Higher-Order Clustering for Heterogeneous Networks via Typed Motifs

Aldo G. Carranza
Stanford University

Ryan A. Rossi
Adobe Research

Anup Rao
Adobe Research

Eunyee Koh
Adobe Research

ABSTRACT

Higher-order connectivity patterns such as small induced subgraphs called graphlets (network motifs) are vital to understand the important components (modules/functional units) governing the configuration and behavior of complex networks. Existing work in higher-order clustering has focused on simple *homogeneous graphs* with a single node/edge type. However, heterogeneous graphs consisting of nodes and edges of different types are seemingly ubiquitous in the real-world. In this work, we introduce the notion of typed-graphlet that explicitly captures the rich (typed) connectivity patterns in heterogeneous networks. Using typed-graphlets as a basis, we develop a general principled framework for higher-order clustering in heterogeneous networks. The framework provides mathematical guarantees on the optimality of the higher-order clustering obtained. The experiments demonstrate the effectiveness of the framework quantitatively for three important applications including (i) clustering, (ii) link prediction, and (iii) graph compression. In particular, the approach achieves a mean improvement of 43x over all methods and graphs for clustering while achieving a 18.7% and 20.8% improvement for link prediction and graph compression, respectively.

KEYWORDS

Clustering, higher-order clustering, heterogeneous networks, typed graphlets, network motifs, spectral clustering, node embedding, graph mining

1 INTRODUCTION

Clustering in graphs has been one of the most fundamental tools for analyzing and understanding the components of complex networks. It has been used extensively in many important applications to distributed systems [40, 81, 86], compression [19, 68], image segmentation [30, 78], document and word clustering [28], among others. Most clustering methods focus on simple flat/*homogeneous* graphs where nodes and edges represent a single entity and relationship type, respectively. However, heterogeneous graphs consisting of nodes and edges of different types are seemingly ubiquitous in the real-world. In fact, most real-world systems give rise to rich heterogeneous networks that consist of multiple types of diversely interdependent entities [77, 83]. This heterogeneity of real systems is often due to the fact that, in applications, data usually contains semantic information. For example in research publication networks, nodes can represent authors, papers, or venues and edges can represent coauthorships, references, or journal/conference appearances. Such heterogeneous graph data can be represented by an arbitrary number of matrices and tensors that are coupled with respect to one or more types as shown in Figure 1.

Clusters in heterogeneous graphs that contain multiple types of nodes give rise to communities that are significantly more complex. Joint analysis of multiple graphs may capture fine-grained clusters that would not be captured by clustering each graph individually as shown in [13, 69]. For instance, simultaneously clustering different types of entities/nodes in the heterogeneous graph based on multiple relations where each relation is represented as a matrix or a tensor (Figure 1). It is due to this complexity and the importance of explicitly modeling how those entity types mix to form complex communities that make the problem of heterogeneous graph clustering a lot more challenging. Moreover, the complexity, representation, and modeling of the heterogeneous graph data itself also makes this problem challenging (See Figure 1). Extensions of clustering methods for homogeneous graphs to heterogeneous graphs are often nontrivial. Many methods require complex schemas and are very specialized, allowing for two graphs with particular structure. Furthermore, most clustering methods only consider first order structures in graphs, *i.e.*, edge connectivity information. However, higher-order structures play a non-negligible role in the organization of a network.

Higher-order connectivity patterns such as small induced subgraphs called graphlets (network motifs) are known to be the fundamental building blocks of simple homogeneous networks [54] and are essential for modeling and understanding the fundamental components of these networks [3, 4, 14]. However, such (untyped) graphlets are *unable* to capture the rich (typed) connectivity patterns in more complex networks such as those that are heterogeneous, labeled, signed, or attributed. In heterogeneous graphs (Figure 1), nodes and edges can be of different types and explicitly modeling such types is crucial. In this work, we introduce the notion of a typed-graphlet and use it to uncover the higher-order organization of rich heterogeneous networks. The notion of a typed-graphlet captures both the connectivity pattern of interest and the types. We argue that typed-graphlets are the fundamental *building blocks of heterogeneous networks*. Note homogeneous, labeled, signed, and attributed graphs are all special cases of heterogeneous graphs as shown in Section 2.

In this paper, we propose a general framework for *higher-order clustering in heterogeneous graphs*. The framework explicitly incorporates heterogeneous higher-order information by counting

typed graphlets that explicitly capture node and edge types. Typed graphlets generalize the notion of graphlets to rich heterogeneous networks as they explicitly capture the higher-order typed connectivity patterns in such networks. Using these as a basis, we propose the notion of *typed-graphlet conductance* that generalizes the traditional conductance to higher-order structures in heterogeneous graphs. The proposed approach reveals the higher-order organization and composition of rich heterogeneous complex networks. Given a graph and a typed-graphlet of interest H, the framework forms the weighted typed-graphlet adjacency matrix $\mathbf{W}_{G^H}$ by counting the frequency that two nodes co-occur in an instance of the typed-graphlet. Next, the typed-graphlet Laplacian matrix is formed from $\mathbf{W}_{G^H}$ and the eigenvector corresponding to the second smallest eigenvalue is computed. The components of the eigenvector provide an ordering σ of the nodes that produce nested sets $S_k = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ of increasing size k. We demonstrate theoretically that S_k with the minimum typed-graphlet conductance is a near-optimal higher-order cluster.

The framework provides mathematical guarantees on the optimality of the higher-order clustering obtained. The theoretical results extend to typed graphlets of arbitrary size and avoids restrictive special cases required in prior work. Specifically, we prove a Cheeger-like inequality for *typed-graphlet conductance*. This gives bounds $OPT \leq APPX \leq C\sqrt{OPT}$ where OPT is the minimum typed-graphlet conductance, $APPX$ is the value given by Algorithm 1, and C is a constant—at least as small as $\sqrt{OPT}$ which depends on the number of edges in the chosen typed graphlet. Notably, the bounds of the method depend directly on the number of edges of the arbitrarily chosen typed graphlet (as opposed to the number of nodes) and inversely on the quality of connectivity of occurrences of the typed graphlet in a heterogeneous graph. This is notable as the formulation for homogeneous graphs and untyped graphlets proposed in [14] is in terms of nodes and requires different theory for untyped-graphlets with a different amount of nodes (*e.g.*, untyped graphlets with 3 nodes vs. 4 nodes and so on). In this work, we argue that it is not the number of nodes in a graphlet that are important, but the number of edges. This leads to a more powerful, simpler, and general framework that can serve as a basis for analyzing higher-order spectral methods. Furthermore, even in the case of untyped graphlets and homogeneous graphs, the formulation in this work leads to tighter bounds for certain untyped graphlets. Consider a 4-node star and 3-node clique (triangle), both have 3 edges, and therefore would have the same bounds in our framework even though the number of nodes differ. However, in [14], the bounds for the 4-node star would be different (and larger) than the 3-node clique. This makes the proposed formulation and corresponding bounds more general and in the above case provides tighter bounds compared to [14].

The experiments demonstrate the effectiveness of the approach quantitatively for three important tasks. First, we demonstrate the approach for revealing high quality clusters across a wide variety of graphs from different domains. In all cases, it outperforms a number of state-of-the-art methods with an overall improvement of 43x over all graphs and methods. Second, we investigate the approach for link prediction. In this task, we derive higher-order typed-graphlet node embeddings (as opposed to clustering) and use these embeddings

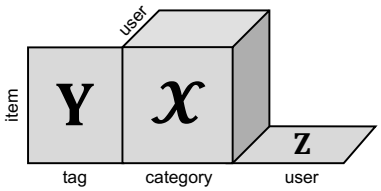


Figure 1: Heterogeneous graph represented as a third-order tensor and two matrices that all share at least one type. The third-order tensor $\mathcal{X}$ can be coupled with the item by tag matrix Y and the social network (user by user) matrix Z.

to learn a predictive model. Compared to state-of-the-art methods, the approach achieves an overall improvement in F_1 and AUC of 18.7% and 14.4%, respectively. Finally, we also demonstrate the effectiveness of the approach quantitatively for graph compression where it is shown to achieve a mean improvement of 20.8% across all graphs and methods. Notably, these application tasks all leverage different aspects of the proposed framework. For instance, link prediction uses the higher-order node embeddings given by our approach whereas graph compression leverages the proposed typed-graphlet spectral ordering (Definition 11).

The paper is organized as follows. Section 2 describes the general framework for higher-order spectral clustering whereas Section 3 proves a number of important results including mathematical guarantees on the optimality of the higher-order clustering. Next, Section 4 demonstrate the effectiveness of the approach quantitatively for a variety of important applications including clustering, link prediction, and graph compression. Section 5 discusses and summarizes related work. Finally, Section 6 concludes.

2 FRAMEWORK

In this work, we propose a general framework for higher-order clustering in heterogeneous graphs. Table 1 lists all our notation.

2.1 Heterogeneous Graph Model

We represent a heterogeneous complex system using the following heterogeneous graph model.

Definition 1 (Heterogeneous Graph). *A heterogeneous graph is an ordered tuple $G = (V, E, \psi, \xi)$ comprised of*

(1) a graph (V, E) where V is the node set and E is the edge set,
(2) a mapping $\psi : V \to \mathcal{T}_V$ referred to as the node-type mapping where $\mathcal{T}_V$ is a set of node types,
(3) a mapping $\xi : E \to \mathcal{T}_E$ referred to as the edge-type mapping where $\mathcal{T}_E$ is a set of edge types.

We denote the node set of a heterogeneous graph G as $V(G)$ and its edge set as $E(G)$.

A homogeneous graph can be seen as a special case of a heterogeneous graph where $|\mathcal{T}_V| = |\mathcal{T}_E| = 1$. Note that a heterogeneous graph may be unweighted or weighted and it may be undirected or directed, depending on the underlying graph structure. Moreover, it may also be signed or labeled $Y = \{y_1, y_2, \ldots\}$ where y_i corresponds to a label assigned to node v_i (or edge e_i).

In general, a heterogeneous network can be represented by an arbitrary number of matrices and tensors that are coupled, *i.e.*, the tensors and matrices share at least one type with each other [1, 70]. See Figure 1 for an example of a heterogeneous network represented as a coupled matrix-tensor.

2.2 Graphlets

Graphlets are small connected induced subgraphs [3, 61]. The simplest nontrivial graphlet is the 1st-order structure of a node pair connected by an edge. Higher-order graphlets correspond to graphlets with greater number of nodes and edges. Most graph clustering algorithms only take into account edge connectivity, 1st-order graphlet structure, when determining clusters. Moreover, these methods are only applicable for homogeneous graphs. For example, spectral clustering on the normalized Laplacian of the adjacency matrix of a graph partitions it in a way that attempts to minimize the amount of edges, 1st-order structures, cut [42].

