Higher-Order Label Homogeneity and Spreading in Graphs

Dhivya Eswaran
deswaran@cs.cmu.edu
Carnegie Mellon University

Srijan Kumar
srijan@gatech.edu
Georgia Institute of Technology

Christos Faloutsos
christos@cs.cmu.edu
Carnegie Mellon University

ABSTRACT

Do higher-order network structures aid graph semi-supervised learning? Given a graph and a few labeled vertices, labeling the remaining vertices is a high-impact problem with applications in several tasks, such as recommender systems, fraud detection and protein identification. However, traditional methods rely on edges for spreading labels, which is limited as all edges are not equal. Vertices with stronger connections participate in higher-order structures in graphs, which calls for methods that can leverage these structures in the semi-supervised learning tasks.

To this end, we propose Higher-Order Label Spreading (HOLS) to spread labels using higher-order structures. HOLS has strong theoretical guarantees and reduces to standard label spreading in the base case. Via extensive experiments, we show that higher-order label spreading using triangles in addition to edges is up to 4.7% better than label spreading using edges alone. Compared to prior traditional and state-of-the-art methods, the proposed method leads to statistically significant accuracy gains in all-but-one cases, while remaining fast and scalable to large graphs.

1 INTRODUCTION

Given an undirected unweighted graph and a few labeled vertices, the graph transductive learning or semi-supervised learning (SSL) aims to infer the labels for the remaining unlabeled vertices [1, 6, 17, 30, 36, 37, 39, 40]. Graph SSL finds applications in a number of settings: in a social network, we can infer a particular characteristic (e.g. political leaning) of a user based on the information of her friends to produce tailored recommendations; in a user-product bipartite rating network, based on a few manually identified fraudulent user accounts, SSL is useful to spot other fraudulent accounts [4, 10, 18, 19]; SSL can identify protein functions from networks of their physical interaction using just a few labels [32].

Traditional graph SSL algorithms leverage a key property of real-world networks: *the homophily of vertices* [5, 21], i.e., the nearby vertices in a graph are likely to have the same label. However, these methods tend to be limited by the fact that all the neighbors of a vertex are not equal. Consider your own friendship network where you have many acquaintances, but only a few close friends. In fact,
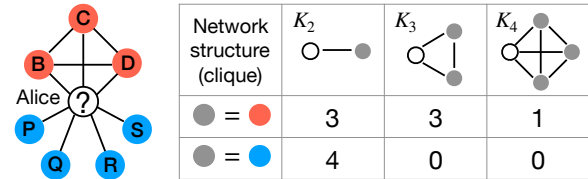
Figure 1: Graph SSL approaches which take only edges into account incorrectly classify the unlabeled central vertex 'Alice' as blue. By leveraging higher-order network structures, the proposed HOLS correctly labels Alice as red.

prior research has shown that vertices with a strong connection participate in several higher-order structures, such as dense sub-graphs and cliques [12–14, 27]. Thus, leveraging the higher-order structure between vertices is crucial to accurately label the vertices.

Let us elaborate this using a small friendship network example, shown in Figure 1. The central vertex, Alice, participates in a closely-knit community with three friends B, C, and D, all of whom know each other. In addition, she has four acquaintances P, Q, R, and S from different walks of her life. Let the vertices be labeled by their ideological beliefs—vertices B, C, and D have the same blue label; and the rest of the vertices have the red label. Even though Alice has more red connections than blue, the connection between Alice, B, C, and D is stronger as Alice participates in three 3-cliques and one 4-clique with them. In contrast, Alice has no 3- and 4- cliques with P, Q, R, and S. Owing to the stronger connection with the red nodes, Alice should be labeled red as well. However, traditional graph SSL techniques that rely on edges alone label Alice as blue [39, 40]. This calls for graph SSL methods that look beyond edges to leverage the signal present in higher-order structures to label vertices.

Our present work focuses on three key research questions:

- **RQ1.** How do the data reveal that higher-order network structures are homogeneous in labels?
- **RQ2.** How can we leverage higher-order network structures for graph SSL in a principled manner?
- **RQ3.** Do higher-order structures help improve graph SSL?

Accordingly, our contributions can be summarized as follows:
(i) Analysis: Through an empirical analysis of four diverse real-world networks, we demonstrate the phenomenon of *higher-order label homogeneity*, i.e., the tendency of vertices participating in a higher-order structure (e.g. triangle) to share the same label.
(ii) Algorithm: We develop Higher-Order Label Spreading (HOLS) to leverage higher-order structures during graph semi-supervised learning. HOLS works for any user-inputted higher-order structure and in the base case, is equivalent to edge-based label spreading [39].
(iii) Effectiveness: We show that label spreading via higher-order structures strictly outperforms label spreading via edges by up to 4.7% statistically significant margin. Notably, HOLS is competitive with recent deep learning based methods, while running 15× faster.

Table 1: Qualitative comparison of HOLS-3 (using edges and triangles) with traditional and recent graph SSL approaches

Desiderata	LP [40]	LS [39]	BP [37]	Planetoid [36]	GCN [17]	MixHop [1]	HOLS-3
Higher-order structures				?	?	?	✓
Theoretical guarantees	✓	✓	?				✓
Fast algorithm	✓	✓	✓				✓

For reproducibility, all the code and datasets are available at https://github.com/dhivyaeswaran/hols.

