and H or F to represent graphlets.

2.1.1 Untyped Graphlets. We begin by defining untyped graphlets for graphs with a single type.

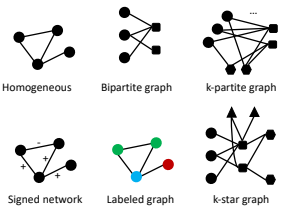
Definition 2 (Untyped Graphlet). *An untyped graphlet H is a connected induced subgraph of G.*

Given a graphlet in some graph, it may be the case that we can find other topologically identical "appearances" of this structure in that graph. We call these "appearances" *graphlet instances*.

Definition 3 (Untyped Graphlet Instance). *An instance of an untyped graphlet H in graph G is an untyped graphlet F in G that is isomorphic to H.*

2.1.2 Typed Graphlets. In heterogeneous graphs, nodes/edges can be of many different types and so explicitly and jointly modeling such types is essential. In this work, we introduce the notion of a *typed graphlet* that explicitly captures both the connectivity pattern of interest and the types. Notice that typed graphlets are a generalization of graphlets to heterogeneous networks.

[3] Additional discussion and results can be found in the longer version of this paper [18].



Graph Type	$\lvert\mathcal{T}_V\rvert$	$\lvert\mathcal{T}_E\rvert$
Homogeneous	1	1
Bipartite	2	1
K-partite	k	$k-1$
Signed	1	2
Labeled	k	ℓ
Star	k	$k-1$

Figure 2: *Heterogeneous graphlets* are useful for a wide variety of different classes of graphs.

Definition 4 (Typed Graphlet). *A typed graphlet of a graph $G = (V, E, \phi, \xi)$ is a connected induced heterogeneous subgraph $H = (V', E', \phi', \xi')$ of G such that*
(1) (V', E') is a graphlet of (V, E),
(2) $\phi' = \phi|_{V'}$, that is, ϕ' is the restriction of ϕ to V',
(3) $\xi' = \xi|_{E'}$, that is, ξ' is the restriction of ξ to E'.

The terms typed graphlet/motif, colored graphlet, and heterogeneous network motif (graphlet) are used interchangeably. See Figure 1 for examples of typed graphlets and untyped graphlets (in which the type structure is ignored).

We can consider the presence of topologically identical "appearances" of a typed graphlet in a graph.

Definition 5 (Typed Graphlet Instance). *An instance of a typed graphlet $H = (V', E', \phi', \xi')$ of graph G is a typed graphlet $F = (V'', E'', \phi'', \xi'')$ of G such that*
(1) (V'', E'') is isomorphic to (V', E'),
(2) $\mathcal{T}_{V''} = \mathcal{T}_{V'}$ and $\mathcal{T}_{E''} = \mathcal{T}_{E'}$, that is, the multisets of node and edge types are correspondingly equal.
The set of typed graphlet instances of H in G is denoted as $I_G(H)$.

Comparing the above definitions of graphlet and typed graphlet, we see at first glance that typed graphlets are nontrivial extensions of their homogeneous counterparts. The "position" of an edge (node) in a typed graphlet is often topologically important, *e.g.*, an edge at the end of the 4-path vs. an edge at the center of a 4-path. These topological differences of a typed graphlet are called (automorphism) *typed orbits* since they take into account "symmetries" between edges (nodes) of a graphlet. Typed graphlet orbits are a generalization of untyped graphlet orbits [14].

2.1.3 Number of Typed Graphlets. For a single K-node untyped motif (*e.g.*, K-clique), the number of *typed motifs* with L types is:

$$\left(\!\!\binom{L}{K}\!\!\right) = \binom{L+K-1}{K} \tag{1}$$

where L = number of types (colors) and K = size of the network motif (# of nodes). Table 1 shows the number of *typed network motifs* that arise from a single motif $H \in \mathcal{H}$ of size $K \in \{2, \ldots, 4\}$ nodes as the number of types varies from $L = 1, 2, \ldots, 9$.

Table 1: Number of *typed graphlets* (for a single untyped graphlet) as the size K and number of types L varies.

	Types L								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
K=2	1	3	6	10	15	21	28	36	45
K=3	1	4	10	20	35	56	84	120	165
K=4	1	5	15	35	70	126	210	330	495

3 FRAMEWORK

This section describes the general framework for counting heterogeneous graphlets. Algorithm 1 shows the general approach for counting typed network motifs. The algorithm naturally handles heterogeneous graphs with arbitrary number of types and structure.

Algorithm 1 Heterogeneous Graphlets

Input: a graph G
Output: nonzero typed graphlet counts $\mathcal{X}_{ij}$ for each edge $(i, j) \in E$

parallel for each $(i, j) \in E$ **do**
  $T^t_{ij} = \Gamma^t_i \cap \Gamma^t_j$, for $t = 1, \ldots, L$ ▷ typed triangles
  $S^t_i = \Gamma^t_i \setminus T^t_{ij}$, for $t = 1, \ldots, L$ ▷ typed 3-paths centered at i
  $S^t_j = \Gamma^t_j \setminus T^t_{ij}$, for $t = 1, \ldots, L$ ▷ typed 3-paths centered at j
  $S^t_{ij} = S^t_i \cup S^t_j$, for $t = 1, \ldots, L$ ▷ typed 3-paths
  Store nonzero counts of the 3-node typed graphlets derived above
  Let $T_{ij} = \bigcup_t T^t_{ij}$, $S_i = \bigcup_t S^t_i$, and $S_j = \bigcup_t S^t_j$
  Given S_i and S_j, derive typed path-based motifs via Algorithm 2
  Given T_{ij}, derive typed triangle-based motifs via Algorithm 3
  for $t, t' \in \{1, \ldots, L\}$ such that $t \leq t'$ **do**
    Derive remaining typed graphlet orbits in constant time via Eq. 13-16 and update counts $\mathbf{x}$ and set of motifs $\mathcal{M}_{ij}$
  for $c \in \mathcal{M}_{ij}$ **do** $\mathcal{X}_{ij} = \mathcal{X}_{ij} \cup \{(c, \mathbf{x}_c)\}$ ▷ nonzero typed motif counts
end parallel

3.1 Counting 3-Node *Typed* Motifs

We begin by introducing the notion of a typed neighborhood *and* typed degree of a node. These are then used as a basis for deriving all typed 3-node motif counts in worst-case $O(\Delta)$ time (Theorem 1).