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

2.2.1 Untyped graphlets. We begin by defining graphlets for homogeneous graphs with a single type.

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

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

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

As we shall soon see, it will be important to refer to the set of all instances of a given graphlet in a graph. Forming this set is equivalent to determining the subgraphs of a graph isomorphic to the given graphlet. Nevertheless, we usually only consider graphlets with up to four or five nodes, and have fast methods for discovering instances of such graphlets [2–4, 6, 72].

2.2.2 Typed graphlets. In heterogeneous graphs, nodes and edges can be of different types and so explicitly (and jointly) modeling such types is essential (Figure 1). To generalize higher-order clustering to handle such networks, 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 untyped-graphlets and thus are a more powerful representation.

Definition 4 (Typed Graphlet). *A typed graphlet of a heterogeneous graph $G = (V, E, \psi, \xi)$ is a connected induced heterogeneous subgraph $H = (V', E', \psi', \xi')$ of G in the following sense:*

(1) (V', E') is an untyped graphlet of (V, E),
(2) $\psi' = \psi|_{V'}$, that is, ψ' is the restriction of ψ to V'
(3) $\xi' = \xi|_{E'}$, that is, ξ' is the restriction of ξ to E'.

We can consider the topologically identical "appearances" of a typed graphlet in a graph that preserve the type structure.

Definition 5 (Typed-Graphlet Instance). *An instance of a typed graphlet $H = (V', E', \psi', \xi')$ of heterogeneous graph G is a typed graphlet $F = (V'', E'', \psi'', \xi'')$ 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 sets of node and edge types are correspondingly equal.

The set of unique typed-graphlet instances of H in G is denoted as $I_G(H)$.

Note that we are not interested in preserving the type structure via the isomorphism, only its existence, that is, we are not imposing the condition that the node and edge types coincide via the graph isomorphism. This condition is too restrictive.

2.2.3 Motifs. Before we proceed, we briefly address some discrepancies between our definition of graphlets and that of papers such as [11, 14]. Although it might be a simple matter of semantics, the differences should be noted and clarified to avoid confusion. Some papers refer to what we refer to graphlets as *motifs*. Yet, motifs usually refer to *recurrent* and *statistically significant* induced subgraphs [54, 61].

To find the motifs of a graph, one must compare the frequency of appearances of a graphlet in the graph to the expected frequency of appearances in an ensemble of random graphs in a null model associated to the underlying graph. Current techniques for computing the expected frequency in a null model requires us to generate a graph that follows the null distribution and then compute the graphlet frequencies in this sample graph [7, 54]. These tasks are computationally expensive for large networks as we have to sample many graphs from the null distribution. On the other hand, any graphlet can be arbitrarily specified in a graph and does not depend on being able to determine whether it is is statistically significant. In any case, a motif is a special type of graphlet, so we prefer to work with this more general object.

2.3 Typed-Graphlet Conductance

In this section, we introduce the measure that will score the quality of a heterogeneous graph clustering built from typed graphlets. It is extended from the notion of conductance defined as:

$$\phi(S, \bar{S}) = \frac{\text{cut}(S, \bar{S})}{\min\left(\text{vol}(S), \text{vol}(\bar{S})\right)}$$

where $(S, \bar{S})$ is a cut of a graph, $\text{cut}(S, \bar{S})$ is the number of edges crossing cut $(S, \bar{S})$ and $\text{vol}(S)$ is the total degrees of the vertices in cluster S [31, 42]. Note that its minimization achieves the sparsest balanced cut in terms of the total degree of a cluster.

The following definitions apply for a fixed heterogeneous graph and typed graphlet. Assume we have a heterogeneous graph G and a typed graphlet H of G.

Note. *We denote the set of unique instances of H in G as $I_G(H)$.*

Definition 6 (Typed-Graphlet Degree). *The typed-graphlet degree based on H of a node $v \in V(G)$ is the total number of incident edges to v over all unique instances of H. We denote and compute this*

Table 1: Summary of notation. Matrices are bold, upright roman letters.

G	graph
$V(G)$	node set of G
$E(G)$	edge set of G
H, F	graphlet of G
$I_G(H)$	set of unique instances of H in G
$\mathbf{W}_{G^H}$	typed-graphlet adjacency matrix of G based on H
$\mathbf{L}_{G^H}$	typed-graphlet normalized Laplacian of G based on H
G^H	weighted heterogeneous graph induced by $\mathbf{W}_G^H$
S	subset of $V(G)$
$(S, \bar{S})$	cut of G where $\bar{S} = V(G) \backslash S$
$\deg_G(v)$	degree of node $v \in V(G)$
$\deg_G^H(v)$	typed-graphlet degree of node $v \in V(G)$ based on H
$\mathrm{vol}_G(S)$	volume of S under G
$\mathrm{vol}_G^H(S)$	typed-graphlet volume of S based on H under G
$\mathrm{cut}_G(S, \bar{S})$	cut size of $(S, \bar{S})$ under G
$\mathrm{cut}_G^H(S, \bar{S})$	typed-graphlet cut size of $(S, \bar{S})$ based on H under G
$\phi_G(S, \bar{S})$	conductance of $(S, \bar{S})$ under G
$\phi_G^H(S, \bar{S})$	typed-graphlet conductance of $(S, \bar{S})$ based on H under G
$\phi(G)$	conductance of G
$\phi^H(G)$	typed-graphlet conductance of G based on H

as

$$\deg_G^H(v) = \sum_{F \in I_G(H)} \left| \{ e \in E(F) \mid v \in e \} \right|.$$

Definition 7 (Typed-Graphlet Volume). *The typed-graphlet volume based on H of a subset of nodes $S \subset V(G)$ is the total number of incident edges to any node in S over all instances of H. In other words, it is the sum of the typed-graphlet degrees based on H over all nodes in S. We denote and compute this as*

$$\mathrm{vol}_G^H(S) = \sum_{v \in S} \deg_G^H(v).$$

Recall that a cut in a graph G is a partition of the underlying node set $V(G)$ into two proper, nonempty subsets S and $\bar{S}$ where $\bar{S} = V(G) \backslash S$. We denote such a cut as an ordered pair $(S, \bar{S})$. For any given cut in a graph, we can define a notion of cut size.

Definition 8 (Typed-Graphlet Cut Size). *The typed-graphlet cut size based on H of a cut $(S, \bar{S})$ in G is the number of unique instances of H crossing the cut. We denote and compute this as*

$$\mathrm{cut}_G^H(S, \bar{S}) = \left| \{ F \in I_G(H) \mid V(F) \cap S \neq \varnothing, V(F) \cap \bar{S} \neq \varnothing \} \right|.$$

Note that a graphlet can cross a cut with any of its edges. Therefore, it has more ways in which it can add to the cut size than just an edge.

Having extended notions of volume and cut size for higher-order typed substructures, we can naturally introduce a corresponding notion of conductance.

Definition 9 (Typed-Graphlet Conductance). *The typed-graphlet conductance based on H of a cut $(S, \bar{S})$ in G is*

$$\phi_G^H(S, \bar{S}) = \frac{\mathrm{cut}_G^H(S, \bar{S})}{\min\left(\mathrm{vol}_G^H(S), \mathrm{vol}_G^H(\bar{S}) \right)},$$

and the typed-graphlet conductance based on H of G is defined to be the minimum typed-graphlet conductance based on H over all possible cuts in G:

$$\phi^H(G) = \min_{S \subset V(G)} \phi_G^H(S, \bar{S}). \tag{1}$$

The cut which achieves the minimal typed-graphlet conductance corresponds to the cut that minimizes the amount of times instances of H are cut and still achieves a balanced partition in terms of instances of H in the clusters.

2.4 Typed-Graphlet Laplacian

In this section, we introduce a notion of a higher-order Laplacian of a graph. Assume we have a heterogeneous graph G and a typed graphlet H of G.

2.4.1 *Typed-graphlet adjacency matrix.*

Suppose we have the set $I_G(H)$. Then, we can form a matrix that has the same dimensions as the adjacency matrix of G and has its entries defined by the count of unique instances of H containing edges in G.

Definition 10 (Typed-Graphlet Adjacency Matrix). *Suppose that $V(G) = \{v_1, \ldots, v_n\}$. The typed-graphlet adjacency matrix $\mathbf{W}_{G^H}$ of G based on H is a* weighted matrix *defined by*

$$(\mathbf{W}_{G^H})_{ij} = \sum_{F \in I_G(H)} \mathbf{1}\big(\{v_i, v_j\} \in E(F)\big)$$

for $i, j = 1, \ldots, n$. That is, the ij-entry of $\mathbf{W}_{G^H}$ is equal to the number of unique instances of H that contain nodes $\{v_i, v_j\} \subset V(G)$ as an edge.

Having defined $\mathbf{W}_{G^H}$, a weighted adjacency matrix on the set of nodes $V(G)$, we can induce a weighted graph. We refer to this graph as the graph induced by $\mathbf{W}_{G^H}$ and denote it as G^H.

Note. *From the definition of $\mathbf{W}_{G^H}$, we can easily show that $E(F) \subset E(G^H)$ for any $F \in I_G(H)$.*

2.4.2 *Typed-graphlet Laplacian.*

We can construct the weighted normalized Laplacian of W_{G^H}:

$$\mathbf{L}_{G^H} = \mathbf{I} - \mathbf{D}_{G^H}^{-1/2} \mathbf{W}_{G^H} \mathbf{D}_{G^H}^{-1/2}$$

where $\mathbf{D}_{G^H}$ is defined by

$$(\mathbf{D}_{G^H})_{ii} = \sum_j (\mathbf{W}_{G^H})_{ij}$$

for $i = 1, \ldots, n$. We also refer to this Laplacian as the *typed-graphlet normalized Laplacian* based on H of G. The normalized typed-graphlet Laplacian is the fundamental structure for the method we present in Section 2.5.

2.5 Typed-Graphlet Spectral Clustering

In this section, we present an algorithm for approximating the optimal solution to the minimum typed-graphlet conductance optimization problem:

$$S_{\text{best}} = \operatorname*{argmin}_{S \subset V(G)} \phi_G^H(S, \bar{S}) \tag{2}$$

Minimizing the typed-graphlet conductance encapsulates what we want: the solution achieves a bipartition of G that minimizes the number of instances of H that are cut and is balanced in terms

of the total graphlet degree contribution of all instances of H on each partition.

The issue is that minimizing typed-graphlet conductance is *NP*-hard. To see this, consider the case where your graphlet is the 1st-order graphlet, that is, a pair of nodes connected by an edge. Minimizing the standard notion of conductance, which is known to be NP-hard [26], reduces to minimizing this special case of 1st-order untyped-graphlet conductance minimization. Therefore, obtaining the best graphlet-preserving clustering for large graphs is an intractible problem. We can only hope to achieve a near-optimal approximation.

2.5.1 *Algorithm.*

We present a typed-graphlet spectral clustering algorithm for finding a provably near-optimal bipartition in Algorithm 1. We build a sweeping cluster in a greedy manner according to the typed-graphlet spectral ordering defined as follows.