2 RELATED WORK

Traditional graph SSL approaches: By far, the most widely used graph SSL techniques are label propagation [40] and label spreading [39]. Label propagation (LP) clamps labeled vertices to their provided values and uses a graph Laplacian regularization, while label spreading (LS) uses a squared Euclidean penalty as supervised loss and *normalized* graph Laplacian regularization which is known to be better-behaved and more robust to noise [33]. Both these techniques permit closed-form solution and are extremely fast in practice, scaling well to billion-scale graphs. Consequently, a number of techniques build on top of these approaches, for example, to allow inductive generalization [6, 34] and to incorporate certainty [30]. When the graph is viewed as pairwise Markov random field, belief propagation (BP) [37] may be used to recover the exact marginals on the vertices. BP can handle network effects beyond just homophily; however, it has well-known convergence problems from a practitioner's point of view [26]. While traditional techniques, in general, show many desirable theoretical properties such as closed-form solution, convergence guarantees, connections to spectral graph theory [40] and statistical physics [37], as such, they do not account for higher-order network structures.

Recent graph SSL approaches differ from traditional SSL methods in training embeddings of vertices to jointly predict labels as well as the neighborhood context in the graph. Specifically, Planetoid [36] uses skipgrams, while GCN [17] uses approximate spectral convolutions to incorporate neighborhood information. MixHop [1] can learn a general class of neighborhood mixing functions for graph SSL. As such, these do not incorporate *specific higher-order structures* provided by the user. Further, their performance in practice tends to be limited by the availability of 'good' vertex features for initializing the optimization procedure.

Hybrid approaches for graph SSL: Another way to tackle the graph SSL problem is a hybrid approach to first extract vertex embeddings using an unsupervised approach such as node2vec [11], DeepWalk [23] or LINE [31] and then use the available labels to learn a transductive classifier such as an SVM [16]. Such methods, however, neither have well-understood theoretical properties nor do they optimize for a single objective in an end-to-end manner.

Comparison: We compare the best performing HOLS algorithm (HOLS-3 which uses triangles in addition to edges) *qualitatively* to

Table 2: Statistics of datasets used

Dataset	Domain	$|\mathcal{V}|$	$|\mathcal{E}|$	C
EuEmail [20]	Email communication	1005	16.0K	42
PolBlogs [3]	Blog hyperlinks	1224	16.7K	2
Cora [28]	Article citations	23.1K	89.1K	10
Pokec [29]	Friendship	1.6M	22.3M	10

prominent SSL approaches in Table 1 and *quantitatively* via experiments to representative methods from the above categories: LP and LS (traditional), GCN (recent) and node2vec + TSVM (hybrid).

3 HIGHER-ORDER LABEL HOMOGENEITY

Recent work has shown that graphs from diverse domains have many striking higher-order network structures [8] which can be leveraged to improve tasks such as clustering [38], link prediction [2, 7] and ranking [24]. In this section, we motivate the need to consider such structures for semi-supervised learning through empirical analysis of four diverse large real-world networks. We define and quantify *higher-order label homogeneity*–i.e., the tendency of vertices participating in higher-order structures to share the same label. We will show that the higher-order label homogeneity is remarkably more common than expected in real-world graphs.

3.1 Dataset Description

We use four network datasets for our empirical analysis and experiments. Table 2 summarizes some important dataset statistics.

- **EuEmail [20]** is an e-mail communication network from a large European research institution. Vertices indicate members of the institution and an edge between a pair of members indicates that they exchanged at least one email. Vertex labels indicate membership to one of the 42 departments.
- **PolBlogs [3]** is a network of hyperlinks between blogs about US politics during the period preceding the 2004 presidential election. Blogs are labeled as right-leaning or left-leaning.
- **Cora [28]** is a citation network among papers published at computer science conferences. Vertex labels indicate one of 10 areas (e.g. Artificial Intelligence, Databases, Networking) that the paper belongs to based on its venue of publication.
- **Pokec [29]** is the most popular online social network in Slovakia. Vertices indicate users and edges indicate friendships. From the furnished user profile information, we extract the locality or 'kraj' that users belong to and use them as labels.

These datasets exhibit homophily [5, 21]: people typically e-mail others within the same department; blogs tend to link to others having the same political leaning; papers mostly cite those from the same area; people belonging to the same locality are more likely to meet and become friends. In all cases, we omit self-loops and take the edges as undirected and unweighted.

