Graph Representation Learning with Motif Structures

Yan Ge

Department of Computer Science

University of Sheffield

This dissertation is submitted for the degree of

Doctor of Philosophy

June 2021

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Yan Ge

29th June 2021

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Dr. Haiping Lu, for his consistent guidance, support, patience. I want to thank him for giving me the opportunity to do this PhD under his supervision.

I would like to thank Dr. Jun Ma, Dr. Peng Pan and Li Zhang for their invaluable advice, guidance and close research collaboration. I sincerely thank Amazon.com.Inc for providing financial supports through Amazon Research Awards (PI: Dr. Haiping Lu).

Special thanks to my friends in the Machine Learning group who have been great colleagues: Xianyuan, Shuo, Chao, Lawrence, Johanna, Chunchao, Peizhen, Juan, Tianqi, Wenwen and some colleges in the Testing group: Prof. Gordon Fraser, Nasser, Sina, José Campos, José Miguel Rojas.

Finally, I want to thank my love, my parents for their endless love and support.

Abstract

Graphs are important data structures that can be found in a wide variety of real-world scenarios. It is well recognised that the primitive graph representation is sparse, high-dimensional and noisy. Therefore, it is challenging to analyse such primitive data for downstream graph-related tasks (e.g., community detection and node classification). Graph representation learning (GRL) aims to map graph data into a low-dimensional dense vector space in which the graph information is maximally preserved. It allows primitive graphs to be easily analysed in the new mapped vector space.

GRL methods typically focus on simple connectivity patterns that only explicitly model relations between two nodes. Motif structures that capture relations among three or more nodes have been recognised as functional units of graphs, and can gain new insights into the organisation of graphs. Therefore, in this thesis we propose new GRL methods modelling motif structures for different graph-related tasks and applications along three directions: (1) a method to learn a spectral embedding space with both edge-based and triangle-based structures for clustering nodes; (2) a graph transformer by unifying homophily and heterophily representation for role classification and motif structure completion; (3) a method to tackle noises in knowledge graph representations with motif structures for recommendations and knowledge graph completion. Experimental studies show that the proposed methods have outperformed related state-of-the-art methods for targeted tasks and applications.

Important Notations and Abbreviations

Symbol	Definition
$a_{ij}^{(l)}$	lth-order proximity value between node v_i and v_j
$assoc_2(S_1)$	Total degrees of vertices in the subgraph induced by vertices in S_1
$assoc_3(S_1)$	Number of nodes in triangles in the subgraph induced by vertices in S_1
$\mathbf{A}^k$	kth-order proximity
$\mathbf{A}_M$	Homophily representation with motif M
$cut_2(S_1, S_2)$	Number of edges between S_1, S_2
$cut_3(S_1, S_2)$	Number of triangles between S_1, S_2
$\mathbf{c}$	Containing (e_h, r, e_t) community indicator vector
C	Cluster set
$\mathbf{D}$	Degree matrix
e_h, r, e_t	Head, relation, tail
$\mathbb{E}[w(v_d, v_k)]$	Expectation of the weight of the edge connecting nodes v_i and v_k
g_E	Energy function score for TransE
$\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$	KG entity set $\mathcal{E}$ and relation set $\mathcal{R}$
$\mathbf{H}$	Heterophily representation
$\mathbf{L}$	Laplacian matrix
$\mathcal{L}_k^{(n)}$	Neural margin-based ranking loss
$\mathcal{L}_k^{(w)}$	Weighted margin-based ranking loss

$\mathbf{m}(e_h, r, e_t)$	Motif feature vector for the triple (e_h, r, e_t)
$\mathcal{M}^L$	$L-$layer fully-connected neural network
$p(e_t\|e_h)$	Value of the resource on the tail entity from head entity
p, q	Weight of edges for intra-communities and inter-communities under PPM, respectively
$\mathbf{p}_u$, $\mathbf{q}_i$	Latent vector for user u and item i
$\mathcal{P}$	Transition tensor
$\mathbf{Q}$	Unification of homophily and heterophily representation
S_1^*, S_2^*	Two disjointed true clusters
S_1, S_2	Two disjointed clusters produced by an algorithm
$\hat{t}(e_h, r, e_t)$	Trustworthiness value of the triple (e_h, r, e_t)
$\hat{t}(e)$	Trustworthiness of entity e
$\mathcal{T}$	Adjacency tensor
$vol_2(S_1)$	Total degrees of vertices in S_1
$vol_3(S_1)$	Number of nodes in triangles that reside in S_1
$\mathbf{W}$	Adjacency matrix
$\hat{y}_{upv}$	Probability that user u will engage with item v
ε_N	Number of mis-clustered nodes,
ε_E	Number of mis-clustered edges
ε_T	Number of mis-clustered triangles
λ	Weight parameter
$\phi_3(S)$	Triangle conductance of S
$\oplus$	Symmetric difference
$\delta(\cdot)$	Non-linear sigmoid activation function