Definition 11 (Typed-Graphlet Spectral Ordering). *Let v denote the eigenvector corresponding to the 2nd smallest eigenvalue of the normalized typed-graphlet Laplacian $\mathbf{L}_{G^H}$. The typed-graphlet spectral ordering is the permutation*

$$\sigma = (i_1, i_2, \ldots, i_n)$$

of coordinate indices $(1, 2, \ldots, n)$ such that

$$v_{i_1} \leq v_{i_2} \leq \cdots \leq v_{i_n},$$

that is, σ is the permutation of coordinate indices of v that sorts the corresponding coordinate values from smallest to largest.

Algorithm 1: Typed-Graphlet Spectral Clustering

Input: Heterogeneous graph G, typed graphlet H
Output: Near-optimal cluster
$\mathbf{W}_{G^H} \leftarrow$ typed-graphlet adjacency matrix of G based on H
$N \leftarrow$ number of connected components of G^H
$\phi_{\min} \leftarrow \infty$
$S_{\text{best}} \leftarrow$ initialize space for best cluster
for $i \leftarrow 1$ ***to*** N **do**
  $\mathbf{W} \leftarrow$ submatrix of $\mathbf{W}_{G^H}$ on connected component i
  $\mathbf{L} \leftarrow$ typed-graphlet normalized Laplacian of W
  $\mathbf{v}_2 \leftarrow$ eigenvector of $\mathbf{L}$ with 2nd smallest eigenvalue
  $\sigma \leftarrow \text{argsort}(\mathbf{v}_2)$
  $\phi \leftarrow \min_k \phi_{G^H}(S_k, \bar{S}_k)$, where $S_k = \{\sigma_1, \ldots, \sigma_k\}$
  if $\phi < \phi_{\min}$ **then**
    $\phi_{\min} \leftarrow \phi$
    $S \leftarrow \text{argmin}_k \phi_{G^H}(S_k, \bar{S}_k)$
    if $|S| < |\bar{S}|$ **then**
      $S_{\text{best}} \leftarrow S$
    else
      $S_{\text{best}} \leftarrow \bar{S}$
    end
  end
end
return S_{best}

2.5.2 *Extensions.*

Algorithm 1 generalizes the spectral clustering method for standard conductance minimization [78] and untyped-graphlet conductance minimization. We demonstrated the reduction of standard conductance minimization above. Untyped-graphlet conductance minimization is also generalized since homogeneous graphs can be seen as heterogeneous graphs with a single node and edge type. It is straightforward to adapt the framework to other arbitrary (sparse) cut functions such as ratio cuts [34], normalized cuts [78], bisectors [34], normalized association cuts [78], among others [31, 75, 78].

Multiple clusters can be found through simple recursive bipartitioning [42]. We could also embed the lower k eigenvectors of the normalized typed-graphlet Laplacian into a lower dimensional Euclidean space and perform k-means, or any other Euclidean clustering algorithm, then associate to each node its corresponding cluster in this space [42, 58]. It is also straightforward to use multiple typed-graphlets for clustering or embeddings as opposed to using only a single typed-graphlet independently. For instance, the higher-order typed-graphlet adjacency matrices can be combined in some fashion (*e.g.*, summation) and may even be assigned weights based on the importance of the typed-graphlet. Moreover, the typed-graphlet conductance can be adapted in a straightforward fashion to handle multiple typed-graphlets.

2.5.3 *Discussion.*

Benson et al. [14] refers to their higher-order balanced cut measure as *motif conductance* and it differs from our proposed notion of typed-graphlet conductance. However, the definition used matches more with a generalization known as the *edge expansion*. The edge expansion of a cut $(S, \bar{S})$ is defined as

$$\psi(S, \bar{S}) = \frac{\text{cut}(S, \bar{S})}{\min(|S|, |\bar{S}|)}. \tag{3}$$

The balancing is in terms of the number of vertices in a cluster. Motif conductance was defined with a balancing in terms of the number of vertices in any graphlet instance. To be precise, for any set of vertices S, let the cluster size of S in G based on H be

$$|S|_G^H = \sum_{F \in I_G(H)} \sum_{v \in V(F)} \mathbf{1}(v \in S) = \sum_{v \in S} \sum_{F \in I_G(H)} \mathbf{1}\left(v \in V(F)\right). \tag{4}$$

Note that this does not take into account the degree contributions of each graphlet, only its node count contributions to a cluster S. In terms of our notation, untyped "motif conductance" of a cut $(S, \bar{S})$ is defined in that work as

$$\psi_G^H(S, \bar{S}) = \frac{\text{cut}_G^H(S, \bar{S})}{\min\left(|S|_G^H, |\bar{S}|_G^H\right)}.$$

Since this does not take into account node degree information, this is more of a generalization of edge expansion [9, 41], "graphlet expansion", if you will, rather than conductance. The difference is worth noting because it has been shown that conductance minimization gives better partitions than expansion minimization [42]. By only counting nodes, we give equal importance to all the vertices in a graphlet. Arguably, it is more reasonable to give greater importance to the vertices that not only participate in many graphlets but also have many neighbors within a graphlet and give lesser importance to vertices that have more neighbors that do not participate in a graphlet or do not have many neighbors within a graphlet. Our

definition of typed-graphlet volume captures this idea to give an appropriate general notion of conductance.

2.6 Typed-Graphlet Node Embeddings

Algorithm 2 summarizes the method for deriving higher-order typed motif-based node embeddings (as opposed to clusters/partitions of nodes, or an ordering for compression/analysis, see Section 4.3). In particular, given a typed-graphlet adjacency matrix, Algorithm 2 outputs a $N \times D$ matrix $\mathbf{Z}$ of node embeddings. For graphs with many connected components, Algorithm 2 is called for each connected component of G^H and the resulting embeddings are stored in the appropriate locations in the overall embedding matrix $\mathbf{Z}$.

Multiple typed-graphlets can also be used to derive node embeddings. One approach that follows from [66] is to derive low-dimensional node embeddings for each typed-graphlet of interest using Algorithm 2. After obtaining all the node embeddings for each typed-graphlet, we can simply concatenate them all into one single matrix $\mathbf{Y}$. Given $\mathbf{Y}$, we can simply compute another low-dimensional embedding to obtain the final node embeddings that capture the important latent features from the node embeddings from different typed-graphlets.

Algorithm 2: Typed-Graphlet Spectral Embedding

Input: Heterogeneous graph G, typed graphlet H, embedding dimension D

Output: Higher-order embedding matrix $\mathbf{Z} \in \mathbb{R}^{N \times D}$ for H

$\left(\mathbf{W}_{G^H}\right)_{ij} \leftarrow$ # instances of H containing i and j, $\forall (i,j) \in E$

$\mathbf{D}_{G^H} \leftarrow$ typed-graphlet degree matrix $\left(\mathbf{D}_{G^H}\right)_{ii} = \sum_j \left(\mathbf{W}_{G^H}\right)_{ij}$

$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D \leftarrow$ eigenvectors of D smallest eigenvalues of $\mathbf{L}_{G^H} = \mathbf{I} - \mathbf{D}_{G^H}^{-1/2} \mathbf{W}_{G^H} \mathbf{D}_{G^H}^{-1/2}$

$Z_{ij} \leftarrow X_{ij} \Big/ \sqrt{\sum_{j=1}^{D} X_{ij}^2}$

return $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_n \end{bmatrix}^T \in \mathbb{R}^{N \times D}$

3 THEORETICAL ANALYSIS

In this section, we show the near-optimality of Algorithm 1. The idea is to translate what we know about ordinary conductance for weighted homogeneous graphs, for which there has been substantial theory developed [22–24], to this new measure we introduce of typed-graphlet conductance by relating these two quantities. Through this association, we can derive Cheeger-like results for $\phi^H(G)$ and for the approximation given by the typed-graphlet spectral clustering algorithm (Algorithm 1). As in the previous section, assume we have a heterogeneous graph G and a typed graphlet H. Also, assume we have the weighted graph G^H induced from the typed-graphlet adjacency matrix $\mathbf{W}_G^H$.

We prove two lemmas from which our main theorem will immediately hold. Lemma 1 shows that the typed-graphlet volume and ordinary volume measures match: total typed-graphlet degree contributions of typed-graphlet instances matches with total counts of typed-graphlet instances on edges for any given subset of nodes. In contrast, Lemma 2 shows that equality does not hold for the notions of cut size. The reason lies in the fact that for any typed-graphlet instance, typed-graphlet cut size on G only counts the number of typed-graphlet instances cut whereas ordinary cut size on G^H counts the number of times typed-graphlet instances are cut. Therefore, these two measure at least match and at most differ by a factor equal to the size of the H, which is a fixed value that is small for the typed graphlets we are interested in, of size 3 or 4. Thus, we are able to reasonably bound the discrepancy between the notions of cut sizes.

Using these two lemmas, we immediately get our main result in Theorem 1 which shows the relationship between $\phi^H(G)$ and $\phi(G^H)$ in the form of tightly bound inequality that is dependent only on the number of edges in H. From this theorem, we arrive at two important corollaries. In Corollary 1, we prove Cheeger-like bounds for typed-graphlet conductance. In Corollary 2, we show that the output of Algorithm 1 gives a near-optimal solution up to a square root factor and it goes further to show bounds in terms of the optimal value $\phi^H(G)$ to show the constant of the approximation algorithm which depends on the second smallest eigenvalue of the typed-graphlet adjacency matrix and the number of edges in H. This last result does not give a purely constant-factor approximation to the graph conductance because of their dependence on G and H, yet it still gives a very efficient, and non-trivial, approximation for fixed G and H. Moreover, the second part of Corollary 2 provides intuition as to what makes a specific typed-graphlet a suitable choice for higher-order clustering. Typed-graphlets that have a good balance of small edge set size and strong connectivity in the heterogeneous graph—in the sense that the second eigenvalue of the normalized typed-graphlet Laplacian is large—will have a tighter upper bound to their approximation for minimum typed-graphlet conductance. Therefore, this last result in Corollary 2 provides a way to quickly and quantitatively measure how good a typed graphlet is for determining higher-order organization before even executing the clustering algorithm.

Note. *In the case of the simple 1st-order untyped graphlet,* i.e., *a node pair with an interconnecting edge, we recover the results for traditional spectral clustering since* $|E(H)| = 1$ *in this case. Furthermore, if* G *is a homogeneous graph,* i.e., $|\mathcal{T}_V| = |\mathcal{T}_E| = 1$, *we get the special case of untyped graphlet-based spectral clustering. Therefore, our framework generalizes the methods of traditional spectral clustering and untyped-graphlet spectral clustering for homogeneous graphs.*

In the following analysis, we let $\mathbf{1}(\cdot)$ represent the Boolean predicate function and let $(\mathbf{W}_{G^H})_e$ be the edge weight of edge e in G^H.