3.2 Empirical Evidence

We now examine the label homogeneity of higher-order k-cliques, as they form the essential building blocks of many networks [12–14, 27] and moreover, can be counted and enumerated efficiently [9, 15]. We will stick to $k \in \{2, 3, 4, 5\}$ for computational reasons.
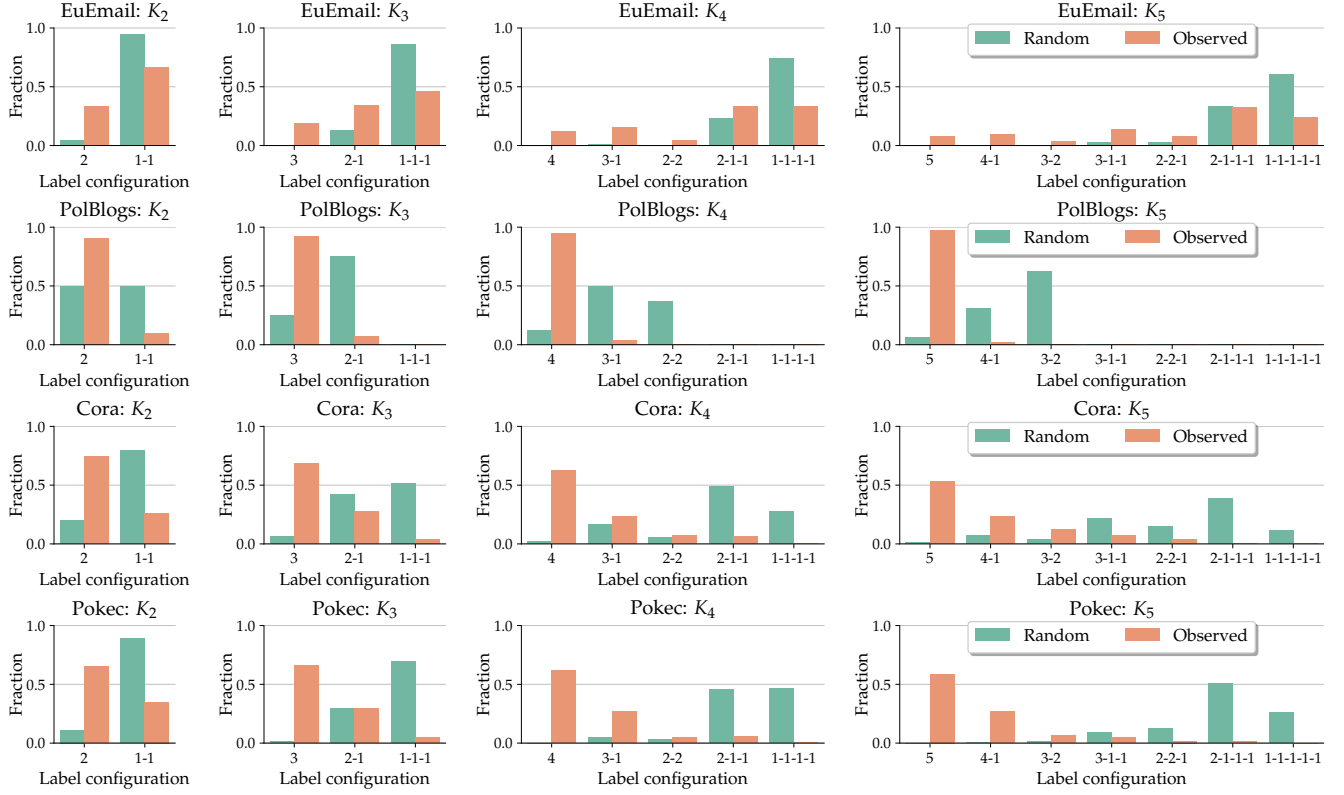
Figure 2: *Prevalence of various label configurations in real-world graphs (orange) relative to a random baseline (green) which shuffles vertex labels fixing the graph structure:* **We note that more homogeneous label configurations (towards left) are strikingly more prevalent than expected, while less homogeneous label configurations (towards right) are unusually rare.**

Methodology. We quantify label homogeneity of a given higher-order structure by measuring the distribution over what we term as its *label configurations*. Simply put, label configuration captures the extent to which participating vertices share the same label and is a function of vertex-label assignments that is invariant under the permutation of vertices and labels. A 2-clique has two label configurations: '2' where both incident vertices have the same label and '1-1' where they have different labels. A 3-clique has three label configurations: '3' where all three vertices have the same label, '2-1' where two of them share the same label and third vertex has a different label and '1-1-1' where each vertex has a different label. Similarly, a 4-clique has 5 label configurations (4, 3-1, 2-2, 2-1-1, 1-1-1-1) and a 5-clique has 7 label configurations (5, 4-1, 3-2, 3-1-1, 2-2-1, 2-1-1-1, 1-1-1-1-1). Note that not all label configurations may be possible (e.g., 1-1-1 is impossible for a triangle in a 2-class problem) and still fewer may actually occur in practice.

We will now compare the *observed* distribution over label configurations to its commensurate distribution under a *random* baseline or null model, which shuffles vertex labels fixing the graph structure and that marginal distribution of labels. A priori, there is no reason to expect the *observed* distribution to be any different from *random*. But suppose that the *observed* probability mass for homogeneous label configurations (e.g. '*k*' for *k*-cliques) exceeds that of *random* and vice versa for non-homogeneous label configurations

(e.g. '1-...-1'); this would suggest higher-order label homogeneity. Similarly, if the opposite occurs, we may conclude that vertices mix *disassortatively* [22] to form higher-order structures.

Observations. Figure 2 plots the observed distribution over *k*-clique label configurations (orange), comparing it to random (green). Under the baseline, most of the probability mass is concentrated on less homogeneous label configurations displayed towards the right, which is expected since vertex labels are assigned at random.

In sharp contrast, the observed distribution is heavily concentrated towards the label configurations on the left. Notably, the most homogeneous label configuration (i.e. '*k*' for *k*-clique, where all participating vertices have the same label), is 1.8-5.9×, 3.7-60×, 7.5-464×, and 15-3416× more common than expected for $k \in \{2, 3, 4, 5\}$ respectively. On the other hand, the least homogeneous label configuration ('1-...-1', where each participating vertex has a different label) is 1.4-5.3×, 1.8-15× and 22.2-90× rarer than expected when possible for $k \in \{2, 3, 4\}$ respectively. For CORA dataset in particular, the '1-1-1-1-1' label configuration is expected about once in eight or nine 5-clique occurrences (11.6%), but does not occur even once among its over twenty-two thousand 5-cliques.