Abbreviation	Description
AROPE	Arbitrary-order proximity embedding
CF	Collaborative Filtering
GDC	Graph Diffusion Convolution
GL	Graph Laplacian
GNN	Graph Neural Networks
HOP	Higher-order Proximity
HOSPLOC	High-Order Structure Preserving Local Clustering
H^2GT	Homophily and Heterophliy Graph Transformation
KG	Knowledge Graph
KGR	Knowledge Graph Representation
MF	Matrix Factorisation
MOSC	Mixed-Order Spectral Clustering
MRL	Margin-based Ranking Loss
MSC	Motif-based Spectral Clustering
GRL	Graph Representation Learning
PPM	Planted Partition Model
RS	Recommender Systems
RW	Random Walks
SC	Spectral Clustering
TrustRec	Trustworthiness-aware Knowledge Graph for recommendations
TTM	Tensor Trace Maximisation

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graphs (a.k.a. networks) are important data structures that abstract relations between discrete objects, such as social network [116], citation networks [27] and brain networks [83]. A graph is composed of nodes (a.k.a. vertices) and edges (a.k.a. links) representing node interactions. For example, in social networks (Fig.1.1), nodes are people and there is an edge between two people if they are friends. In citation networks, one node represents one paper, and one edge is directed from one paper to another that it cites.

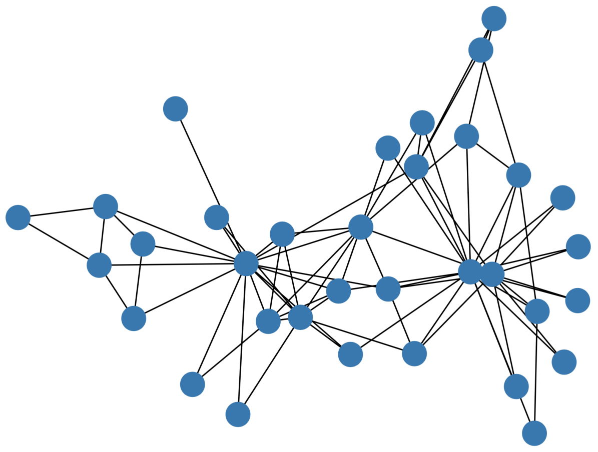


Fig. 1.1 The illustration of Zachary Karate Club Network [127] represents the friendship between members in a karate club, which is produced by the author. An edge connects two people if they interacted outside of the club.

Machine learning in graph analysis plays an essential role in a range of tasks across many disciplines. For example, in social networks, clustering is valuable for targeted advertising and friends recommendation. There are three important graph analysis tasks that are clustering [85], node classification [11, 132, 130, 131] and link prediction [16]. Specifically, clustering aims to find subsets of similar nodes and group them together. Node classification is used to predict labels of nodes based on labelled nodes and graph structures. Link prediction aims to complete graphs by predicting some missing links or links that are likely to occur in the future.

To complete the above graph analysis tasks, a key problem is about how to incorporate graph structural information into machine learning models. Therefore, conventional methods heavily rely on graph statistics (e.g., degrees, common neighbours) to extract graph structural information [11]. However, these methods are inflexible and time-consuming because such hand-engineered features cannot adapt during the learning process and manual design is very expensive. In recent years, there has been emerging a large number of methods that can automatically learn representations that encode graph structural information. The idea is that representation learning approaches aim to learn a mapping function that embeds nodes, edges, or entire (sub)graphs, into a low-dimensional vector space [47]. The goal is to optimise this mapping function so that geometric relationships in this learned space can reflect and preserve the structural information of the original graph. After optimising the embedding space, the learned embeddings can be used as feature inputs for downstream machine learning tasks.