Definition 6 (Typed Neighborhood). *Given an arbitrary node i in G, the* typed neighborhood Γ^t_i *is the set of nodes with type t that are reachable by following edges originating from i within* 1*-hop distance. More formally,*

$$\Gamma^t_i = \{j \in V \mid (i, j) \in E \wedge \phi_j = t\} \tag{2}$$

Intuitively, a node $j \in \Gamma^t_i$ iff there exists an edge $(i, j) \in E$ between i and j and the type of node j denoted as ϕ_j is t.

Definition 7 (Typed Degree). *The* typed-degree d^t_i *of node i with type t is defined as $d^t_i = |\Gamma^t_i|$ where d^t_i is the number of nodes connected to node i with type t.*

Using these notions as a basis, we can define S^t_i, S^t_j, and T^t_{ij} for $t = 1, \ldots, L$. Obtaining these sets is equivalent to computing all 3-node typed motif counts. These sets are all defined with respect to a given edge $(i, j) \in E$ between node i and j with types ϕ_i and ϕ_j. Since typed graphlets are counted for each edge $(i, j) \in E$, the types ϕ_i and ϕ_j are fixed ahead of time. Thus, there is only one remaining type to select for 3-node typed motifs.

Definition 8 (Typed Triangle Nodes). *Given an edge $(i, j) \in E$ between node i and j with types ϕ_i and ϕ_j, let T^t_{ij} denote the set of nodes of type t that complete a typed triangle with i and j defined as:*

$$T^t_{ij} = \Gamma^t_i \cap \Gamma^t_j \tag{3}$$

where $|T^t_{ij}|$ denotes the number of nodes that form triangles with node i and j of type t. Every node $k \in T^t_{ij}$ is of type t and completes a typed triangle with i and j consisting of types ϕ_i, ϕ_j, and $\phi_k = t$.

Definition 9 (Typed 3-Star Nodes Centered at i). *Given an edge $(i, j) \in E$ between node i and j with types ϕ_i and ϕ_j. Let S^t_i denote the set of nodes of type t that form 3-node stars (or equivalently 3-node paths) centered at node i (and not including j). More formally,*

$$S^t_i = \left\{k \in (\Gamma^t_i \setminus \{j\}) \;\middle|\; k \notin \Gamma^t_j\right\} \tag{4}$$
$$= \Gamma^t_i \setminus \left(\Gamma^t_j \cup \{j\}\right) = \Gamma^t_i \setminus T^t_{ij} \tag{5}$$

where $|S^t_i|$ denotes the number of nodes of type t that form 3-stars centered at node i (not including j).

Similarly, it is straightforward to define the set S^t_j of typed 3-star/path nodes of type t centered at j in a similar fashion:

$$S^t_j = \left\{k \in (\Gamma^t_j \setminus \{i\}) \;\middle|\; k \notin \Gamma^t_i\right\} \tag{6}$$
$$= \Gamma^t_j \setminus \left(\Gamma^t_i \cup \{i\}\right) = \Gamma^t_j \setminus T^t_{ij} \tag{7}$$

Property 1.

$$T_{ij} = \bigcup_{t=1}^{L} T^t_{ij}, \qquad S_i = \bigcup_{t=1}^{L} S^t_i, \qquad S_j = \bigcup_{t=1}^{L} S^t_j \tag{8}$$

These lower-order 3-node typed motif counts are used to derive the counts of many higher-order typed motifs in $o(1)$ constant time (Section 3.3).

Definition 10 (Typed 3-Stars (for an edge)). *Given an edge $(i, j) \in E$ between node i and j with types ϕ_i and ϕ_j, the number of typed 3-node stars that contain $(i, j) \in E$ with types ϕ_i, ϕ_j, t is:*

$$|S^t_{ij}| = |S^t_i| + |S^t_j| \tag{9}$$

where $|S^t_{ij}|$ denotes the number of typed 3-stars that contain nodes i and j with types ϕ_i, ϕ_j, t.

Moreover, the number of typed triangles centered at $(i, j) \in E$ with types ϕ_i, ϕ_j, t is simply $|T^t_{ij}|$ (Definition 8) whereas the number of typed 3-node stars that contain $(i, j) \in E$ with types ϕ_i, ϕ_j, t is $|S^t_{ij}| = |S^t_i| + |S^t_j|$ (Definition 10). We do not need to actually store the sets S^t_i, S^t_j, and T^t_{ij} for every type $t = 1, \ldots, L$. We only need to store the *size/cardinality* of the sets (as shown in Algorithm 1) since these are the counts of all possible 3-node typed graphlets. For convenience, we denote the size of those sets as $|S^t_i|$, $|S^t_j|$, and $|T^t_{ij}|$ for all $t = 1, \ldots, L$, respectively. At this point, all typed 3-node graphlets with nonzero counts have been computed for edge $(i, j) \in E$ in $O(|\Gamma_i| + |\Gamma_i|) = O(\Delta)$ time where Δ is max degree (See Section 5.1 for proof). Note $|\Gamma_i| = \sum_t |\Gamma^t_i|$.

3.2 Counting 4-Node *Typed* Motifs

To derive k-node typed graphlets, the framework leverages the lower-order $(k-1)$-node *typed graphlets*. Therefore, 4-node typed graphlets are derived by leveraging the *typed* sets $T^t_{ij} = \Gamma^t_i \cup \Gamma^t_j$, $S^t_j = \Gamma^t_j \setminus T^t_{ij}$, and $S^t_i = \Gamma^t_i \setminus T^t_{ij}$ (for $t \in \{1, \ldots, L\}$) computed from the lower-order 3-node typed graphlets along with the set I^t of

Algorithm 2 Typed *Path*-based Graphlets

for each $w_k \in S_i$ **do**
  for $w_r \in \Gamma_{w_k} \setminus \{i, j\}$ **do**
    if $w_r \notin (\Gamma_i \cup \Gamma_j)$ **then** ▷ *typed 4-path-edge* orbit
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_3, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
    else if $w_r \in S_i \wedge w_r \leq w_k$ **then** ▷ *typed tailed-tri (tail orbit)*
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_7, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
for each $w_k \in S_j$ **do**
  for $w_r \in \Gamma_{w_k} \setminus \{i, j\}$ **do**
    if $w_r \notin (\Gamma_i \cup \Gamma_j)$ **then** ▷ *typed 4-path-edge* orbit
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_3, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
    else if $w_r \in S_j \wedge w_r \leq w_k$ **then** ▷ *typed tailed-tri (tail orbit)*
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_7, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
    else if $w_r \in S_i$ **then** ▷ *typed 4-cycle*
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_6, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
return set of typed motifs $\mathcal{M}_{ij}$ between i and j and counts $\mathbf{x}$