Overall, our observations establish concrete evidence for the phenomenon of *higher-order label homogeneity*: vertices participating in real-world *k*-cliques indeed share the same label to a greater extent than can be explained by random chance.

4 HIGHER-ORDER LABEL SPREADING

We now derive our higher-order label spreading (HOLS) algorithm and show its theoretical properties including the connection to label spreading [39], closed-form solution and convergence guarantee.

4.1 Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Edges are undirected with w_{ij} representing the edge weight between vertices $i, j \in \mathcal{V}$. Each vertex has a unique label $\ell(i) \in \{1, 2, \ldots, C\}$.

Let $\mathcal{K}$ be the set of network structures or motifs (e.g., edges, triangles) which we want to leverage for graph semi-supervised learning. For a given motif $\kappa \in \mathcal{K}$, let $|\kappa|$ denote its *size* which is its number of participating vertices. For example, when κ is a triangle, $|\kappa| = 3$. Further, suppose Q_κ is the set of occurrences of a motif κ in graph $\mathcal{G}$ and each such occurrence $q \in Q_\kappa$ has a weight w_q (e.g. computed as the product of incident edge weights). Use $\mathbb{1}[\cdot]$ to denote the indicator function, which evaluates to 1 when the enclosed expression is true. For example, $\mathbb{1}[i \in q]$ is one if the vertex i is part of the subgraph q.

4.2 Generalized Loss Function

Let $\mathbf{y}_i \in \{0, 1\}^C$ be the one-hot vector of provided label for a labeled vertex i such that $y_{ic} = 1$ if vertex i has a label c and is zero otherwise. We propose to infer the final labels $\mathbf{x}_i \in [0, 1]^C$ (where $\sum_c x_{ic} = 1 \ \forall \ i \in \mathcal{V}$) by minimizing the following loss function:

$$\mathcal{L} = (1 - \eta)\mathcal{L}_s + \eta\mathcal{L}_g = (1 - \eta)\mathcal{L}_s + \eta \sum_{\kappa \in \mathcal{K}} \alpha_\kappa \mathcal{L}_{g,\kappa} \qquad (1)$$

where $\mathcal{L}_s = \frac{1}{2} \sum_i ||\mathbf{x}_i - \mathbf{y}_i||^2$ is the *supervised loss*, which penalizes the deviation of inferred labels from their provided values, while $\mathcal{L}_{g,\kappa}$ is the *graph loss* with respect to motif κ, which enforces the inferred labels to be 'smooth' over all occurrences of κ as:

$$\mathcal{L}_{g,\kappa} = \frac{1}{2} \sum_{q \in Q_\kappa} w_q \sum_{i, j \in q} ||\mathbf{x}_i - \mathbf{x}_j||^2 \qquad (2)$$

A parameter $\eta \in (0, 1)$ trades off supervised and graph losses, while $\alpha_\kappa \in (0, 1)$ captures the importance weight of κ in semi-supervised learning. Note $\sum_{\kappa \in \mathcal{K}} \alpha_\kappa = 1$. Now, define κ-participation matrix as $\mathbf{W}^{(\kappa)} = [w_{ij}^{(\kappa)}]$ where each entry $w_{ij}^{(\kappa)}$ denotes the total weight of κ-motifs that vertices i and j participate in. We have:

$$w_{ij}^{(\kappa)} = \sum_{q \in Q_\kappa} w_q \cdot \mathbb{1}[i \in q \wedge j \in q] \qquad (3)$$

Observe that each pairwise loss term $||\mathbf{x}_i - \mathbf{x}_j||^2$ in Equation (1) appears with a total weight w_{ij}' given by $w_{ij}' = \sum_{\kappa \in \mathcal{K}} \alpha_\kappa w_{ij}^{(\kappa)}$ using which we may simplify the graph loss as:

$$\mathcal{L}_g = \frac{\eta}{2} \sum_{i, j} w_{ij}' ||\mathbf{x}_i - \mathbf{x}_j||^2 \qquad (4)$$

Thus, Equation (4) establishes that the graph loss from Equation (1) is equivalent to that of label propagation [40] on a modified graph with adjacency matrix $\mathbf{W}' = \sum_{\kappa \in \mathcal{K}} \alpha_\kappa \mathbf{W}^{(\kappa)}$ where each edge of the original graph has been re-weighted according to the total weight of κ-motifs it participates in, scaled by its importance α_κ, and finally summed over all motifs $\kappa \in \mathcal{K}$ of interest. We will use this connection to derive a closed-form solution to HOLS.