Lemma 1. *Let* S *be a subset of nodes in* $V(G)$. *Then,*

$$\mathrm{vol}_{G^H}(S) = \mathrm{vol}_G^H(S).$$

Proof.

$$\begin{aligned}
\mathrm{vol}_{G^H}(S) &= \sum_{v\in S} \deg_{G^H}(v) && (5)\\
&= \sum_{v\in S}\sum_{e\in E(G^H)} \mathbf{1}(v\in e)\cdot(\mathbf{W}_{G^H})_e && (6)\\
&= \sum_{v\in S}\sum_{e\in E(G^H)} \mathbf{1}(v\in e)\cdot\sum_{F\in I_G(H)} \mathbf{1}(e\in E(F)) && (7)\\
&= \sum_{v\in S}\sum_{F\in I_G(H)}\sum_{e\in E(G^H)} \mathbf{1}(v\in e)\cdot\mathbf{1}(e\in E(F)) && (8)\\
&= \sum_{v\in S}\sum_{F\in I_G(H)}\sum_{e\in E(F)} \mathbf{1}(v\in e) && (9)\\
&= \sum_{v\in S}\sum_{F\in I_G(H)} |\{e\in E(F)\mid v\in e\}| && (10)\\
&= \sum_{v\in S} \deg^H_G(v) && (11)\\
&= \mathrm{vol}^H_G(S). && (12)
\end{aligned}$$

■

Lemma 2. *Let $(S,\bar{S})$ be a cut in G. Then,*

$$\frac{1}{|E(H)|}\,\mathrm{cut}_{G^H}(S,\bar{S}) \le \mathrm{cut}^H_G(S,\bar{S}) \le \mathrm{cut}_{G^H}(S,\bar{S}).$$

Proof. For subsequent simplification, we define $[S,\bar{S}]$ to be the set of edges in $E(G^H)$ that cross cut $(S,\bar{S})$:

$$[S,\bar{S}] := \{e\in E(G^H)\mid e\cap S\neq\varnothing, e\cap\bar{S}\neq\varnothing\}. \quad (13)$$

Then,

$$\begin{aligned}
\mathrm{cut}_{G^H}(S,\bar{S}) &= \sum_{e\in E(G^H)} \mathbf{1}\left(e\in[S,\bar{S}]\right)\cdot(\mathbf{W}_{G^H})_e && (14)\\
&= \sum_{e\in E(G^H)} \mathbf{1}\left(e\in[S,\bar{S}]\right)\cdot\sum_{F\in I_G(H)}\mathbf{1}(e\in E(F)) && (15)\\
&= \sum_{F\in I_G(H)}\sum_{e\in E(G^H)} \mathbf{1}\left(e\in[S,\bar{S}]\right)\cdot\mathbf{1}(e\in E(F)) && (16)\\
&= \sum_{F\in I_G(H)}\sum_{e\in E(F)} \mathbf{1}\left(e\in[S,\bar{S}]\right) && (17)\\
&= \sum_{F\in I_G(H)} |E(F)\cap[S,\bar{S}]| && (18)
\end{aligned}$$

Note that for an instance $F\in I_G(H)$ such that $E(F)\cap[S,\bar{S}]\neq\varnothing$, there exists at least one edge in $E(F)$ cut by $(S,\bar{S})$ and at most all edges in $E(F)$ are cut by $(S,\bar{S})$. Clearly, if $E(F)\cap[S,\bar{S}]=\varnothing$, then no edge is cut by $(S,\bar{S})$. This shows that for such an instance we have

$$1\le|E(F)\cap[S,\bar{S}]|\le|E(F)|=|E(H)|. \quad (19)$$

Therefore, Equation 18 satisfies the following inequalities:

$$\sum_{F\in I_G(H)} \mathbf{1}\left(E(F)\cap[S,\bar{S}]\neq\varnothing\right) \le \mathrm{cut}_{G^H}(S,\bar{S}) \quad (20)$$

$$|E(H)|\cdot\sum_{F\in I_G(H)} \mathbf{1}\left(E(F)\cap[S,\bar{S}]\neq\varnothing\right) \ge \mathrm{cut}_{G^H}(S,\bar{S}). \quad (21)$$

Referring to Definition 8 for typed-graphlet cut size and noting that since H is a connected graph,

$$\mathbf{1}(E(F)\cap[S,\bar{S}]\neq\varnothing) = \mathbf{1}(V(F)\cap S\neq\varnothing, V(F)\cap\bar{S}\neq\varnothing), \quad (22)$$

we find that

$$\sum_{F\in I_G(H)} \mathbf{1}(E(F)\cap[S,\bar{S}]) = \mathrm{cut}^H_G(S,\bar{S}). \quad (23)$$

Plugging this into Inequalities 20-21, we get

$$\mathrm{cut}^H_G(S,\bar{S}) \le \mathrm{cut}_{G^H}(S,\bar{S}) \le |E(H)|\,\mathrm{cut}^H_G(S,\bar{S}) \quad (24)$$

or, equivalently,

$$\frac{1}{|E(H)|}\,\mathrm{cut}_{G^H}(S,\bar{S}) \le \mathrm{cut}^H_G(S,\bar{S}) \le \mathrm{cut}_{G^H}(S,\bar{S}) \quad (25)$$

■

Theorem 1.

$$\frac{1}{|E(H)|}\cdot\phi(G^H) \le \phi^H(G) \le \phi(G^H)$$

Proof. Let $(S,\bar{S})$ be any cut in G. From Lemma 2, we have that

$$\frac{1}{|E(H)|}\,\mathrm{cut}_{G^H}(S,\bar{S}) \le \mathrm{cut}^H_G(S,\bar{S}) \le \mathrm{cut}_{G^H}(S,\bar{S}). \quad (26)$$

Lemma 1 shows that $\mathrm{vol}_{G^H}(S)=\mathrm{vol}^H_G(S)$. Therefore, if we divide these inequalities above by $\mathrm{vol}_{G^H}(S)=\mathrm{vol}^H_G(S)$, we get that

$$\frac{1}{|E(H)|}\phi_{G^H}(S,\bar{S}) \le \phi^H_G(S,\bar{S}) \le \phi_{G^H}(S,\bar{S}) \quad (27)$$

by the definitions of conductance and typed-graphlet conductance. Since this result holds for any subset $S\subset V(G)$, it implies that

$$\frac{1}{|E(H)|}\cdot\phi(G^H) \le \phi^H(G) \le \phi(G^H). \quad (28)$$

■

Corollary 1. *Let λ_2 be the second smallest eigenvalue of $\mathbf{L}_{G^H}$. Then,*

$$\frac{\lambda_2}{2|E(H)|} \le \phi^H(G) \le \sqrt{2\lambda_2}.$$

Proof. Cheeger's inequality for weighted undirected graphs (see proof in [23]) gives

$$\frac{\lambda_2}{2} \le \phi(G^H) \le \sqrt{2\lambda_2}. \quad (29)$$

Using these bounds for $\phi(G^H)$ and applying them to Theorem 1, we find that

$$\frac{\lambda_2}{2|E(H)|} \le \phi^H(G) \le \sqrt{2\lambda_2}. \quad (30)$$

■

Corollary 2. *Let S be the cluster output of Algorithm 1 and let $\alpha=\phi^H_G(S,\bar{S})$ be its corresponding typed-graphlet conductance on G based on H. Then,*

$$\phi^H(G) \le \alpha \le \sqrt{4|E(H)|\phi^H(G)}.$$

Moreover, if we let λ_2 be the second smallest eigenvalue of $\mathbf{L}_{G^H}$, then

$$\phi^H(G) \le \alpha \le \beta\cdot\phi^H(G),$$

where

$$\beta = \sqrt{\frac{8}{\lambda_2}} \cdot |E(H)|,$$

showing that, for a fixed G and H, Algorithm 1 is a β-approximation algorithm to the typed-graphlet conductance minimization problem.

Proof. Clearly $\phi^H(G) \leq \alpha$ since $\phi^H(G)$ is the minimal typed-graphlet conductance. To prove the upper bound, let $(T, \bar{T})$ be the cut that achieves the minimal conductance on G^H, that is, $\phi_{G^H}(T, \bar{T}) = \phi(G^H)$. Then,

$$\alpha \leq \phi^H_G(T, \bar{T}) \tag{31}$$
$$\leq \phi_{G^H}(T, \bar{T}) \tag{32}$$
$$\leq \sqrt{2\lambda_2} \tag{33}$$
$$\leq \sqrt{4|E(H)|\phi^H(G)}. \tag{34}$$

Inequality 31 follows from the fact that α achieves the minimal typed-graphlet conductance. Inequality 32 follows from Inequality 27 in Theorem 1. Inequality 33 follows from Cheeger's inequality for weighted graphs (see [22] for a proof). Inequality 34 follows from the lower bound in Corollary 1.

We can go a bit further to express the bounds entirely in terms of $\phi^H(G)$ by noting that

$$\alpha \leq \sqrt{4|E(H)|\phi^H(G)} \tag{35}$$
$$= \sqrt{\frac{4|E(H)|}{\phi^H(G)}} \cdot \phi^H(G) \tag{36}$$
$$\leq \sqrt{\frac{8}{\lambda_2}} \cdot |E(H)| \cdot \phi^H(G) \tag{37}$$

where Inequality 37 follows from the fact that $\sqrt{\phi^H(G)} \geq \frac{\lambda_2}{2|E(H)|}$ by the lower bound of Corollary 1.

■

4 EXPERIMENTS

This section empirically investigates the effectiveness of the proposed approach quantitatively for typed-graphlet spectral clustering (Section 4.1), link prediction using the higher-order node embeddings from our approach (Section 4.2) and the typed-graphlet spectral ordering for graph compression (Section 4.3). Unless otherwise mentioned, we use all 3 and 4-node graphlets.

4.1 Clustering

We quantitatively evaluate the proposed approach by comparing it against a wide range of state-of-the-art community detection methods on multiple heterogeneous graphs from a variety of application domains with fundamentally different structural properties [64].