Most graph analysis methods are based on an edge structure that is the simplest structure to reflect a pairwise relation. However, in many real-world graphs (e.g., social networks), the minimal and functional structural unit of a graph is not a simple edge but a small network subgraph that involves more than two nodes [9], which is called motifs (a.k.a higher-order structures) [1]. This term is introduced in the paper [94] when they studied the frequency of interactions of more than two nodes through subgraph analysis. Subsequently, Milo *et al.* [76] define motifs as recurrent and statistically significant subgraphs or patterns of a larger graph. In follow-up work, Milo *et al.* [75] showed that the frequencies of the 13 connected, 3-node directed subgraphs (Fig.1.2) were sufficient to distinguish biological, social, and

[1]This thesis uses "motifs" when describing connectivity patterns in graphs and use the term "higher-order structures" when discussing the more mathematical points.

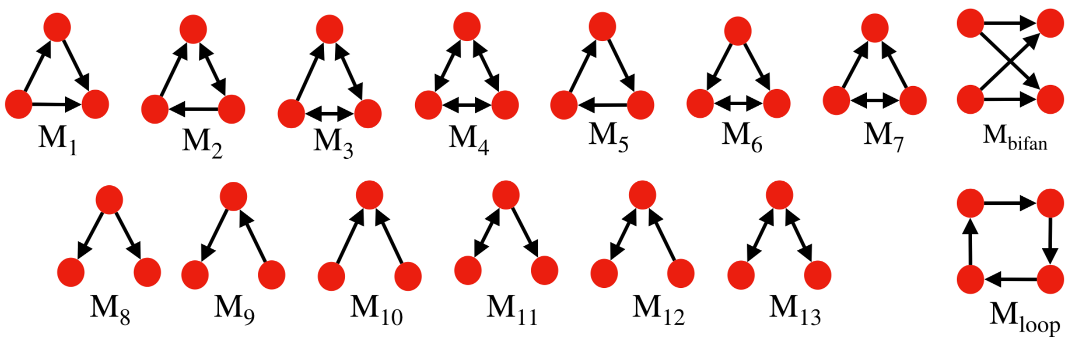


Fig. 1.2 Illustration of all possible three-nodes motifs ($M_1 - M_{13}$) and two examples of four-node motifs (M_{bifan} and M_{loop}).

linguistic networks. In subsequent research, important motifs have been identified in a variety of domains, including brain science [98], biology [118], and social network analysis [61].

There is substantial evidence that motif structures, or small subgraph patterns between a few nodes, are essential to the behaviour of many complex systems modelled by graphs [76, 75] because they can reflect functional properties. Fig. 1.2 lists all possible thirteen different three-node motifs. For example, triangular structures M4, with three reciprocated edges connecting three nodes, play important roles in social networks [61]. In this case, two people who share a common friend are more likely to become friends themselves, which forms a triangular structure. Motif M5 that is a feed-forward loop is a characteristic motif in neurons connectivity [10]. The reason for these characteristics relies on the functionality of such small subgraphs on the performance of specific graphs.

1.1 Motivation and Research Questions

Graph representation learning (GRL) has been widely recognised as an effective approach to learn the low-dimensional representations of vertices in graphs. The primitive graph representation suffers from overwhelming high dimensionality and sparsity. GRL aims to learn low-dimensional and dense continuous latent representations of nodes on a graph. Meanwhile, it can preserve the structure and the inherent properties of the graph, which can then be exploited effectively in downstream tasks. Fig.1.3 shows the GRL framework