Algorithm 1 Higher-Order Label Spreading (HOLS)

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, number of classes C, set of labeled vertices $\mathcal{V}_l \subset \mathcal{V}$ and their labels $\ell : \mathcal{V}_l \to \{1, \ldots, C\}$ (at least one labeled vertex per class)

Parameters: motif set $\mathcal{K}$, motif weights $\alpha_\kappa \in (0, 1)$ such that $\sum_{\kappa \in \mathcal{K}} \alpha_\kappa = 1$, weight $\eta \in (0, 1)$ for supervised loss

Output: final label assignments $\ell^*(i)$ for all vertices $i \in \mathcal{V}$

1: **procedure** HigherOrderLabelSpreading$(\mathcal{G}, \mathcal{V}_l, \ell, \mathcal{K}, \alpha, \eta)$
    ▷ *Construct higher-order normalized graph Laplacian for regularization*
2:     **for** $\kappa \in \mathcal{K}$ **do**
3:         Construct κ-participation matrix $\mathbf{W}^{(\kappa)} = [w_{ij}^{(\kappa)}]$
        ▷ $w_{ij}^{(\kappa)}$: *total weight of κ-motifs where i and j appear together*
4:     $\mathbf{W}' \leftarrow \sum_{\kappa \in \mathcal{K}} \alpha_\kappa \mathbf{W}^{(\kappa)}$
5:     $\mathbf{D}' \leftarrow diag(d_{ii}')$ where $d_{ii}' = \sum_j w_{ij}'$
6:     $\tilde{\mathbf{L}}' \leftarrow \mathbf{D}'^{-1/2}\mathbf{W}'\mathbf{D}'^{-1/2}$
    ▷ *Construct label matrices* $\mathbf{Y} = [y_{ic}]$ *(prior) and* $\mathbf{X} = [x_{ic}]$ *(inferred)*
7:     $\mathbf{Y} \leftarrow \mathbf{0}_{|\mathcal{V}| \times C}$
8:     $y_{i\ell(i)} \leftarrow 1 \ \forall \ i \in \mathcal{V}_l$
9:     $\mathbf{X} \leftarrow \mathbf{Y}$
    ▷ *Label inference using HOLS*
10:    **while** not converged **do**
11:       $\mathbf{X} \leftarrow \eta(\mathbf{I} - \tilde{\mathbf{L}}')\mathbf{X} + (1 - \eta)\mathbf{Y}$ ▷ *Equation* (8)
12:    $\ell^*(i) \leftarrow \text{argmax}_c \ x_{ic} \ \forall \ i \in \mathcal{V}$
13:    **return** ℓ^*

4.3 Closed-Form and Iterative Solutions

Let $\mathbf{Y} = [\mathbf{y}_1 \ldots \mathbf{y}_N]^T$ and $\mathbf{X} = [\mathbf{x}_1 \ldots \mathbf{x}_N]^T$ be the $N \times C$ matrices of prior and inferred labels where $|\mathcal{V}| = N$ is the total number of vertices. Let $\mathbf{D}' = [d_{ij}']$ be the diagonal degree matrix for the modified graph adjacency $\mathbf{W}' = [w_{ij}']$. Thus, $d_{ii}' = \sum_j w_{ij}'$ and $d_{ij}' = 0$ if $i \neq j$. Let $\mathbf{L}' = \mathbf{D}' - \mathbf{W}'$ be the Laplacian matrix for the modified graph. Equation (4) can be re-written in matrix format as:

$$\mathcal{L} = \frac{1 - \eta}{2}||\mathbf{X} - \mathbf{Y}||_F^2 + \frac{\eta}{2}\mathbf{X}^T\mathbf{L}'\mathbf{X} \qquad (5)$$

We also consider a version of the loss function which uses the *normalized* Laplacian $\tilde{\mathbf{L}}' = \mathbf{D}'^{-1/2}\mathbf{L}'\mathbf{D}'^{-1/2}$ for regularization:

$$\tilde{\mathcal{L}} = \frac{1 - \eta}{2}||\mathbf{X} - \mathbf{Y}||_F^2 + \frac{\eta}{2}\mathbf{X}^T\tilde{\mathbf{L}}'\mathbf{X} \qquad (6)$$

Using $\tilde{\mathbf{L}}'$ in place of $\mathbf{L}'$ performs as well if not better in practice; and moreover provides certain theoretical guarantees (see Proposition 4.2, and also [33]). Therefore, we will use Equation (6) as the loss function for our higher-order label spreading and refer to it as $\mathcal{L}_{\text{HOLS}}$. A closed-form solution can now be obtained by differentiating $\mathcal{L}_{\text{HOLS}}$ with respect to $\mathbf{X}$ and setting it to zero. Thus:

$$\mathbf{X} = (1 - \eta)\left(\mathbf{I} - \eta(\mathbf{I} - \tilde{\mathbf{L}}')\right)^{-1}\mathbf{Y} \qquad (7)$$

Thus, using Equation (7), we are able to compute the optimal solution to HOLS, as long as the inverse of $\mathbf{I} - \eta(\mathbf{I} - \tilde{\mathbf{L}}')$ exists. Due to the use of normalized Laplacian regularization, the following holds:

PROPOSITION 4.1 (GENERALIZED LABEL SPREADING). *The proposed HOLS algorithm reduces to traditional label spreading [39] for the base case of using only edge motifs, i.e., $\mathcal{K} = \{K_2\}$.*

This generalization grants HOLS its name. In practice, matrix inversion is computationally intensive and tends to be numerically unstable. Hence, we resort to an iterative approach to solve Equation (7) by first initializing $\mathbf{X}$ to an arbitrary value and then repeatedly applying the following update:

$$\mathbf{X} \leftarrow \eta(\mathbf{I} - \tilde{\mathbf{L}}')\mathbf{X} + (1 - \eta)\mathbf{Y} \tag{8}$$

Proposition 4.2 describes the theoretical properties of this approach.