- **Densest Subgraph (DS-H)** [44]: This baseline finds an approximation of the densest subgraph in G using degeneracy ordering [29, 63]. Given a graph G with n nodes, let H_i be the subgraph induced by i nodes. At the start, $i = n$ and thus $H_i = G$. At each step, node v_i with smallest degree is selected from H_i and removed to obtain H_{i-1}. Afterwards, we update the corresponding degrees of H_{i-1} and density $\rho(H_{i-1})$. This is repeated to obtain $H_n, H_{n-1}, \ldots, H_1$. From $H_n, H_{n-1}, \ldots, H_1$, we select the subgraph H_k with maximum density $\rho(H_k)$.
- **KCore Communities (KCore-H)** [68, 79]: Many have observed the maximum k-core subgraph of a real-world network to be a highly dense subgraph that often contains the maximum clique [68]. The KCore baseline simply uses the maximum k-core subgraph as S and $\bar{S} = V \setminus S$.
- **Label Propagation (LP-H)** [62]: Label propagation takes a labeling of the graph, then for each node, the node label is updated according to the label with maximal frequency among its neighbors. This is repeated until the node labeling does not change. The final labeling induces a clustering of the graph and the cluster with maximum modularity is selected.
- **Louvain (Louv-H)** [15]: Louvain performs a greedy optimization of modularity by forming small, locally optimal communities then grouping each community into one node. It iterates over this two-phase process until modularity cannot be maximized locally. The community with maximum modularity is selected.
- **Spectral Clustering (Spec-H)** [23]: This baseline executes spectral clustering on the normalized Laplacian of the adjacency matrix to greedily build the sweeping cluster that minimizes conductance.
- **Untyped-Graphlet Spec. Clustering (GSpec-H)** [14]: This baseline computes the untyped-graphlet adjacency matrix and executes spectral clustering on the normalized Laplacian of this

Table 2: Network properties and statistics. Note $|\mathcal{T}_V|$ = # of node types. Comparing the number of unique typed motifs that occur for each induced subgraph (*e.g.*, there are 3 different typed 3-path graphlets that appear in yahoo).

Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{T}_V\rvert$								
yahoo-msg	100.1k	739.8k	2	3	2	3	4	3	3	3	2
dbpedia	495.9k	921.7k	4	8	0	6	10	5	0	0	0
digg	283.2k	4.6M	2	4	3	4	5	4	4	4	2
movielens	28.1k	170.4k	3	7	1	6	9	6	3	3	0
citeulike	907.8k	1.4M	3	5	0	3	6	3	0	0	0
fb-CMU	6.6k	250k	3	10	10	15	15	15	15	15	15
reality	6.8k	7.7k	2	4	3	4	5	4	4	4	2
gene	1.1k	1.7k	2	4	4	5	5	5	5	5	5
citeseer	3.3k	4.5k	6	56	40	124	119	66	98	56	19
cora	2.7k	5.3k	7	82	49	202	190	76	157	73	19
webkb	262	459	5	31	21	59	59	23	51	32	8
pol-retweet	18.5k	48.1k	2	4	4	5	5	5	5	5	4
web-spam	9.1k	465k	3	10	10	15	15	15	15	15	15
fb-relationship	7.3k	44.9k	6	50	47	112	109	85	106	89	77
Enzymes-g123	90	127	2	4	3	5	5	5	4	3	0
Enzymes-g279	60	107	2	4	4	5	5	5	5	5	0
Enzymes-g293	96	109	2	4	1	5	5	1	2	1	0
Enzymes-g296	125	141	2	4	1	4	5	2	1	1	0
NCI109-g4008	90	105	2	3	0	3	3	0	0	0	0
NCI109-g1709	102	106	3	5	0	5	5	1	0	0	0
NCI109-g3713	111	119	3	4	0	6	4	0	0	0	0
NCI1-g3700	111	119	3	4	0	6	4	0	0	0	0

Table 3: Quantitative evaluation of the methods (external conductance [8]). Note TGS is the approach proposed in this work. The best result for each graph is bold.

	DS-H	KCore-H	LP-H	Louv-H	Spec-H	GSpec-H	TGS
yahoo-msg	0.5697	0.6624	0.2339	0.3288	0.0716	0.2000	**0.0588**
dbpedia	0.7414	0.5586	0.4502	0.8252	0.9714	0.9404	**0.0249**
digg	0.4122	0.4443	0.7555	0.3232	0.0006	**0.0004**	**0.0004**
movielens	0.9048	0.9659	0.7681	0.8620	0.9999	0.6009	**0.5000**
citeulike	0.9898	0.9963	0.9620	0.8634	0.9982	0.9969	**0.7159**
fb-CMU	0.6738	0.9546	0.9905	0.8761	0.5724	0.8571	**0.5000**
reality	0.7619	0.3135	0.2322	0.1594	0.6027	0.0164	**0.0080**
gene	0.8108	0.9298	0.9151	0.8342	0.4201	0.1667	**0.1429**
citeseer	0.5000	0.6667	0.6800	0.6220	0.0526	0.0526	**0.0333**
cora	0.0800	0.9057	0.8611	0.8178	0.0870	0.0870	**0.0500**
webkb	0.2222	0.9286	0.6154	0.8646	0.6667	0.3333	**0.2222**
pol-retweet	0.5686	0.6492	0.0291	0.0918	0.6676	0.0421	**0.0220**
web-spam	0.8551	0.9331	0.9844	0.7382	0.9918	0.5312	**0.5015**
fb-relationship	0.6249	0.9948	0.5390	0.8392	0.9999	0.5866	**0.4972**
Enzymes-123	0.8667	0.8889	0.5696	0.6364	0.6768	0.5204	**0.3902**
Enzymes-279	0.9999	0.4444	0.5179	0.4444	0.2929	0.3298	**0.2747**
Enzymes-293	1.0000	0.4857	0.9444	0.3793	0.7677	0.5000	**0.3023**
Enzymes-296	1.0000	0.7073	0.9286	0.7344	0.6406	0.5000	**0.3212**
NCI109-4008	0.7619	0.4324	0.8462	0.8235	0.3500	0.4556	**0.3204**
NCI109-1709	0.4000	0.3171	0.1429	0.4615	0.3922	0.3654	**0.1333**
NCI109-3713	0.4074	0.3793	0.7500	0.4583	0.6667	1.0000	**0.2000**
NCI1-3700	0.4074	0.3793	0.7500	0.4583	0.3333	0.6667	**0.2500**
Avg. Rank	**4.59**	**4.77**	**4.64**	**4.32**	**4.27**	**3.27**	**1**

matrix to greedily build the sweeping cluster that minimizes the untyped-graphlet conductance.

Note that we append the original method name with **−H** to indicate that it was adapted to support community detection in arbitrary heterogeneous graphs (Figure 1) since the original methods were not designed for such graph data.

We evaluate the quality of communities using their external conductance score [8, 36]. This measure has been identified as one of the most important cut-based measures in a seminal survey by Schaeffer [75] and extensively studied in many disciplines and applications [8, 23, 36, 42, 75, 78, 88]. Results are reported in Table 3. As an aside, all methods take as input the same heterogeneous graph G. Overall, the results in Table 3 indicate that the proposed approach is able to reveal better high quality clusters across a wide range of heterogeneous graphs. The heterogeneous network statistics and properties including the number of unique typed motifs for each induced subgraph pattern is shown in Table 2.

We also provide the improvement (gain) achieved by TGS clustering over the other methods in Table 4. Note improvement is simply $\frac{\mathbb{E}(\mathcal{A}_i)}{\mathbb{E}(\mathcal{A}_*)}$ where $\mathbb{E}(\mathcal{A}_i)$ is the external conductance of the solution given by algorithm $\mathcal{A}_i$ and $\mathcal{A}_*$ denotes the TGS algorithm. Values less than 1 indicate that TGS performed worse than the other method whereas values > 1 indicate the improvement factor achieved by TGS. Overall, TGS achieves a mean improvement of 43.53x over all graph data and baseline methods (Table 4). Note the last column of Table 4 reports the mean improvement achieved by TGS over all methods for each graph whereas the last row reports the mean improvement achieved by TGS over all graphs for each method. Figure 2 shows how *typed graphlet conductance* (Eq. 1) changes as a function of community size $|S|$ for three different typed-graphlets.

4.2 Link Prediction in Heterogeneous Graphs

This section quantitatively demonstrates the effectiveness of TGS for link prediction.

4.2.1 Higher-order Typed-Graphlet Embeddings. In Section 4.1 we used the approach for higher-order clustering and quantitatively evaluated the quality of them. In this section, we use the approach proposed in Section 2 to derive higher-order typed-graphlet node embeddings and quantitatively evaluate them for link prediction. Algorithm 2 summarizes the method for deriving higher-order typed motif-based node embeddings (as opposed to clusters/partitions of nodes, or an ordering for compression/analysis, see Section 4.3). In particular, given a typed-graphlet adjacency matrix, Algorithm 2 outputs a $N \times D$ matrix $\mathbf{Z}$ of node embeddings. For graphs with many connected components, Algorithm 2 is called for each connected component of G^H and the resulting embeddings

Table 4: Gain/loss achieved by TGS over the other methods. Overall, TGS achieves a mean improvement of 43.53x over all graph data and baseline methods. Note the last column reports the mean improvement achieved by TGS over all methods for each graph whereas the last row reports the mean improvement achieved by TGS over all graphs for each method.

	DS	KC	LP	Louv	Spec	GSpec	Mean Gain
yahoo-msg	9.69x	11.27x	3.98x	5.59x	1.22x	3.40x	**5.86x**
dbpedia	29.78x	22.43x	18.08x	33.14x	39.01x	37.77x	**30.03x**
digg	1030x	1110x	1888x	808x	1.50x	1.00x	**806.75x**
movielens	1.81x	1.93x	1.54x	1.72x	2.00x	1.20x	**1.70x**
citeulike	1.38x	1.39x	1.34x	1.21x	1.39x	1.39x	**1.35x**
fb-CMU	1.35x	1.91x	1.98x	1.75x	1.14x	1.71x	**1.64x**
reality	95.24x	39.19x	29.02x	19.92x	75.34x	2.05x	**43.46x**
gene	5.67x	6.51x	6.40x	5.84x	2.94x	1.17x	**4.75x**
citeseer	15.02x	20.02x	20.42x	18.68x	1.58x	1.58x	**12.88x**
cora	10.00x	13.33x	17.22x	16.36x	1.74x	1.74x	**10.07x**
webkb	1.00x	4.18x	2.77x	3.89x	3.00x	1.50x	**2.72x**
pol-retweet	25.85x	29.51x	1.32x	4.17x	30.35x	1.91x	**15.52x**
webkb-spam	1.71x	1.86x	1.96x	1.47x	1.98x	1.06x	**1.67x**
fb-relationship	1.26x	2.00x	1.08x	1.69x	2.01x	1.18x	**1.54x**
Enzymes-g123	2.22x	2.28x	1.46x	1.63x	1.73x	1.33x	**1.78x**
Enzymes-g279	3.64x	1.62x	1.89x	1.62x	1.07x	1.20x	**1.84x**
Enzymes-g293	3.31x	1.61x	3.12x	1.25x	2.54x	1.65x	**2.25x**
Enzymes-g296	3.11x	2.20x	2.89x	2.29x	1.99x	1.56x	**2.34x**
NCI109-g4008	2.38x	1.35x	2.64x	2.57x	1.09x	1.42x	**1.91x**
NCI109-g1709	3.00x	2.38x	1.07x	3.46x	2.94x	2.74x	**2.60x**
NCI109-g3713	2.04x	1.90x	3.75x	2.29x	3.33x	5.00x	**3.05x**
NCI1-g3700	1.63x	1.52x	3.00x	1.83x	1.33x	2.67x	**2.00x**
Mean Gain	56.89x	58.23x	91.62x	42.74x	8.24x	3.47x	**(43.53x)**