Algorithm 3 Typed *Triangle*-based Graphlets

for each $w_k \in T_{ij}$ **do**
  for $w_r \in \Gamma_{w_k} \setminus \{i, j\}$ **do**
    if $w_r \in T_{ij} \wedge w_r \leq w_k$ **then** ▷ *typed 4-clique*
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_{12}, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
    else if $w_r \in (S_i \cup S_j)$ **then** ▷ *typed chord-cycle-edge* orbit
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_{10}, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
    else if $w_r \notin (\Gamma_i \cup \Gamma_j)$ **then** ▷ *typed tailed-tri-center* orbit
      $\langle \mathbf{x}, \mathcal{M}_{ij} \rangle = \textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, \mathbb{F}(g_8, \Phi_i, \Phi_j, \Phi_{w_k}, \Phi_{w_r}))$
return set of typed motifs $\mathcal{M}_{ij}$ between i and j and counts $\mathbf{x}$

non-adjacent nodes of type t *w.r.t.* $(i, j) \in E$ defined formally as:

$$I^t = V^t \setminus (\Gamma_i^t \cup \Gamma_j^t) = V^t \setminus (T_{ij}^t \cup S_i^t \cup S_j^t \cup \{i, j\}). \tag{10}$$

where $V^t \subseteq V$ is the set of nodes in V of type t.

Property 2.

$$|V^t| = |I^t| + |\Gamma_i^t| + |\Gamma_j^t| \tag{11}$$

The proof is straightforward by Eq. 10 and applying the principle of inclusion-exclusion.

3.2.1 General Principle for Counting Typed *Graphlets.* We now introduce a general typed graphlet formulation. Let $f_{ij}(H, \mathbf{t})$ denote the number of distinct k-node typed graphlet orbits of H with the type vector $\mathbf{t}$ that contain edge $(i, j) \in E$ and have properties $P \in \{S_i^t, S_j^t, T_{ij}^t, I^t\}$ and $Q \in \{S_i^{t'}, S_j^{t'}, T_{ij}^{t'}, I^{t'}\}$ for any $t, t' \in \{1, \ldots, L\}$ defined as:

$$f_{ij}(H, \mathbf{t}) = \Big| \Big\{ \{i, j, w_k, w_r\} \;\Big|\; w_k \in P \wedge w_r \in Q \wedge \mathbb{I}\{(w_k, w_r) \in E\} \wedge w_r \neq w_k \wedge \mathbf{t} = \begin{bmatrix} \phi_i & \phi_j & \phi_{w_k} & \phi_{w_r} \end{bmatrix} \Big\} \Big| \tag{12}$$

where $\mathbb{I}\{(w_k, w_r) \in E\} = 1$ if $(w_k, w_r) \in E$ holds and 0 otherwise (*i.e.*, $\mathbb{I}\{(w_k, w_r) \in E\} = 0$ if $(w_k, w_r) \notin E$). For clarity and simplicity, $(w_k, w_r) \in E$ or $(w_k, w_r) \notin E$ is sometimes used (*e.g.*, Table 2) as opposed to $\mathbb{I}\{(w_k, w_r) \in E\} = 1$ or $\mathbb{I}\{(w_k, w_r) \in E\} = 0$. The equations for deriving every typed graphlet orbit of size 4 are provided in Table 2. Notice that all typed graphlets with k-nodes are formulated with respect to the typed node sets derived from the typed graphlets with $(k-1)$-nodes. Hence, higher-order typed graphlets of order k are derived from lower-order $(k-1)$-node typed graphlets. We classify typed motifs as path-based or triangle-based. Typed path-based motifs are the typed 4-node motifs derived from the sets $S_i = \bigcup_t S_i^t$ and $S_j = \bigcup_t S_j^t$ of nodes that form 3-node typed paths centered at node i and j, respectively (Algorithm 2). Conversely, typed triangle-based motifs are the typed 4-node motifs derived from the set $T_{ij} = \bigcup_t T_{ij}^t$ of nodes that form typed triangles (typed 3-cliques) with node i and j (Algorithm 3).

The typed graphlet equations in Table 2 are mainly used to characterize the typed graphlets, and of course can be used to count them. However, using those equations to count all typed graphlets is still expensive since some non-negligible work is required to count every typed graphlet. Instead, we count only a few typed graphlets and use newly discovered combinatorial relationships (see Section 3.3) to derive the others directly in $o(1)$ constant time.

Algorithm 4 Update Typed Graphlets

procedure $\textsc{Update}(\mathbf{x}, \mathcal{M}_{ij}, c = \mathbb{F}(g, \Phi_i, \Phi_j, \Phi_k, \Phi_r))$
  if $c \notin \mathcal{M}_{ij}$ **then** $\mathcal{M}_{ij} \leftarrow \mathcal{M}_{ij} \cup \{c\}$ and set $\mathbf{x}_c = 0$
  $\mathbf{x}_c = \mathbf{x}_c + 1$
  return updated set of typed graphlets $\mathcal{M}_{ij}$ and counts $\mathbf{x}$

3.3 Combinatorial Relationships

Now, we show the existence of combinatorial relationships between the different *typed motifs* and demonstrate how they can be leveraged to derive the counts of typed graphlets efficiently. These combinatorial relationships allow us to derive many *typed motifs* in $o(1)$ constant time (avoiding explicit enumeration of the nodes and types involved in those typed motifs) and play a significant role in the speed/efficiency of the proposed approach (see Section 6.1). Since we derive all typed graphlet counts for a given edge $(i, j) \in E$, we already have two types ϕ_i and ϕ_j. Thus, these types are fixed ahead of time. In the case of 4-node typed graphlets, there are two remaining types that need to be selected. Notice that for typed graphlet orbits, we must solve $\frac{L(L-1)}{2} + L$ equations in the worst-case. The counts of all remaining typed graphlets are derived in $o(1)$ constant time (Eq. 13-16) using the counts of the lower-order $(k-1)$-node typed graphlets and a few other counts from the k-node typed graphlets.