PROPOSITION 4.2 (CONVERGENCE GUARANTEE FOR HOLS). *The iterative update in Equation* (8) *always converges to the unique fixed point given in Equation* (7) *for any choice of initial* $\mathbf{X}$.

This can be proved using the theory of sparse linear systems [25].

The overall algorithm of HOLS is summarized in Algorithm (1). First, for each motif $\kappa \in \mathcal{K}$, construct its κ-participation matrix by enumerating all its occurrences. Note that the enumerated occurrences are processed one by one on the fly to update the participation matrix and discarded (no need for storage). Moreover, the enumeration for different motifs can be done in parallel. The participation matrices are combined into a single modified graph adjacency $\mathbf{W}'$; applying the iterative updates from Equation (8) finally results in labels for the unlabeled vertices. In practice, the iterative updates are applied until entries in $\mathbf{X}$ do not change up to a precision ϵ or until a maximum number of iterations T is reached.

4.4 Complexity Analysis

When only cliques are used as motifs $\mathcal{K}$ for semi-supervised learning, the following space and time complexity bounds hold:

PROPOSITION 4.3 (SPACE COMPLEXITY OF HOLS). *The space complexity of HOLS for a graph with N vertices, M edges and C classes is $O(M + NC)$ independent of motif size and number of motifs used, provided all motifs are cliques.*

PROPOSITION 4.4 (TIME COMPLEXITY OF HOLS). *The time complexity of HOLS over a graph with M edges, C classes and a degeneracy (core number) of $k_{\max}$ using $\mathcal{K} = \{K_2, \ldots, K_n\}$ is given by $O\left(M \sum_{k=2}^n k \left(\frac{k_{\max}}{2}\right)^{k-2}\right)$ for the construction of K_k-participation matrices plus $O(MC)$ per iterative update using Equation* (8).

The proofs rely on the sparsity structure of the modified adjacency $\mathbf{W}'$ and also Theorem 5.7 of [9]. Despite the exponential complexity in k, we are able to enumerate cliques quickly using the sequential KCLIST algorithm [9]. For example, our largest POKEC dataset has 21M edges, 32M triangles, 43M 4-cliques and 53M 5-cliques; and the enumeration of each took at most 20 seconds on a stock laptop. Thus, HOLS remains fast and scalable when reasonably small cliques are used. Further, as we show in experiments, using triangles (K_3) in addition to edges typically suffices to achieve the best classification performance across a wide range of datasets.

5 EXPERIMENTS

We empirically evaluate the proposed algorithm on the four real-world network datasets described in Section 3.1.

5.1 Experimental Setup

We implement HOLS in MATLAB and run the experiments on MacOS with 2.7 GHz Intel Core i5 processor and 16 GB main memory.

Baselines. We compare HOLS to the following baselines: *(1) Label Propagation (LP)* [40] which uses Laplacian regularization. *(2) Label Spreading (LS)* [39] which uses normalized Laplacian regularization. *(3) node2vec+TSVM* which generates unsupervised vertex embeddings using node2vec [11] and learns decision boundaries in the embedding space using one-vs-rest transductive SVMs [16]. *(4) Graph Convolutional Network (GCN)* [17] which is an end-to-end semi-supervised learner using neural networks. We implement LP and LS in MATLAB, and use open-sourced code for the rest.

Parameters. By default, we use a weight of $\eta = 0.5$ for supervised loss and $\mathcal{K} = \{K_2, K_3\}$ motifs (edges and triangles) for HOLS. The importance weight for triangles α_{K_3} is tuned in $\{0.1, 0.2, \ldots, 0.9\}$ for each dataset and results are reported on the best performing value. We use $\eta = 0.5$ for LS as well. LP, LS and HOLS are run until labels converge to a precision of ϵ or until T iterations are completed, whichever occurs sooner. We set $\epsilon = 10^{-6}$ and $T = 500$. We use the default hyperparameters for GCN, node2vec and TSVM. We supply 100, 20, 100 and 1000 labels for EUEMAIL, POLBLOGS, CORA and POKEC datasets, where the vertices to label are chosen by stratified sampling based on class. These correspond to label fractions of 5%, 1.6%, 0.4% and 0.06% and on an average, 1, 10, 10 and 100 labeled vertices per class respectively.

Evaluation metrics. We quantify the success of semi-supervised learning using *accuracy*, which is the fraction of unlabeled vertices which are correctly classified. We also note down the end-to-end *running time* for all computation including any pre-processing such as clique enumeration, but excluding I/O operations.

5.2 Results

We present our experimental results. All reported values are averaged over five runs, each run differing in the set of labeled vertices.

Accuracy. Accuracies of all methods are summarized in Table 3 (left). The values for node2vec+TSVM on POKEC dataset are missing as the method did not terminate within 24 hours ('T.L.E.').

First, we observe in Table 3 (left) that HOLS consistently leads to statistically significant improvements over LS, showing that using higher-order structures for label spreading helps. In addition, HOLS outperforms *all* baselines in three out of four datasets. The improvements over the best baseline are statistically significant ($p < 0.05$) according to a two-sided micro-sign test [35] in at least three out of five runs. Importantly, for the smaller datasets (EUEMAIL and POLBLOGS), while GCN outperforms LS, GCN loses to HOLS when triangles are used. node2vec+TSVM performs slightly better than HOLS on POLBLOGS, however, the increase over HOLS is not statistically significant. For the larger datasets with < 0.5% labeled vertices, HOLS performs the best and LS follows closely.