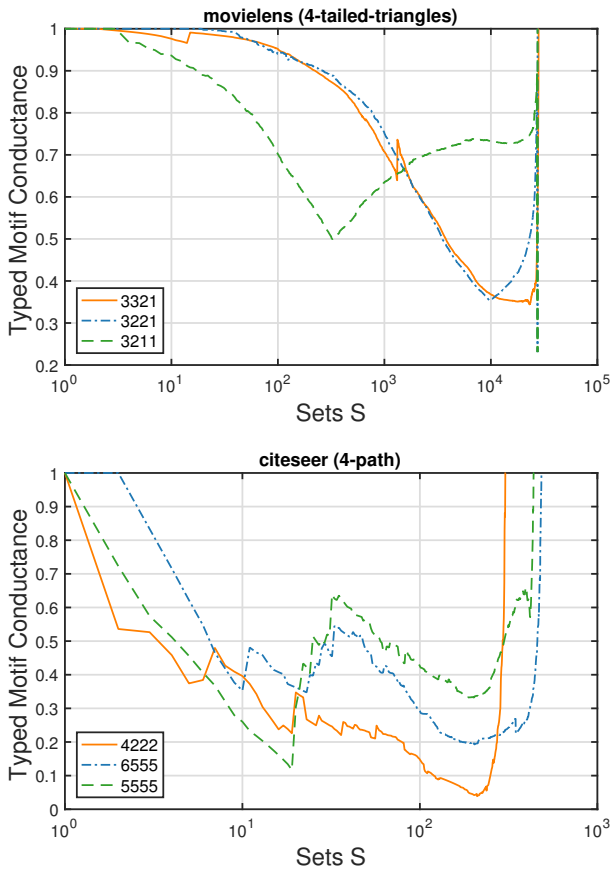


Figure 2: Typed graphlet conductance as a function of S from the sweep in Algorithm 1 for a variety of typed 4-path graphlets and typed 4-tailed-triangle graphlets. For this experiment, we consider the largest connected component for the graph derived from each typed graphlet.

are stored in the appropriate locations in the overall embedding matrix $\mathbf{Z}$.

Table 5: Link prediction edge types and semantics. We bold the edge type that is predicted by the models.

Graph	$\lvert\mathcal{T}_V\rvert$	Heterogeneous Edge Types
movielens	3	**user-by-movie**, user-by-tag tag-by-movie
dbpedia	4	**person-by-work** (produced work), person-has-occupation, work-by-genre (work-associated-with-genre)
yahoo-msg	2	**user-by-user** (communicated with), user-by-location (communication location)

4.2.2 Experimental Setup. We evaluate the higher-order typed-graphlet node embedding approach (Algorithm 2) against the following methods: DeepWalk (DW) [60], LINE [84], GraRep [20], spectral embedding (untyped edge motif) [58], and spectral embedding using untyped-graphlets. All methods output (D=128)-dimensional node embeddings $\mathbf{Z} = \left[\, \mathbf{z}_1 \cdots \mathbf{z}_n \,\right]^T$ where $\mathbf{z}_i \in \mathbb{R}^D$. For DeepWalk (DW) [60], we perform 10 random walks per node of length 80 as mentioned in [38]. For LINE [84], we use 2nd-order proximity and perform 60 million samples. For GraRep (GR) [20], we use $K = 2$. In contrast, the spectral embedding methods do not have any hyper-parameters besides D which is fixed for all methods. As an aside, all methods used for comparison were modified to support heterogeneous graphs (similar to how the other baseline methods from Section 4.1 were modified). In particular, we adapted the methods to allow multiple graphs as input consisting of homogeneous or bipartite graphs that all share at least one node type (See Table 5 and Figure 1) and from these graphs we construct a single large graph by simply ignoring the node and edge types and relabeling the nodes to avoid conflicts.

4.2.3 Comparison. Given a partially observed graph G with a fraction of missing/unobserved edges, the link prediction task is to predict these missing edges. We generate a labeled dataset of edges. Positive examples are obtained by removing 50% of edges uniformly at random, whereas *negative examples* are generated by randomly sampling an equal number of node pairs $(i, j) \notin E$. For each method, we learn embeddings using the remaining graph. Using the embeddings from each method, we then learn a logistic regression (LR) model to predict whether a given edge in the test set exists in E or not. Experiments are repeated for 10 random seed initializations and the average performance is reported. All methods are evaluated against four different evaluation metrics including F_1, Precision, Recall, and AUC.

Table 6: Link prediction results.

		DW	LINE	GR	Spec	GSpec	TGS
movielens	$\mathbf{F_1}$	0.8544	0.8638	0.8550	0.8774	0.8728	**0.9409**
	Prec.	0.9136	0.8785	0.9235	0.9409	0.9454	**0.9747**
	Recall	0.7844	0.8444	0.7760	0.8066	0.7930	**0.9055**
	AUC	0.9406	0.9313	0.9310	0.9515	0.9564	**0.9900**
dbpedia	$\mathbf{F_1}$	0.8414	0.7242	0.7136	0.8366	0.8768	**0.9640**
	Prec.	0.8215	0.7754	0.7060	0.7703	0.8209	**0.9555**
	Recall	0.8726	0.6375	0.7323	0.9669	0.9665	**0.9733**
	AUC	0.8852	0.8122	0.7375	0.9222	0.9414	**0.9894**
yahoo	$\mathbf{F_1}$	0.6927	0.6269	0.6949	0.9140	0.8410	**0.9303**
	Prec.	0.7391	0.6360	0.7263	0.9346	0.8226	**0.9432**
	Recall	0.5956	0.5933	0.6300	0.8904	0.8699	**0.9158**
	AUC	0.7715	0.6745	0.7551	0.9709	0.9272	**0.9827**

⋆ Note DW=DeepWalk and GR=GraRep.

For link prediction [5, 50], entity resolution/network alignment, recommendation and other machine learning tasks that require edge embeddings (features) [73], we derive edge embedding vectors by combining the learned node embedding vectors of the corresponding nodes using an edge embedding function Φ. More formally, given D-dimensional embedding vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ for node i and j, we derive a D-dimensional edge embedding vector $\mathbf{z}_{ij} = \Phi(\mathbf{z}_i, \mathbf{z}_j)$ where Φ is defined as one of the following *edge*

embedding functions:

$$\Phi \in \left\{ \frac{\mathbf{z}_i + \mathbf{z}_j}{2},\ \mathbf{z}_i \odot \mathbf{z}_j,\ \left|\mathbf{z}_i - \mathbf{z}_j\right|,\ (\mathbf{z}_i - \mathbf{z}_j)^{\circ 2},\ \max(\mathbf{z}_i, \mathbf{z}_j),\ \mathbf{z}_i + \mathbf{z}_j \right\}$$

Note $\mathbf{z}_i \odot \mathbf{z}_j$ is the element-wise (Hadamard) product, $\mathbf{z}^{\circ 2}$ is the Hadamard power, and $\max(\mathbf{z}_i, \mathbf{z}_j)$ is the element-wise max.

Table 5 summarizes the heterogeneous network data used for link prediction. In particular, the types used in each of the heterogeneous networks are shown in Table 5 as well as the specific types involved in the edges that are predicted (*e.g.*, the edge type being predicted). The results are provided in Table 6. Results are shown for the best edge embedding function. In Table 6, TGS is shown to outperform all other methods across *all* four evaluation metrics. In all cases, the higher-order typed-graphlet spectral embedding outperforms the other methods (Table 6) with an overall mean gain (improvement) in F_1 of 18.7% (and up to 48.4% improvement) across all graph data. In terms of AUC, TGS achieves a mean gain of 14.4% (and up to 45.7% improvement) over all methods. We posit that an approach similar to the one proposed in [66] could be used with the typed-graphlet node embeddings to achieve even better predictive performance. This approach would allow us to leverage multiple typed-graphlet Laplacian matrices for learning more appropriate higher-order node embeddings.

4.3 Graph Compression

In Section 4.1 we used the approach for higher-order clustering whereas Section 4.2 demonstrated the effectiveness of the approach for link prediction. However, the framework can be leveraged for many other important applications including graph compression [16, 17, 21, 49, 71]. In this section, we explore the proposed approach for graph compression. Compression has two key benefits. First, it reduces the amount of IO traffic [71]. Second, it can speed up existing algorithms by reducing the amount of work required [43]. Graph compression methods rely on a "good" ordering of the vertices in the graph to achieve a good compression [16, 17].

Table 7: Graph compression results. Size in bytes required to store heterogeneous graphs using the bvgraph compression scheme with different orderings.

	Bytes				
Graph	**Native**	**Spec**	**GSpec**	**TGS**	Gain
movielens	585588	471246	464904	444252	**14.18%**
yahoo-msg	3065499	2694151	2708700	2427325	**16.29%**
dbpedia	4800584	3520721	3469878	3111728	**26.31%**
digg	15989475	10462874	10296144	9677741	**26.57%**

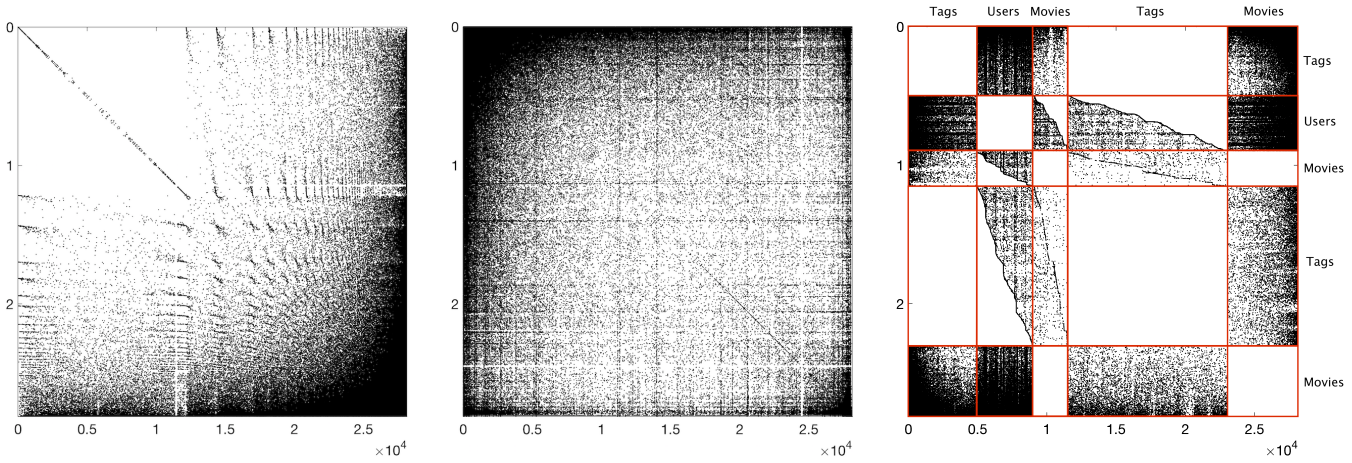
In this work, we order the vertices by the *typed-graphlet spectral ordering* introduced previously in Definition 11. Notice in this case, the output of our approach is the typed-graphlet spectral ordering (Definition 11) as opposed to clusters (Section 4.1) or node embeddings (Section 4.2). We then evaluate how well the bvgraph [17] compression method reduces the graph size using this ordering. Given an ordering, we permute the graph to use this ordering and use the bvgraph compression algorithm [17] with all the default settings to compress the networks. Results are reported in Table 7 for four large heterogeneous graphs. We compare the compression obtained by reporting the size of each heterogeneous graph in bytes after compression. We evaluate four orderings of the vertices: the native order, spectral ordering (untyped edge), untyped-graphlet ordering and the typed-graphlet spectral ordering proposed in this work. For untyped-graphlet and typed-graphlet spectral ordering we report the best result given by an ordering from any untyped or typed-graphlet. We find that the typed-graphlet ordering results in better compression across all other methods and graphs. Overall, typed-graphlet spectral ordering achieves a mean improvement of 20.8% over all graphs and all orderings.