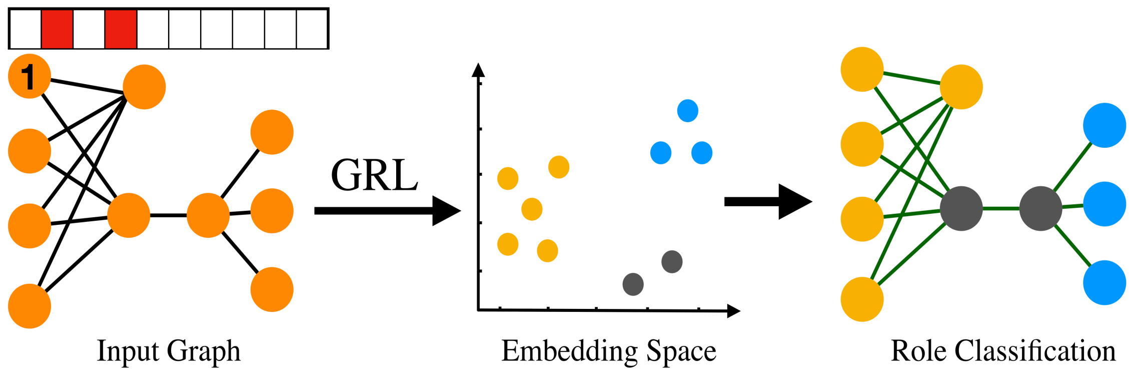


Fig. 1.3 Illustration of the graph representation learning framework that takes the role classification as a downstream task. In order to encode the connectivity information for each node in a primitive input graph, we need to use a high-dimension and sparse vector. In this example, such a vector is shown as a heat map, in which non-zero values are highlighted with red. After using GRL techniques, we can learn the low-dimensional embedding space of each node while preserving the inherent graph structure. The learned representations are used as feature inputs for downstream machine learning models (e.g., logistic regression).

that takes high-dimensional and sparse connectivity information as an input for a role classification task.

Motif structures have proven fundamental to gain insights into diverse complex graphs. Additionally, the usage of motifs allows for greater modelling flexibility. Among all the possible motifs, a simple motif structure is the triangle which represents a basic unit of transitivity in graphs. For triangular motif structures, total thirteen different interactions among three nodes can be found when considering directions (Fig. 1.2), but only two different edge structures. The application will decide which motif structures should be modelled. Despite its key importance, motif structures have not been well integrated into the analyses, models, and algorithms that we actually use to study the graph representation. Although much work in recent years has been dedicated to refining and improving the performance of GRL algorithms, most methods still are limited to model pairwise relations [135, 56, 87, 82, 101]. That is, these works perform graph prediction (e.g., link prediction, node classification) tasks based on pairwise relations between nodes and ignore motifs structures involving more than two nodes. It is widely known that motifs are essential to the structure and function of complex graphs.

Therefore, the motivation of this thesis is to develop new algorithms that encode motif structures into GRL to improve some remarkable graph-related tasks. This thesis will focus on the clustering, classification and link prediction tasks. In order to improve the performance of these tasks, this thesis answers and solves three key research questions. Note that before describing research questions, this thesis will give a brief background about them and the detailed background will be introduced in Chapter 2.

1. Spectral clustering (SC) is a popular approach for gaining insights from complex graphs by grouping similar nodes together. Conventional SC methods focus on the simple edge structures without direct consideration of motif structures. This has motivated SC extensions that directly consider motif structures. However, both conventional SC methods and its extensions are limited to considering a single type. Therefore, the first research question is about *how to incorporate both edge and motif structures into spectral embedding space for graph clustering.*

2. Higher-order proximity (HOP) is fundamental for most graph embedding methods due to its significant effects on the quality of node embedding and performance on downstream graph analysis tasks. Most existing HOP definitions are based on either homophily to place close and highly interconnected nodes tightly in embedding space or heterophily to place distant but structurally similar nodes together after embedding. In real-world graphs, both can co-exist, and thus considering only one could limit the prediction performance and interpretability. Additionally, there is no general and universal solution that takes both into consideration. Therefore, the second research question is about *how to leverage motif structures to unify homophily and heterophily graph representation via a universal solution?*

3. Incorporating knowledge graphs (KGs) into recommender systems (RS) has recently attracted increasing attention. For large-scale KGs, due to limited labour supervision, noises are inevitably introduced during automatic construction. However, most existing KG-aware RS do not consider such noises in KGs, which can degrade the performance of KG-aware RS. Therefore, the third research question is about *how to use motif*

structures to estimate the noises of triples for KG representation and improve the performance of RS?

1.2 Research Challenges and Hypothesis

The overall objective of this thesis is to automatically learn graph representation with motif structures. To achieve this, this thesis first looks at the potential challenges this raises, and then discusses how each challenge can be addressed with motif representation, as follows:

1. **Structure preservation of different orders in clusters.** The real-world graphs have both edge and higher-order structures, and both can be important. Existing conventional and higher-order SC methods model only either edges or higher-order structures. For example, edge-based SC does not take motifs into consideration, while higher-order SC loses information of some edges, in particular, those that do not belong to any motifs. That is, only preserving one type of structures will lead to largely missing another structures. The challenge is the preservation of both types of structures in spectral embedding space and generates clusters with rich structures.