Typed 4-Path Center Orbit Count: To count the typed 4-path center orbits for a given edge $(i, j) \in E$ with types ϕ_i and ϕ_j, we simply select the remaining two types denoted as t and t' to obtain the 4-dimensional type vector $\mathbf{t} = \begin{bmatrix} \phi_i & \phi_j & t & t' \end{bmatrix}$ and derive the count directly as follows:

$$f_{ij}(g_4, \mathbf{t}) = \begin{cases} (|S_i^t| \cdot |S_j^t|) - f_{ij}(g_6, \mathbf{t}) & \text{if } t = t' \\ (|S_i^t| \cdot |S_j^{t'}|) + & \text{otherwise} \\ (|S_i^{t'}| \cdot |S_j^t|) - f_{ij}(g_6, \mathbf{t}) & \end{cases} \tag{13}$$

where $f_{ij}(g_6, \mathbf{t})$ is the typed 4-cycle count for edge $(i, j) \in E$ with type vector $\mathbf{t}$.

Typed 4-Star Count: To count the typed 4-stars for a given edge $(i, j) \in E$ with types ϕ_i and ϕ_j, we simply select the remaining two types denoted as t and t' to obtain the 4-dimensional type vector

Table 2: Typed graphlet orbit equations. All typed graphlet orbits with 4-nodes are formulated with respect to the typed node sets $S_i^t, S_j^t, T_{ij}^t, I^t$ for $t = 1, \ldots, L$ derived from the typed 3-node graphlets. Recall $T_{ij}^t = \Gamma_i^t \cap \Gamma_j^t$, $S_j^t = \Gamma_j^t \setminus T_{ij}^t$, $S_i^t = \Gamma_i^t \setminus T_{ij}^t$, and $I^t = V^t \setminus (\Gamma_i^t \cup \Gamma_j^t) = V^t \setminus (T_{ij}^t \cup S_i^t \cup S_j^t \cup \{i, j\})$ where V^t is the set of nodes in V of type t. In all cases, $w_r \neq w_k$.

Typed Motif H	Orbit	$\lvert E(H)\rvert$	$f_{ij}(H, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in P \wedge w_r \in Q \wedge \mathbb{I}\{(w_k, w_r) \in E\} \wedge w_r \neq w_k \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
4-path	edge	3	$f_{ij}(g_3, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in S_i^t \wedge w_r \in I^{t'} \wedge (w_k, w_r) \in E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
	center	3	$f_{ij}(g_4, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid (w_k \in S_j^t) \wedge (w_r \in S_i^{t'}) \wedge (w_k, w_r) \notin E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
4-star		3	$f_{ij}(g_5, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in S_i^t \wedge w_r \in S_i^{t'} \wedge (w_k, w_r) \notin E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
4-cycle		4	$f_{ij}(g_6, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in S_j^t \wedge w_r \in S_i^{t'} \wedge (w_k, w_r) \in E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
tailed-triangle	tail-edge	4	$f_{ij}(g_7, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in S_i^t \wedge w_r \in S_i^{t'} \wedge w_r \neq w_k \wedge (w_k, w_r) \in E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
	center	4	$f_{ij}(g_8, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in T_{ij}^t \wedge w_r \in I^{t'} \wedge (w_k, w_r) \in E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
	tri-edge	4	$f_{ij}(g_9, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in T_{ij}^t \wedge w_r \in S_i^{t'} \wedge (w_k, w_r) \notin E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
chordal-cycle	edge	5	$f_{ij}(g_{10}, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in T_{ij}^t \wedge w_r \in (S_i^{t'} \cup S_j^{t'}) \wedge w_r \neq w_k \wedge (w_k, w_r) \in E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
	center	5	$f_{ij}(g_{11}, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in T_{ij}^t \wedge w_r \in T_{ij}^{t'} \wedge w_r \neq w_k \wedge (w_k, w_r) \notin E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$
4-clique		6	$f_{ij}(g_{12}, \mathbf{t}) = \left\lvert \left\{ \{i, j, w_k, w_r\} \mid w_k \in T_{ij}^t \wedge w_r \in T_{ij}^{t'} \wedge w_r \neq w_k \wedge (w_k, w_r) \in E \wedge \mathbf{t} = \left[\phi_i \;\; \phi_j \;\; \phi_{w_k} \;\; \phi_{w_r}\right] \right\} \right\rvert$

$\mathbf{t} = \left[\phi_i \;\; \phi_j \;\; t \;\; t'\right]$. We derive the typed 4-star counts with the type vector $\mathbf{t}$ for edge $(i, j) \in E$ in constant time as follows:

$$f_{ij}(g_5, \mathbf{t}) = \begin{cases} \binom{|S_i^t|}{2} + \binom{|S_j^t|}{2} - f_{ij}(g_7, \mathbf{t}) & \text{if } t = t' \\ (|S_i^t| \cdot |S_i^{t'}|) + & \text{otherwise} \\ (|S_j^t| \cdot |S_j^{t'}|) - f_{ij}(g_7, \mathbf{t}) & \end{cases} \tag{14}$$

where $f_{ij}(g_7, \mathbf{t})$ is the tailed-triangle tail-edge orbit count for edge $(i, j) \in E$ with type vector $\mathbf{t}$. The only path-based typed graphlet containing a triangle is the tailed-triangle tail-edge orbit. Observe that this is the only orbit needed to derive typed 4-star counts in $o(1)$ constant time.

Typed Tailed-Triangle Tri-Edge Orbit Count:

$$f_{ij}(g_9, \mathbf{t}) = \begin{cases} \left((|T_{ij}^t| \cdot (|S_i^t| + |S_j^t|)\right) - f_{ij}(g_{10}, \mathbf{t}) & \text{if } t = t' \\ \left((|T_{ij}^t| \cdot (|S_i^{t'}| + |S_j^{t'}|)\right) + & \text{otherwise} \\ \left((|T_{ij}^{t'}| \cdot (|S_i^t| + |S_j^t|)\right) - f_{ij}(g_{10}, \mathbf{t}) & \end{cases} \tag{15}$$

where $f_{ij}(g_{10}, \mathbf{t})$ is the chordal-cycle edge orbit count for edge $(i, j) \in E$ with type vector $\mathbf{t}$.