In addition to the quantitative compression results shown in Table 7, we use the proposed ordering for exploratory analysis. In particular, we use the orderings to permute the rows/columns of the original adjacency matrix and visualize the nonzero structure of the resulting matrices in Figure 3. Using the ordering from TGS (Definition 11) gives rise to partitions (sub-matrices) that are significantly more homogeneous (completely connected or disconnected) than the other methods as shown in Figure 3 and thus are able to achieve a better compression as shown quantitatively in Table 7. Furthermore, TGS is able to uncover the type of the nodes by grouping nodes into partitions based on their types (users, movies, tags). Moreover, the partitions are also meaningful as they partition movies and the tags used to describe those movies into genres. This allows us to understand the tags that best describe that genre as well as the movies from that genre that align with those tags. Other typed-graphlet spectral orderings from different typed graphlets were removed due to space, though many of them also gave interesting and explainable block partitions as well.

5 RELATED WORK

Community Detection in Homogeneous Graphs. Most research in community detection has traditionally focused on homogeneous graphs [18]. This problem has been extensively researched as evidenced by the multiple survey papers [27, 31, 33, 52, 56, 75] and empirical comparisons of algorithms [11, 39, 46, 48] on this topic. Many works have focused on community detection techniques using modularity-based optimization. Modularity was introduced in the seminal paper [57] as a quantitative measure for assigning scores to a community structure. It became a standard measure for comparing clustering algorithms and suggested a framework for community detection as an optimization task of a quality function. Modularity maximization is an NP-hard problem, but there exist efficient heuristics such as greedy methods, semidefinite programming, simulated annealing, and spectral methods [56]. Nevertheless, it suffers from many drawbacks such as runtime dependence on the size of graph [33], resolution limit [32], and nonuse of between-community connectivity information [56].

Graph conductance is another very popular community quality function [23, 42]. Computing the conductance of a graph is an NP-hard problem as well [80], but there exist spectral methods that give good, *theoretically-supported* approximations [23, 42, 47, 58, 87] and have only weak runtime dependence on the size of the graph since there exist fast methods for computing eigenvalues [37]. Moreover,



(a) Spectral clustering (untyped edge) (b) Untyped 3-path (c) Typed 4-cycle

Figure 3: Typed graphlet-based spectral ordering achieves significant compression by partitioning users, tags, and movies (from the movielens data) into homogeneous groups that are either nearly fully-connected (near-clique) or fully-disconnected. Strikingly, TGS partitions the rows/columns according to types without explicitly leveraging types (*i.e.*, types are not used when deriving the typed-graphlet spectral ordering). For instance, the first ≈5k rows/columns correspond to tags, whereas the following ≈4k rows/columns are users, and so on. This is in contrast to the other methods where the rows/columns of different types are mixed with one another in a seemingly random fashion. Moreover, these approaches fail to partition the graph into homogeneous groups that are dense or completely empty. The typed 4-cycle graphlet used above consists of 2 nodes representing movies and the other two representing tags assigned to the movies. Other typed-graphlets gave other interesting results with comparable compression.

conductance takes into account the internal and external connectedness of a community [33]. As an aside, existing higher-order clustering methods that extend modularity [10] and conductance [14, 85] are all designed for *homogeneous graphs* with a single node/edge type *and* are also based on the existing notion of *untyped graphlets*. However, we discuss these methods along side other higher-order methods that leverage untyped graphlets. In this work, we propose a higher-order clustering framework that generalizes to heterogeneous graphs. However, since homogeneous graphs are a special case of heterogeneous graphs (where nodes/edges have a single type), the proposed framework can be used for higher-order clustering in homogeneous graphs as well.

Community Detection in Heterogeneous Graphs. Recently, researchers have started to extend community detection methods for multi-relational, multi-typed graphs [18, 77]. In the literature, these graphs are referred to as *heterogeneous graphs* or *heterogeneous information networks* [77]. In recent years, many methods have been proposed for community detection in heterogeneous networks in ways that consolidate both structural and compositional information. Weight modification methods reduce a heterogeneous graph to a weighted homogeneous graph through an edge-weighting function based on node types. Afterwards, any homogeneous community detection algorithm can be applied to this modified graph. For example, [55] and [82] use a matching coefficient function that quantifies the number of similar node types. Under a certain viewpoint, our method can be classified under this category of algorithms. We discuss this later.

A different paradigm for combining both structural and compositional information of a network would be to take the opposite approach of weight modification. One could transform a heterogeneous graph to a point cloud by converting structural information coupled with node type data into a node distance function [77]. Then, any distance-based clustering method such as k-means can be applied on this point cloud. This approach incorporates both structure and composition of the network. Linear combination methods take a linear combination of type similarity and structural similarity functions as proposed in [25]. Walk strategies on heterogeneous graphs have also been used to compute vertex distance functions. The work in [90] defines a random walk on a heterogeneous graph such that more paths—alongside the paths from the network structure alone—exist between nodes of the same type, thus measuring vertex proximity with two modes of data. Another distance function based on breadth-first search is proposed in [35] that uses the node types to determine the next visited node, and thus the distance.

These similarity reduction methods have a feature that nodes that are structurally far from each other but share similar attributes may become close after this modification [18]. As a consequence, clusters may contain disconnected portions of the graph which is generally not seen as a characteristic of communities. Using motif-based clustering in our work allows us to preserve this connectedness property. Another standard homogeneous clustering approach that has been extended to heterogeneous graphs is statistical inference such as generative models [51], stochastic block models [12], and Bayesian inference [89].

In this work, we develop a principled framework for *higher-order clustering in heterogeneous graphs*. Furthermore, while most existing methods for heterogeneous graphs lack a sound theoretically grounded framework, we rigorously prove mathematical guarantees on the optimality of the higher-order clustering obtained from the framework.

Graphlets. Graphlets (network motifs) were first introduced in [54, 76] to study the structural design principles of single-typed biological networks. In that work, graphlets were found to be the fundamental building blocks of complex homogeneous (single-typed) networks. Various algorithms have been developed to count the occurrences of all graphlets up to a given size on the nodes [53] and edges [3, 4] of a graph. Motif discovery algorithms are limited in that they are computationally expensive for larger motifs and they search for motifs operating in isolation. In [45], it was shown that network context, *i.e.*, the connections of the motif to the rest of the network, is important in inferring the functionality of a motif. Motifs have recently been used in other higher order-network analysis methods such as role discovery [65], network embeddings [67], inductive network representation learning [73], and temporal network analysis [59].

Motifs were first used for community detection in [10]. In that work, motif modularity was introduced as a generalization of the standard notion of modularity. Once motif modularity is defined, their method essentially becomes a modularity maximization problem. As mentioned above, modularity-based methods ignore any between-community connectivity information. Therefore, the connectivity of the communities to the rest of the network is not considered in the motif-based modularity method, and thus we lose information that may be of value in correctly capturing useful community structure. Moreover, this method still suffers from the resolution limit [32] and requires longer computation. More recently, [14, 85] extended the definition of graph conductance based on the existing notion of untyped-motifs for homogeneous graphs. This definition is a special case of the proposed framework when untyped graphlets are used and the graph is homogeneous.

Previous work has focused entirely on *untyped motifs/graphlets* [3, 4, 6, 74]. In this work, we introduce the generalized notion of *typed graphlets* and use this more powerful representation as a basis for higher-order clustering. Typed graphlets generalize the notion of graphlets to rich heterogeneous networks as they explicitly capture the higher-order typed connectivity patterns in such networks. Using this more appropriate and general notion, we develop a principled general higher-order clustering framework by introducing *typed-graphlet conductance* that generalizes the traditional conductance to higher-order structures in heterogeneous graphs. Recall that homogeneous, labeled, signed, and attributed graphs are all special cases of heterogeneous graphs. The framework provides mathematical guarantees on the optimality of the higher-order clustering obtained. The theoretical results extend to typed graphlets of arbitrary size and avoids restrictive special cases required in prior work. In addition, existing work on higher-order motif-based methods have focused entirely on simple homogeneous graphs whereas our work focuses on rich heterogeneous networks with an arbitrary number of node and edge types (Figure 1). Furthermore, while previous work on higher-order clustering was designed for homogeneous graphs *and* untyped-graphlets, they also focused only on community detection whereas this work also leverages the proposed framework for deriving higher-order embeddings and graph compression based on the typed-graphlet spectral ordering.

6 CONCLUSION

This work proposed a general framework for higher-order spectral clustering in heterogeneous graphs. The framework explicitly incorporates heterogeneous higher-order information by counting typed graphlets that leverage node and edge types. It is shown that typed-graphlets generalize the notion of graphlets to rich heterogeneous networks and that these explicitly capture the higher-order typed connectivity patterns in such networks. Using these as a basis, we proposed the notion of *typed-graphlet conductance* that generalizes the notion of conductance to higher-order structures in heterogeneous graphs. Typed-graphlet conductance minimization, for a given typed graphlet, provides a cut in the heterogeneous graph that preserves instances of the typed graphlet in a balanced manner.

The framework provides mathematical guarantees on the optimality of the higher-order clustering obtained. The theoretical results extend to typed graphlets of arbitrary size and avoids restrictive special cases required in prior work. The framework unifies prior work and serves as a basis for analysis of higher-order spectral clustering methods. It was shown that spectral clustering and untyped-graphlet spectral clustering are special cases in the proposed framework. The experiments demonstrated the effectiveness and utility of the proposed framework for three important tasks including (i) clustering, (ii) predictive modeling, and (iii) graph compression. For these tasks, the approach was shown to outperform other state-of-the-art methods with a significant improvement in all cases. The approach achieves an overall improvement in F_1 and AUC of 18.7% and 14.4% for link prediction whereas for graph compression it achieves a mean improvement of 20.8% across all graphs and methods. Finally, typed-graphlet spectral clustering is shown to uncover better clusters than state-of-the-art methods with a mean improvement of 43x over all graphs and methods.