2. **Inflexibility of GRL assumptions.** Most existing GRL methods are based on either the homophily assumption to place close and highly interconnected nodes tightly in embedding space or the heterophily assumption to place distant but structurally similar nodes together after embedding. Such a "one-size-fit-all" representation potentially limits the performance and interpretation of many graph-based tasks. Moreover, there is no universal solution to unify both assumptions. Therefore, a challenge is to develop a universal method to flexibly unify both homophily and heterophily assumptions in the graph embedding space.

3. **Noise-tolerant KGR for RS.** The KGs usually contain tremendous structured facts. Incorporating such rich facts into RS has attracted increasing attention recently. However, plenty of noises are inevitably introduced in large-scale KGs during automatic

construction due to limited labour supervision. These noises can degrade the performance of knowledge-aware RS. Therefore, a challenge is about automatically estimating noises and developing a noise-tolerant method to improve the performance of knowledge-aware RS.

The hypothesis of this thesis is that motif-aware GRL can capture complex interactions among nodes to learn better node representation, which can improve the performance of graph-based tasks and applications.

1.3 Structure and Contributions

This section outlines the structure of this thesis, alongside presenting the key novel contributions of this thesis. Prior to it, this thesis structures the contributions and important tasks in this thesis as shown in Fig. 1.4.

Chapter 2: "Background" presents a survey of the literature relating to the topics explored in this thesis. This chapter begins with describing definitions of motif structures and graph representation learning. It then presents spectral clustering including some important steps : normalised graph Laplacian, sorting nodes and cut criteria. Next, this chapter introduces the concept of higher-order proximity that is commonly used for structure preservation in graph embedding space. Next, this chapter shows two GRL assumptions that are homophily and heterophily, which will help to describe the motivation of the work (Chapter 4). Additionally, this chapter reviews some important works about knowledge graph embedding methods. Finally, the recommender system as one important application of trustworthiness-aware knowledge graph embedding will be introduced.

Chapter 3: "Mixed-order Spectral Clustering for Graphs" presents two spectral clustering methods by modelling both edge and triangle structures in spectral embedding space simultaneously. The first method is based on graph Laplacian, with its theoretical performance guarantee derived by proving a Cheeger inequality. For the second method, this chapter defines a new random walks model for a probabilistic interpretation. After that, this chapter introduces newly designed cut criteria to enable existing single-order SC to preserve

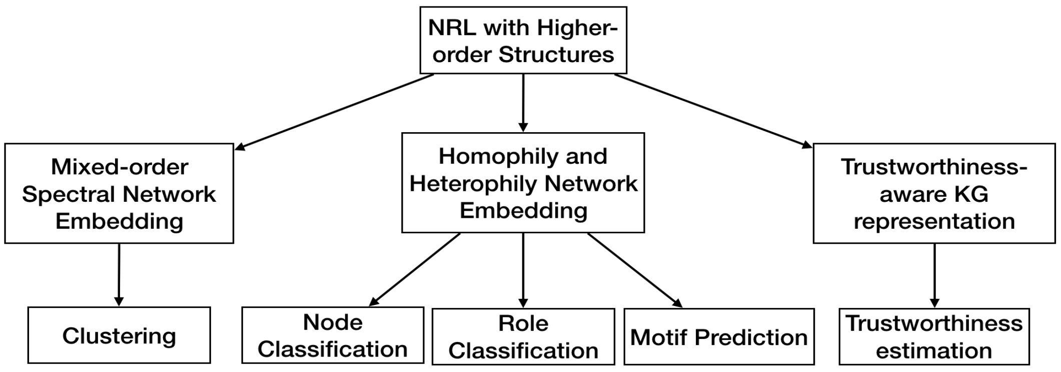

NRL with Higher-order Structures
Mixed-order Spectral Network Embedding
Homophily and Heterophily Network Embedding
Trustworthiness-aware KG representation
Clustering
Node Classification
Role Classification
Motif Prediction
Trustworthiness estimation

Fig. 1.4 The structured contributions of this thesis in motif-aware GRL and important tasks.

mixed-order structures, and new mixed-order evaluation metrics for structure evaluation. This chapter introduces experiments on community detection and superpixel segmentation tasks to show the superior performance of our algorithms over existing methods.