Typed Chordal-Cycle Center Orbit Count:

$$f_{ij}(g_{11}, \mathbf{t}) = \begin{cases} \binom{|T_{ij}^t|}{2} - f_{ij}(g_{12}, \mathbf{t}) & \text{if } t = t' \\ (|T_{ij}^t| \cdot |T_{ij}^{t'}|) - f_{ij}(g_{12}, \mathbf{t}) & \text{otherwise} \end{cases} \tag{16}$$

where $f_{ij}(g_{12}, \mathbf{t})$ is the typed 4-clique count for edge $(i, j) \in E$ with type vector $\mathbf{t}$.

3.4 From *Typed* Orbits to Graphlets

Counts of the *typed graphlets* for each edge $(i, j) \in E$ can be derived from the *typed graphlet orbits* using the following equations:

$$f_{ij}(h_3, \mathbf{t}) = f_{ij}(g_3, \mathbf{t}) + f_{ij}(g_4, \mathbf{t}) \tag{17}$$
$$f_{ij}(h_4, \mathbf{t}) = f_{ij}(g_5, \mathbf{t}) \tag{18}$$
$$f_{ij}(h_5, \mathbf{t}) = f_{ij}(g_6, \mathbf{t}) \tag{19}$$
$$f_{ij}(h_6, \mathbf{t}) = f_{ij}(g_7, \mathbf{t}) + f_{ij}(g_8, \mathbf{t}) + f_{ij}(g_9, \mathbf{t}) \tag{20}$$
$$f_{ij}(h_7, \mathbf{t}) = f_{ij}(g_{10}, \mathbf{t}) + f_{ij}(g_{11}, \mathbf{t}) \tag{21}$$
$$f_{ij}(h_8, \mathbf{t}) = f_{ij}(g_{12}, \mathbf{t}) \tag{22}$$

where h is the graphlet without considering the orbit (Table 2).

3.5 *Typed* Motif Hash Functions

Given a general heterogeneous graph with L unique types such that $L < 10$, then a simple and efficient typed motif hash function $\mathbb{F}$ is defined as follows:

$$\mathbb{F}(g, \mathbf{t}) = g10^4 + t_1 10^3 + t_2 10^2 + t_3 10^1 + t_4 \tag{23}$$

where g encodes the k-node motif orbit (*e.g.*, 4-path center) and t_1, t_2, t_3, t_4 encode the type of the nodes in $H \in \mathcal{H}$ with type vector $\mathbf{t} = \left[t_1 \;\; t_2 \;\; t_3 \;\; t_4\right]$. Since the maximum hash value resulting from Eq. 23 is small (and fixed for any arbitrarily large graph G), we can leverage a perfect hash table to allow for fast $o(1)$ constant time lookups to determine if a typed motif was previously found or not as well as updating the typed motif count in $o(1)$ constant time. For k-node motifs where $k < 4$, we simply set the last $4 - k$ types to 0. Note the simple typed motif hash function defined above can be extended trivially to handle graphs with $L \geq 10$ types:

$$\mathbb{F}(g, \mathbf{t}) = g10^8 + t_1 10^6 + t_2 10^4 + t_3 10^2 + t_4 \tag{24}$$

In general, any non-cryptographic hash function $\mathbb{F}$ can be used. Thus, the approach is independent of $\mathbb{F}$ and can always leverage the best known hash function. Thus far we have not made any assumption on the ordering of types in $\mathbf{t}$. As such, the hash function

$\mathbb{F}$ discussed above can be used directly in the framework for counting typed graphlets such that the type structure and position are preserved. However, since we are interested in counting all typed graphlets *w.r.t.* Definition 5, then we map all such orderings of the types in $\mathbf{t}$ to the same hash value using a precomputed hash table. This allows us to obtain the unique hash value in $o(1)$ constant time for any ordering of the types in $\mathbf{t}$. In our implementation, we compute $s = t_1 10^3 + t_2 10^2 + t_3 10^1 + t_4$ and then use s as an index into the precomputed hash table to obtain the unique hash value c in $o(1)$ constant time.

4 GLOBAL TYPED GRAPHLET COUNTS

Definition 11 (Global Typed Graphlets). *Given a graph G with L types, the global typed graphlet counting problem is to find the set of all typed motifs that occur in G along with their frequencies.*

A general equation for solving the above problem for any arbitrary *typed graphlet* H is given below. Let H denote an arbitrary typed graphlet and $\mathbf{x}$ be an M-dimensional vector of counts of H for every edge $(i, j) \in E$, then the frequency of H in G is:

$$C_H = \frac{1}{|E(H)|} \mathbf{x}^\top \mathbf{e} \tag{25}$$

where $|E(H)|$ is the number of edges in the typed graphlet H and $\mathbf{e} = [\, 1 \; \cdots \; 1 \,]$ is an M-dimensional vector of all 1's.

5 THEORETICAL ANALYSIS

5.1 Time Complexity

Theorem 1. *The worst-case time complexity for counting all 3-node typed graphlets for a given edge $(i, j) \in E$ is:*

$$O(2|\Gamma_i| + |\Gamma_j|) = O(\Delta) \tag{26}$$

where $|\Gamma_i|$ and $|\Gamma_j|$ denote the number of nodes connected to node i and j, respectively. Further, Δ is the maximum degree in G.

Proof. It takes at most $O(|\Gamma_i| + |\Gamma_j|)$ time to compute typed triangles (*i.e.*, T^t_{ij}, for all $t = 1, \ldots, L$) by hashing neighbors of i in $O(|\Gamma_i|)$ time, and then checking if each node $w \in \Gamma_j$ is hashed or not, taking $O(|\Gamma_j|)$ time. Similarly, if $w \in \Gamma_j$ is not hashed, then $S^t_j \leftarrow S^t_j \cup \{w\}$ where $t = \phi_w$. Now all that remains is computing S^t_i, for all t. Notice $|S^t_i| = |\Gamma^t_i| - |T^t_{ij}|$, for all $t = 1, \ldots, L$. ■

5.1.1 Typed 4-Node Graphlets. We first provide the worst-case time complexity of deriving *typed path*-based and *typed triangle*-based graphlet orbits in Lemma 5.1-5.2, and then give the total worst-case time complexity of all 3 and 4-node typed graphlets in Theorem 2 based on these results. Note that Lemma 5.1-5.2 includes the time required to derive all typed 3-node typed graphlets.