Running Time. The running time of HOLS and all the baselines is summarized in Table 3 (right). Notably, we see that HOLS runs in less than 2 minutes for graphs with over 21 million edges (the POKEC graph), demonstrating its real-world practical scalability.

We observe that LS is the fastest of all methods and HOLS comes a close second for three out of four datasets. The small difference in running time predominantly stems from the construction of triangle participation matrix. Furthermore, HOLS is over 15× faster than the recent GCN and node2vec+TSVM baselines, *for comparable and often better values of accuracy.*

Table 3: Accuracy and Running Time (averaged over five runs): In each column, the best value is bold and underlined, and the second best is underlined. Asterisk (*) denotes statistically significant differences ($p < 0.05$) compared to the second best.

Metric	Accuracy				Running time (seconds)			
Method	EuEmail	PolBlogs	Cora	Pokec	EuEmail	PolBlogs	Cora	Pokec
Label Propagation (LP) [40]	0.2905	0.5814	0.2765	0.1994	0.11	0.070	2.1	1320
Label Spreading (LS) [39]	0.5228	0.9361	0.4921	0.5514	**0.040***	**0.036***	**0.21***	**93***
node2vec+TSVM [11, 16]	0.4563	**0.9481**	0.4233	T.L.E.	46	29	3060	>1 day
Graph Convolution Networks (GCN) [17]	0.5251	0.9470	0.4673	0.5290	1.8	1.3	6.4	2880
HOLS (proposed)	**0.5473***	0.9476	**0.4953***	**0.5593***	0.089	0.083	0.41	117
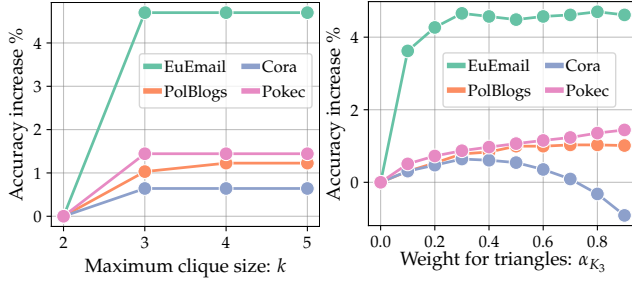
Figure 3: Variation of accuracy with maximum clique size k (left), and importance weight α_{K_3} to triangles (right).
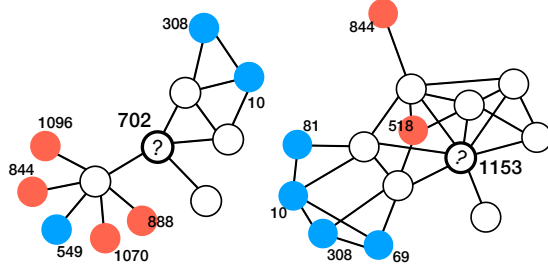
Figure 4: Case studies from PolBlogs dataset showing extended ego-networks of vertices v702 (left) and v1153 (right) which are both incorrectly classified by LS but correctly classified when triangles are taken into account using HOLS.

Accuracy vs. Maximum Clique Size. Fixing the motif set as $\mathcal{K} = \{K_2, K_3, \ldots, K_k\}$, we vary $k = 2, 3, 4, 5$ to study the marginal benefit of including higher-order cliques in graph SSL. The motif weights are tuned in $\alpha_{K_j} \in \{0, 0.1, \ldots, 0.9\}$, ensuring that edges are given a weight $\alpha_{K_2} \geq 0.1$ for a connected graph, and further, all motif weights sum to 1. The best performing motif weights were used to generate Figure 3(a), which plots the relative improvement in accuracy over LS that uses edges only. We note that label spreading via higher-order structures strictly outperforms label spreading via edges. The gain is the most when using 3-cliques (triangles). Subsequent higher-order cliques did not lead to additional performance gain in most cases, presumably because their occurrences tend to be concentrated around a few high-degree vertices.

Accuracy vs. Importance Weight To Triangles. Fixing the motif set to $\mathcal{K} = \{K_2, K_3\}$, we vary the importance weight α_{K_3} to triangles in $\{0, 0.1, \ldots, 0.9\}$ to understand its effect on accuracy.

Figure 3(b) shows that the accuracy gain of HOLS over LS increases with an increase in triangle weight for most graphs. The only exception is Cora, where the accuracy gain grows until $\alpha_{K_3} = 0.4$ before decreasing and eventually turning negative. Overall, triangles consistently help over a large range of motif weights.

Case Studies. In Figure 4, we look at real examples from the PolBlogs dataset to dig deeper into when HOLS improves over LS. Question mark denotes the central vertices v702 and v1153 of interest with ground truth labels 'blue' and 'red' respectively. The direct neighbors of the both vertices are unlabeled and a few second hop neighbors are labeled with one of two labels: 'blue' or 'red'.

In both cases, both LS and HOLS classify the unlabeled 1-hop neighbors correctly. However, LS, relying only on the edge-level information (roughly the ratio of blue to red labeled 2-hop neighbors in this case), incorrectly labels both v702 and v1153. The proposed HOLS, on the other hand, accurately recognizes that v702 (v1153) is more tightly connected with its blue (red) neighbors via the higher-order triangle structures and thus leads to the correct classification.