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 Ahmed, Ted Willke, and Ryan A. Rossi. 2016. Exact and Estimation of Local Edge-centric Graphlet Counts. In *KDD BigMine*. 16.
[3] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. 2015. Efficient Graphlet Counting for Large Networks. In *ICDM*. 1–10.
[4] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, Nick Duffield, and Theodore L. Willke. 2016. Graphlet Decomposition: Framework, Algorithms, and Applications. *KAIS* (2016), 689–722.
[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 *arXiv:1802.02896*.
[6] Nesreen K. Ahmed, Theodore L. Willke, and Ryan A. Rossi. 2016. Estimation of Local Subgraph Counts. In *IEEE BigData*. 586–595.
[7] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1.
[8] Hélio Almeida, Dorgival Guedes, Wagner Meira, and Mohammed J Zaki. 2011. Is there a best quality metric for graph clusters?. In *ECML/PKDD*. Springer, 44–59.
[9] Noga Alon. 1997. On the edge-expansion of graphs. *Combinatorics, Probability and Computing* 6, 2 (1997), 145–152.
[10] Alex Arenas, Alberto Fernandez, Santo Fortunato, and Sergio Gomez. 2008. Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical* 41, 22 (2008), 224001.

[11] Leon Danon Arenas, Albert Díaz-Guilera, Jordi Duch, and Alex. 2005. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment* 09 (2005), P09008.

[12] Ramnath Balasubramanyan and William W Cohen. 2011. Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In *SDM*. SIAM, 450–461.

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

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

[15] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[16] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *WWW*. 587–596.

[17] Paolo Boldi and Sebastiano Vigna. 2004. The webgraph framework I: compression techniques. In *WWW*. 595–602.

[18] Cecile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenková. 2015. Clustering attributed graphs: Models, measures and methods. *Network Science* 3, 3 (2015), 408–444.

[19] Gregory Buehrer and Kumar Chellapilla. 2008. A scalable pattern mining approach to web graph compression with communities. In *WSDM*. 95–106.

[20] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *CIKM*. 891–900.

[21] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. 2009. On compressing social networks. In *KDD*. 219–228.

[22] Fan RK Chung. 1996. Laplacians of graphs and Cheeger's inequalities. *Combinatorics, Paul Erdos is Eighty* 2, 157-172 (1996), 13–2.

[23] Fan RK Chung. 1997. *Spectral graph theory*. Number 92. Amer. Math. Soc.

[24] Fan RK Chung and Kevin Oden. 2000. Weighted graph Laplacians and isoperimetric inequalities. *Pacific J. Math.* 192, 2 (2000), 257–273.

[25] D Combe, C Largeron, E Egyed-Zsigmond, and M Géry. 2012. Combining Relations and Text in Scientific Network Clustering. In *ASONAM*. 1248–1253.

[26] Stephen A. Cook. 1971. The Complexity of Theorem-proving Procedures. In *STOC*. ACM, New York, NY, USA, 151–158.

[27] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. 2011. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining* 4, 5 (sep 2011), 512–546.

[28] I.S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *SIGKDD*. 269–274.

[29] Paul Erdős and András Hajnal. 1966. On chromatic number of graphs and set-systems. *Acta Mathematica Academiae Scientiarum Hungarica* 17, 1-2 (1966), 61–99.

[30] Pedro F Felzenszwalb and Daniel P Huttenlocher. 2004. Efficient graph-based image segmentation. *IJCV* 59, 2 (2004), 167–181.

[31] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174.

[32] Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *PNAS* 104, 1 (2007), 36–41.

[33] Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659 (2016), 1–44.

[34] Marco Gaertler. 2005. Clustering. In *Network analysis*. Springer, 178–215.

[35] Rong Ge, Martin Ester, Byron J Gao, Zengjian Hu, Binay Bhattacharya, and Boaz Ben-Moshe. 2008. Joint Cluster Analysis of Attribute Data and Relationship Data: The Connected K-center Problem, Algorithms and Applications. *TKDD* 2, 2 (jul 2008), 7:1—-7:35.

[36] David F Gleich and C Seshadhri. 2012. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *SIGKDD*. 597–605.

[37] Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3. JHU Press.

[38] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.

[39] Steve Harenberg, Gonzalo Bello, L Gjeltema, Stephen Ranshous, Jitendra Harlalka, Ramona Seay, Kanchana Padmanabhan, and Nagiza Samatova. 2014. Community detection in large-scale networks: a survey and empirical evaluation. *Wiley Interdisciplinary Reviews: Computational Statistics* 6, 6 (2014), 426–439.

[40] Bruce Hendrickson and Robert Leland. 1995. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing* 16, 2 (1995), 452–469.

[41] Shlomo Hoory, Nathan Linial, and Avi Wigderson. 2006. Expander graphs and their applications. *Bull. Amer. Math. Soc.* 43, 4 (2006), 439–561.

[42] Ravi Kannan, Santosh Vempala, and Adrian Vetta. 2004. On Clusterings: Good, Bad and Spectral. *J. ACM* 51, 3 (May 2004), 497–515.

[43] Chinmay Karande, Kumar Chellapilla, and Reid Andersen. 2009. Speeding up algorithms on compressed web graphs. *Internet Mathematics* 6, 3 (2009), 373–398.

[44] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*. Springer, 597–608.

[45] Johannes F Knabe, Chrystopher L Nehaniv, and Maria J Schilstra. 2008. Do motifs reflect evolved function? No convergent evolution of genetic regulatory network subgraph topologies. *Biosystems* 94, 1 (2008), 68–74.

[46] Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: A comparative analysis. *Physical Review E* 80, 5 (nov 2009), 56117.

[47] Tom Leighton and Satish Rao. 1999. Multicommodity Max-flow Min-cut Theorems and Their Use in Designing Approximation Algorithms. *J. ACM* 46, 6 (nov 1999), 787–832.

[48] Jure Leskovec, Kevin J Lang, and Michael Mahoney. 2010. Empirical Comparison of Algorithms for Network Community Detection. In *WWW*. New York, NY, USA, 631–640.

[49] Panagiotis Liakos, Katia Papakonstantinopoulou, and Michael Sioutis. 2014. Pushing the envelope in graph compression. In *CIKM*. 1549–1558.

[50] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *CIKM*. 556–559.

[51] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. 2009. Topic-link LDA: Joint Models of Topic and Author Community. In *ICML*. New York, NY, USA, 665–672.

[52] Fragkiskos D Malliaros and Michalis Vazirgiannis. 2013. Clustering and community detection in directed networks: A survey. *Physics Reports* 533, 4 (2013), 95–142.

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

[54] 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.

[55] Jennifer Neville, Micah Adler, and David Jensen. 2003. Clustering relational data using attribute and link information. In *IJCAI Workshop*. 9–15.

[56] M E J Newman. 2011. Communities, modules and large-scale structure in networks. *Nature Physics* 8 (dec 2011), 25.

[57] M E J Newman and M Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2 (feb 2004), 26113.

[58] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *NIPS*. 849–856.

[59] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *WSDM*. 601–610.

[60] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[61] N Pržulj, D G Corneil, and I Jurisica. 2004. Modeling interactome: scale-free or geometric? *Bioinformatics* 20, 18 (dec 2004), 3508–3515.

[62] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

[63] Ryan A. Rossi and Nesreen K. Ahmed. 2014. Coloring Large Complex Networks. *Social Network Analysis and Mining* 4, 1, Article 228 (2014), 37 pages.

[64] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. http://networkrepository.com

[65] R A Rossi and N K Ahmed. 2015. Role Discovery in Networks. *TKDE* 27, 4 (2015), 1112–1131.

[66] Ryan A. Rossi, Nesreen K. Ahmed, and Eunyee Koh. 2018. Higher-Order Network Representation Learning. In *WWW*.

[67] 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).

[68] Ryan A. Rossi, David F. Gleich, and Assefaw H. Gebremedhin. 2015. Parallel Maximum Clique Algorithms with Applications to Network Analysis. *SISC* 37, 5 (2015), 28.

[69] Ryan A. Rossi and Rong Zhou. 2015. Scalable Relational Learning for Large Heterogeneous Networks. In *DSAA*. 1–10.

[70] Ryan A. Rossi and Rong Zhou. 2016. Parallel Collective Factorization for Modeling Large Heterogeneous Networks. In *SNAM*. 30.

[71] Ryan A. Rossi and Rong Zhou. 2018. GraphZIP: A Clique-based Sparse Graph Compression Method. *Journal of Big Data* 5, 1 (2018), 14.

[72] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Estimation of Graphlet Statistics. In *arXiv:1701.01772v1*. 1–14.

[73] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *WWW BigNet*. 8.

[74] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Estimation of Graphlet Counts in Massive Networks. In *TNNLS*. 1–14.

[75] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.

[76] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. 2002. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics* 31 (Apr 2002), 64.

[77] C Shi, Y Li, J Zhang, Y Sun, and P S Yu. 2017. A Survey of Heterogeneous Information Network Analysis. *TKDE* 29, 1 (2017), 17–37.

[78] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *TPAMI* 22, 8 (2000), 888–905.
[79] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2016. CoreScope: Graph Mining Using k-Core Analysis–Patterns, Anomalies and Algorithms. In *ICDM*. 469–478.
[80] Jiří Šíma and Satu Elisa Schaeffer. 2006. On the NP-completeness of some graph cluster measures. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 530–537.
[81] Horst D Simon. 1991. Partitioning of unstructured problems for parallel processing. *Computing systems in engineering* 2, 2 (1991), 135–148.
[82] Karsten Steinhaeuser and Nitesh V Chawla. 2008. Community Detection in a Large Real-World Social Network BT - Social Computing, Behavioral Modeling, and Prediction, Huan Liu, John J Salerno, and Michael J Young (Eds.). Springer US, Boston, MA, 168–175.
[83] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explorations* 14, 2 (2013), 20–28.
[84] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding.. In *WWW*.
[85] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable Motif-aware Graph Clustering. In *WWW*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1451–1460.
[86] Rafael Van Driessche and Dirk Roose. 1995. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel computing* 21, 1 (1995), 29–48.
[87] Deepak Verma and Marina Meila. 2003. A comparison of spectral clustering algorithms. *University of Washington Tech Rep UWCSE030501* 1 (2003), 1–18.
[88] Konstantin Voevodski, Shang-Hua Teng, and Yu Xia. 2009. Finding local communities in protein networks. *BMC bioinformatics* 10, 1 (2009), 297.
[89] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A Model-based Approach to Attributed Graph Clustering. In *SIGMOD (SIGMOD '12)*. ACM, New York, NY, USA, 505–516.
[90] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph Clustering Based on Structural/Attribute Similarities. *VLDB* 2, 1 (aug 2009), 718–729.