Lemma 5.1. *For a single edge $(i, j) \in E$, the worst-case time complexity for deriving all* typed path-based graphlet orbits *is:*

$$O\Big(\Delta\big(|S_i| + |S_j|\big)\Big) \tag{27}$$

Note $|S_i|\Delta \geq \sum_{k \in S_i} d_k$ and $|S_j|\Delta \geq \sum_{k \in S_j} d_k$.

Lemma 5.2. *For a single edge $(i, j) \in E$, the worst-case time complexity for deriving all* typed triangle-based graphlet orbits *is:*

$$O\big(\Delta|T_{ij}|\big) \tag{28}$$

Notice $|T_{ij}|\Delta \geq |T_{ij}|\Delta_T \geq \sum_{k \in T_{ij}} d_k$ where Δ is the maximum degree of a node in G and Δ_T is the maximum degree of a node in T_{ij}. Thus, $|T_{ij}|\Delta$ only occurs iff $\forall k \in T_{ij}$, $d_k = \Delta$ where $\Delta =$ maximum degree of a node in G. In sparse real-world graphs, T_{ij} is likely to be smaller than S_i and S_j as triangles are typically more rare than 3-node paths. Conversely, T_{ij} is also more likely to contain high degree nodes, as nodes with larger degrees are obviously more likely to form triangles than those with small degrees.

From Lemma 5.1-5.2, we have the following:

Theorem 2. *For a single edge $(i, j) \in E$, the worst-case time complexity for deriving all 3 and 4-node typed graphlet orbits is:*

$$O\big(\Delta\big(|S_i| + |S_j| + |T_{ij}|\big)\big) \tag{29}$$

Proof. The time complexity of each step is provided below. Hashing all neighbors of node i takes $O(|\Gamma_i|)$. Recall from Lemma 1 that counting all 3-node typed graphlets takes $O(2|\Gamma_i|+|\Gamma_j|) = O(\Delta)$ time for an edge $(i, j) \in E$. This includes the time required to derive the number of typed 3-node stars and typed triangles for all types $t = 1, \ldots, L$. This information is needed to derive the remaining typed graphlet orbit counts in constant time. Next, Algorithm 2 is used to derive a few path-based typed graphlet orbit counts taking $O(\Delta(|S_i| + |S_j|))$ time in the worst-case. Similarly, Algorithm 3 is used to derive a few triangle-based typed graphlet orbit counts taking in the worst-case $O(\Delta|T_{ij}|)$ time. As an aside, updating the count of a typed graphlet count is $o(1)$ (Algorithm 4).

Now, we derive the remaining typed graphlet orbit counts in constant time (Line 10-11). Since each type pair leads to different typed graphlets, we must iterate over at most $L(L-1)/2 + L$ type pairs. For each pair of types selected, we derive the typed graphlet orbit counts in $o(1)$ constant time via Eq. 13-16 (See Line 10-11). Furthermore, the term involving L is for the worst-case when there is at least one node in all L sets (*i.e.*, at least one node of every type L). Nevertheless, since L is a small constant, $L(L-1)/2+L$ is negligible. Therefore, for a single edge, the worst-case time complexity is $O(\Delta(|S_i| + |S_j| + |T_{ij}|))$.

Let $\bar{T}$ and $\bar{S}$ denote the average number of triangle and 3-node stars incident to an edge in G. More formally, $\bar{T} = \frac{1}{M} \sum_{(ij) \in E} |T_{ij}|$ and $\bar{S} = \frac{1}{M} \sum_{(ij) \in E} |S_i| + |S_j|$. The total worst-case time complexity for all M edges is $O(M\Delta(\bar{S} + \bar{T}))$. Note that obviously $\bar{S}M = \sum_{(ij) \in E} |S_i| + |S_j|$ and $\bar{T}M = \sum_{(ij) \in E} |T_{ij}|$. ■

Corollary 1. *The worst-case time complexity of counting typed graphlets using Algorithm 1 matches the worst-case time complexity of the best known untyped graphlet counting algorithm.*

Proof. From Theorem 2 we have that $O\big(\Delta\big(|S_i| + |S_j| + |T_{ij}|\big)\big)$, which is exactly the time complexity of the best known untyped graphlet counting algorithm [3, 4]. ■

5.2 Space Complexity

Since our approach generalizes to graphs with an arbitrary number of types L, the specific set of typed motifs is unknown. As demonstrated in Table 1, it is impractical to store the counts of all possible k-node typed motifs for any graph of reasonable size as typically done in traditional methods for untyped graphlets [3, 11]. Despite this being obviously impractical due to the amount of space that

Table 3: Results comparing the proposed approach to the state-of-the-art methods in terms of runtime performance (seconds). Since existing methods are unable to handle large or even medium-sized graphs as shown below, we include a number of very small graphs (*e.g.*, cora, citeseer, webkb) for comparison. Note Δ = max degree; $|\mathcal{T}_V|$ = # of node types; $|\mathcal{T}_E|$ = # of edge types.

						SECONDS				SPEEDUP (OURS VS.)		
	$\|V\|$	$\|E\|$	Δ	$\|\mathcal{T}_V\|$	$\|\mathcal{T}_E\|$	**GC**	**ESU**	**G-Tries**	**Ours**	**GC**	**ESU**	**G-Tries**
citeseer	3.3k	4.5k	99	6	21	46.27	5937.75	144.08	**0.022**	2103x	269897x	6549x
cora	2.7k	5.3k	168	7	28	467.20	10051.07	351.40	**0.032**	14600x	314095x	10981x
fb-relationship	7.3k	44.9k	106	6	20	1374.60	54,837.69	3789.17	**0.701**	1960x	78227x	5405x
web-polblogs	1.2k	16.7k	351	2	1	28,986.70	26,577.10	1,563.04	**1.055**	27475x	25191x	1481x
ca-DBLP	2.9k	11.3k	69	3	3	149.20	1,188.11	18.90	**0.100**	1492x	11881x	189x
inf-openflights	2.9k	15.7k	242	2	2	9262.20	18,839.36	458.01	**0.578**	16024x	32594x	792x
soc-wiki-elec	7.1k	100.8k	1.1k	2	2	ETL	ETL	26,468.85	**5.316**	∞	∞	45793x
webkb	262	459	122	5	14	85.82	7,158.10	187.22	**0.006**	14303x	1193016x	31203x
terrorRel	881	8.6k	36	2	3	192.6	3130.7	241.1	**0.039**	4938x	80274x	6182x
pol-retweet	18.5k	48.1k	786	2	3	ETL	ETL	ETL	**0.296**	∞	∞	∞
web-spam	9.1k	465k	3.9k	3	6	ETL	ETL	ETL	**210.97**	∞	∞	∞
movielens	28.1k	170.4k	3.6k	3	3	ETL	ETL	ETL	**5.23**	∞	∞	∞
citeulike	907.8k	1.4M	11.2k	3	2	ETL	ETL	ETL	**126.53**	∞	∞	∞
yahoo-msg	100.1k	739.8k	9.4k	2	2	ETL	ETL	ETL	**35.22**	∞	∞	∞
dbpedia	495.9k	921.7k	24.8k	4	3	ETL	ETL	ETL	**56.02**	∞	∞	∞
digg	217.3k	477.3k	219	2	2	ETL	ETL	ETL	**5.592**	∞	∞	∞
bibsonomy	638.8k	1.2M	211	3	3	ETL	ETL	ETL	**3.631**	∞	∞	∞
epinions	658.1k	2.6M	775	2	2	ETL	ETL	ETL	**85.27**	∞	∞	∞
flickr	2.3M	6.8M	216	2	2	ETL	ETL	ETL	**120.79**	∞	∞	∞
orkut	6M	37.4M	166	2	2	ETL	ETL	ETL	**1241.01**	∞	∞	∞