6 CONCLUSION

In this paper, we demonstrated that label homogeneity–the tendency of vertices participating in a higher-order structure to share the same label–is a prevalent characteristic in real-world graphs. We created an algorithm to exploit the signal present in higher-order structures for more accurate semi-supervised learning over graphs. Experiments on real-world data showed that using triangles along with edges for label spreading leads to statistically significant accuracy gains compared to the use of edges alone.

This work opens the avenue for several exciting future research directions. First, we need principled measures quantifying label homogeneity to aid comparison across diverse graphs and higher-order structures. Next, having seen the improvements in real-world graphs, it becomes fundamental to understand the benefits in random graph models. Finally, it is crucial to develop algorithms exploiting higher-order structures which can cater to the increasingly heterogeneous and dynamic nature of real-world graphs at scale.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *ICML (Proceedings of Machine Learning Research)*, Vol. 97. PMLR, 21–29.

[2] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. 2019. Link Prediction via Higher-Order Motif Features. *CoRR* abs/1902.06679 (2019).

[3] Lada A. Adamic and Natalie S. Glance. 2005. The political blogosphere and the 2004 U.S. election: divided they blog. In *LinkKDD*. ACM, 36–43.

[4] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion Fraud Detection in Online Reviews by Network Effects. In *ICWSM*. The AAAI Press.

[5] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[6] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. 2004. Regularization and Semi-supervised Learning on Large Graphs. In *COLT (Lecture Notes in Computer Science)*, Vol. 3120. Springer, 624–638.

[7] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon M. Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proc. Natl. Acad. Sci. U.S.A.* 115, 48 (2018), E11221–E11230.

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

[9] Maximilien Danisch, Oana Denisa Balalau, and Mauro Sozio. 2018. Listing k-cliques in Sparse Real-World Graphs. In *WWW*. ACM, 589–598.

[10] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. 2017. ZooBP: Belief Propagation for Heterogeneous Networks. *PVLDB* 10, 5 (2017), 625–636.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. ACM, 855–864.

[12] Robert A Hanneman and Mark Riddle. 2005. Introduction to social network methods.

[13] Matthew O Jackson. 2010. *Social and economic networks*. Princeton university press.

[14] Matthew O Jackson, Tomas Rodriguez-Barraquer, and Xu Tan. 2012. Social capital and social quilts: Network patterns of favor exchange. *American Economic Review* 102, 5 (2012), 1857–97.

[15] Shweta Jain and C. Seshadhri. 2017. A Fast and Provable Method for Estimating Clique Counts Using Turán's Theorem. In *WWW*. ACM, 441–449.

[16] Thorsten Joachims. 1999. Transductive Inference for Text Classification using Support Vector Machines. In *ICML*. Morgan Kaufmann, 200–209.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[18] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 333–341.

[19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1269–1278.

[20] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *TKDD* 1, 1 (2007), 2.

[21] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[22] Mark EJ Newman. 2003. Mixing patterns in networks. *Physical Review E* 67, 2 (2003), 026126.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. ACM, 701–710.

[24] Ryan A. Rossi, Anup Rao, Sungchul Kim, Eunyee Koh, Nesreen K. Ahmed, and Gang Wu. 2019. Higher-Order Ranking and Link Prediction: From Closing Triangles to Closing Higher-Order Motifs. *CoRR* abs/1906.05059 (2019).

[25] Yousef Saad. 2003. *Iterative methods for sparse linear systems*. Vol. 82. SIAM.

[26] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (2008), 93–106.

[27] Ann Sizemore, Chad Giusti, and Danielle S Bassett. 2017. Classification of weighted networks through mesoscale homological features. *Journal of Complex Networks* 5, 2 (2017), 245–273.

[28] Lovro Subelj and Marko Bajec. 2013. Model of complex networks based on citation dynamics. In *WWW (Companion Volume)*. International World Wide Web Conferences Steering Committee / ACM, 527–530.

[29] Lubos Takac and Michal Zabovsky. 2012. Data analysis in public social networks. In *International Scientific Conference & International Workshop Present Day Trends of Innovations*.

[30] Partha Pratim Talukdar and Koby Crammer. 2009. New Regularized Algorithms for Transductive Learning. In *ECML/PKDD (2) (Lecture Notes in Computer Science)*, Vol. 5782. Springer, 442–457.

[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. *CoRR* abs/1503.03578 (2015).

[32] Alexei Vazquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. 2003. Global protein function prediction from protein-protein interaction networks. *Nature biotechnology* 21, 6 (2003), 697.

[33] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. 2008. Consistency of spectral clustering. *The Annals of Statistics* (2008), 555–586.

[34] Jason Weston, Frédéric Ratle, and Ronan Collobert. 2008. Deep learning via semi-supervised embedding. In *ICML (ACM International Conference Proceeding Series)*, Vol. 307. ACM, 1168–1175.

[35] Yiming Yang and Xin Liu. 1999. A Re-Examination of Text Categorization Methods. In *SIGIR*. ACM, 42–49.

[36] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML (JMLR Workshop and Conference Proceedings)*, Vol. 48. JMLR.org, 40–48.

[37] Jonathan S Yedidia, William T Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*. Morgan Kaufmann Publishers Inc., 239–269.

[38] Hao Yin, Austin R Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. *Physical Review E* 97, 5 (2018), 052306.

[39] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with Local and Global Consistency. In *NIPS*. MIT Press, 321–328.

[40] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *ICML*. AAAI Press, 912–919.