* ETL = Exceeded Time Limit (24 hours / 86,400 seconds)

would be required, existing methods such as GC [9] store counts of all possible typed graphlets, and therefore the space complexity of such methods is at least $O(MT_{\max})$ where $M = |E|$ is the number of edges in G and $T_{\max}$ is the number of different possible typed graphlets with L types. In contrast, Algorithm 1 is orders of magnitude more space-efficient.

LEMMA 5.3. *The space complexity of typed graphlets is* $O(M\bar{T})$.

PROOF. For an edge $(i, j) \in E$, it takes $|\mathcal{X}_{ij}|$ space to store the counts of the nonzero typed graphlets. Let $\bar{T} = \frac{1}{M}\sum_{(ij)\in E}|\mathcal{X}_{ij}|$ denote the average number of typed graphlets with nonzero counts per edge. Therefore, the total space required to store the nonzero typed graphlet counts for all $M = |E|$ edges is only $O(M\bar{T})$. The space of all other data structures used in Algorithm 1 is small in comparison, *e.g.*, Ψ takes at most $O(|V|)$ space, whereas T_{ij}, S_i, and S_j take $O(\Delta)$ space in the worst-case (by Property 1) and can be reused for every edge. In addition, the size of $\mathbf{x}$ is independent of the graph size ($|V| + |E|$) and can also be reused. ■

From Lemma 5.3, it is straightforward to see that

$$O(M\bar{T}) \ll O(MT_{\max}) \tag{30}$$

The space required by the proposed approach (Algorithm 1) is nearly-optimal and orders of magnitude lower than the next best method. This is also shown empirically in Table 4.

6 EXPERIMENTS

The experiments are designed to investigate the effectiveness of the approach for computing heterogeneous graphlets in large networks.

6.1 Runtime Comparison

We first demonstrate how fast the proposed framework is for deriving typed graphlets by comparing the runtime (in seconds) of our approach against the three existing methods, namely, ESU (using fanmod) [22], G-Tries [16], and GC [9]. Since existing methods are inherently serial (and difficult to parallelize), we use a serial version of the proposed approach. We also note that the three existing methods count typed graphlets for every node whereas the proposed approach derives typed graphlets for every edge; and many of these methods count only a subset of the typed graphlets obtained by the proposed approach. Nevertheless, these methods are used for comparison since they are the closest to our own work and solve conceptually simpler problems.

For comparison, we use a variety of heterogeneous and labeled / attributed graphs from different domains. All data can be accessed at NetworkRepository [17]. In Table 3, we report the time (in seconds) required by each method and the speedup of our approach. Strikingly, the existing methods are unable to handle medium to large graphs with hundreds of thousands or more nodes and edges as shown in Table 3. Even small graphs can take hours to finish using existing methods (Table 3). For instance, on the small cora graph with 2.7K nodes and 5.3K edges, GC takes 467 seconds whereas G-Tries takes 351 seconds. However, our approach finishes counting all typed motifs with $\{2, 3, 4\}$-nodes in only 0.03 seconds. This is 10,000 times faster than the next best method. Unlike existing methods, our approach is shown to be significantly faster and able to handle large-scale graphs. This is primarily due to the new non-trivial combinatorial relationships that we leverage to derive a number of typed graphlets in $o(1)$ constant time whereas GC and other typed graphlet methods must enumerate all graphlets in order to obtain their type/color configuration. These methods require a lot of extra work to compute the typed graphlets that we can derive directly in $o(1)$ constant time with a few equations. Across all graphs, the proposed method achieves significant speedups over the existing methods as shown in Table 3 (last 3 columns). These

results demonstrate the effectiveness of our approach for *counting typed graphlets* in large real-world networks.

6.2 Space Efficiency Comparison

We theoretically showed the space complexity of our approach in Section 5.2. In this section, we empirically investigate the space-efficiency of our approach compared to ESU (using fanmod) [22], G-Tries [16], and GC [9]. Table 4 reports the space used by each method for a variety of real-world graphs. Strikingly, the proposed approach uses between 42x and 776x less space than existing methods as shown in Table 4. These results indicate that our approach is space-efficient and practical for large networks.

Table 4: Comparing the *space* used by the proposed typed graphlet approach to the state-of-the-art methods.

	citeseer	cora	movielens	web-spam
GC	30.1MB	50.4MB	ETL	ETL
ESU	13.4MB	46.2MB	ETL	ETL
G-Tries	161.9MB	448.6MB	ETL	ETL
Ours	**316KB**	**578KB**	**22.5MB**	**128.9MB**

* ETL = Exceeded Time Limit (24 hours / 86,400 seconds)

6.3 Parallel Speedup

This section evaluates the parallel scaling of the proposed approach. In these experiments, we used a two processor, Intel Xeon E5-2686 v4 system with 256 GB of memory. None of the experiments came close to using all the memory. Parallel speedup is simply $S_p = \frac{T_1}{T_p}$ where T_1 is the execution time of the sequential algorithm, and T_p is the execution time of the parallel algorithm with p processing units (cores). In Figure 3, we observe nearly linear speedup as we increase the number of cores. These results indicate the effectiveness of the parallel algorithm for counting typed graphlets in general heterogeneous graphs.

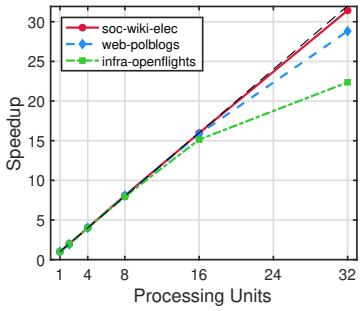


Figure 3: Parallel speedup of the proposed approach.

6.4 Exploratory Analysis

This section demonstrates the use of heterogeneous graphlets for mining and exploratory analysis.

6.4.1 Political retweets. The political retweet data consists of 18,470 Twitter users. To study the (higher-order) structural characteristics of users in this network *w.r.t.* their political orientation, we assign types to nodes based on their political leanings (*i.e.*, left, right). Interestingly, the 24,815 (untyped) triangles in this network are distributed as follows:

$$\mathbf{p} = \begin{bmatrix} \triangle_{LLL} & \triangle_{LLR} & \triangle_{LRR} & \triangle_{RRR} \\ 0.608 & 0.003 & 0.001 & 0.388 \end{bmatrix}$$

Notably, we observe that 60.86% and 38.79% of the 24,815 triangles are formed among users with the same political leanings. These homogeneous typed triangles ($\triangle_{LLL}$, $\triangle_{RRR}$) account for 99.65% of the 24,815 triangles. This implies that three users with the same political leanings are more likely to retweet each other than with users of different political leanings. These results highlight the importance of heterogeneous graphlets, since such key observations and insights would not be possible using untyped graphlets. We also investigated typed 4-clique motifs. Notably, only 4 of the 5 typed 4-clique motifs that arise from 2 types actually occur in the graph. In particular, the typed 4-clique motif with 2 right users and 2 left users does not even appear in the graph. This typed motif might indicate collusion between individuals from different political parties or some other rare anomalous activity.

7 CONCLUSION

In this work, we generalized the notion of network motif to heterogeneous networks. We proposed a fast and space-efficient framework for counting heterogeneous graphlets. The approach counts only a few typed graphlets and derives the others in $o(1)$ constant time using new non-trivial combinatorial relationships that involve counts of lower-order typed graphlets. Thus, it avoids explicit enumeration of any nodes involved in those typed graphlets. Theoretically, the worst-case time complexity of the approach is shown to match the best untyped graphlet algorithm. Given the ubiquity of heterogeneous networks and the predictive and descriptive power of heterogeneous graphlets, we posit that typed graphlets will be an essential tool for many real-world applications. This work gives rise to new opportunities and applications for typed graphlets.

REFERENCES

[1] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. 2011. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv:1105.3422* (2011).

[2] Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. 2017. On Sampling from Massive Graph Streams. In *VLDB*. 1430–1441.

[3] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. 2015. Efficient Graphlet Counting for Large Networks. In *ICDM*. 10.

[4] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, Nick Duffield, and Theodore L. Willke. 2016. Graphlet Decomposition: Framework, Algorithms, and Applications. *KAIS* (2016), 1–32.

[5] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2018. Learning Role-based Graph Embeddings. In *StarAI IJCAI*.

[6] Arindam Banerjee, Sugato Basu, and Srujana Merugu. 2007. Multi-way clustering on relation graphs. In *SDM*. SIAM, 145–156.

[7] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[8] Aldo G. Carranza, Ryan A. Rossi, Anup Rao, and Eunyee Koh. 2018. Higher-order Spectral Clustering for Heterogeneous Graphs. In *arXiv:1810.02959*. 15.

[9] Shawn Gu, John Johnson, Fazle E Faisal, and Tijana Milenković. 2018. From homogeneous to heterogeneous network alignment via colored graphlets. *Scientific reports* 8, 1 (2018), 12524.

[10] Wayne Hayes, Kai Sun, and Nataša Pržulj. 2013. Graphlet-based measures are suitable for biological network comparison. *Bioinformatics* 29, 4 (2013), 483–491.

[11] Dror Marcus and Yuval Shavitt. 2012. RAGE–a rapid graphlet enumerator for large networks. *Computer Networks* 56, 2 (2012), 810–819.

[12] Tijana Milenković and Nataša Pržulj. 2008. Uncovering Biological Network Function via Graphlet Degree Signatures. *Cancer Informatics* 6 (2008), 257.

[13] R Milo, S Shen-Orr, S Itzkovitz, N Kashtan, D Chklovskii, and U Alon. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science* 298, 5594 (2002), 824–827.

[14] Nataša Pržulj. 2007. Biological network comparison using graphlet degree distribution. *Bioinfo.* 23, 2 (2007), e177–e183.

[15] N Pržulj, Derek G Corneil, and Igor Jurisica. 2004. Modeling interactome: scale-free or geometric? *Bioinformatics* 20, 18 (2004), 3508–3515.
[16] Pedro Ribeiro and Fernando Silva. 2014. Discovering colored network motifs. In *Complex Networks V*. Springer, 107–118.
[17] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. 4292–4293. http://networkrepository.com
[18] Ryan A Rossi, Nesreen K Ahmed, Aldo Carranza, David Arbour, Anup Rao, Sungchul Kim, and Eunyee Koh. 2019. Heterogeneous Network Motifs. *arXiv:1901.10026* (2019).
[19] Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi-Yadkori. 2018. HONE: Higher-Order Network Embeddings. *arXiv:1801.09303* (2018).
[20] Nino Shervashidze, Tobias Petri, Kurt Mehlhorn, Karsten M Borgwardt, and Svn Vishwanathan. 2009. Efficient graphlet kernels for large graph comparison. In *AISTATS*.
[21] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *JMLR* 11 (2010), 1201–1242.
[22] Sebastian Wernicke and Florian Rasche. 2006. FANMOD: a tool for fast network motif detection. *Bioinformatics* 22, 9 (2006), 1152–1153.
[23] L. Zhang, R. Hong, Y. Gao, R. Ji, Q. Dai, and X. Li. 2016. Image Categorization by Learning a Propagated Graphlet Path. *TNNLS* 27, 3 (2016), 674–685.