Contribution 1: We develop two new algorithms for preserving both edge and triangle structures information simultaneously in spectral embedding space for node clustering by using graph Laplacian and random walks, along with new cut criteria and evaluation metrics.

Chapter 4: "Unifying Homophily and Heterophily Graph Transformation via Motifs" theoretically studies why most existing GRL methods with higher-order proximity preservation only hold a homophily assumption. Thus, in this chapter we propose a universal framework to unify both homophily and heterophily assumptions by newly designing micro-level and macro-level walk paths, which leverages the motif representation. Additionally in this chapter we conduct experiments on node classification, structural role classification and motif prediction to show the superior prediction performance and computational efficiency over state-of-the-art (SOTA) methods.

Contribution 2: We propose micro-level and macro-level walk paths with the motif representation to preserve homophily and heterophily in HOP by theoretically studying why most HOP preserving embedding methods only hold a homophily assumption, and propose a novel framework to unify homophily and heterophily representations, and three instantiations.

Chapter 5: "Trustworthiness-aware Knowledge Graph Representation for Recommendation" proposes to estimate the trustworthiness of triples through internal structural information: motifs, communities and global information. In this chapter, we then integrate triple trustworthiness into a weighted/neural loss function of KGR to learn noise-tolerant KGR. Meanwhile, in this chapter we integrate entity trustworthiness into RS to learn noise-tolerant item representations for RS. Lastly, we conduct extensive experiments to show the superior performance of our method over SOTA methods for book and movie recommendations.

Contribution 3: We propose a trustworthiness estimator for KGR to take noises in KGs into account by considering motifs, communities and global structural information, and integrate it to learn noise-tolerant KGR and item representations for RS.

Chapter 6: "Conclusion and Future Work" summarises findings and the contributions of this thesis. This chapter also presents future research directions in the area based on the work presented in the prior chapters.

The contents of this thesis are based on the following publications and papers that are currently under review, for which I am a leading author. The technical contents of Chapters 3 has appeared in Elsevier copyrighted materials, with the permission to reprint granted by Elsevier:

1. Ge, Yan, Pan Peng, and Haiping Lu. "Mixed-Order Spectral Clustering for Complex Networks." Pattern Recognition (2021): 107964.

2. Ge, Yan, Jun Ma, Li Zhang, and Haiping Lu. "Unifying Homophily and Heterophily Network Transformation via Motifs." arXiv preprint arXiv:2012.11400 (2020).

3. Ge, Yan, Jun Ma, Li Zhang, and Haiping Lu. Trustworthiness-Aware Knowledge Graph Representation for Recommendation, ECML-PKDD GEM Workshop (2020).

Chapter 2

Background

Graphs have become ubiquitous because data from many different disciplines can be naturally mapped to graph structures, such as social networks, traffic networks and protein–protein interaction graphs. Therefore, to understand the structure and feature of these complex systems, graph analysis has become both crucial and interdisciplinary. Motivated by it, in Chapter 1, we have shown the importance of motifs and GRL. Therefore, in this chapter, we start with the definition of motifs and GRL to gain insights. After that, we categorise GRL methods based on the techniques used and introduce some basis of SC. Additionally, the proximity as an important concept in GRL will be presented in Sec.2.4. Finally, we will show knowledge graph representation learning and its important application recommender systems.

2.1 Motif Structures in Graphs

Graph analysis in terms of the edge structure connecting two nodes is natural since the edge is assumed to be the simplest unit structure in graphs. However, there is substantial evidence that higher-order structures (Fig. 1.2) that are small subgraph connectivity patterns among several nodes, are essential to understand the behaviour of many complex systems modelled by graphs [76, 125, 10]. Following the paper [10], we formally define a motif on k nodes by a tuple $(\mathbf{B}, A)$, where $\mathbf{B}$ is a $k \times k$ binary matrix and $A \in \{1, 2, ..., k\}$ is a set of anchor nodes. The matrix **B** encodes the edge connection pattern between the k nodes, and A labels a relevant subset of nodes for defining motif conductance. Third-order structures model interactions among three nodes, which are the cases of k =3. For example, the motif M4 in Fig.1.2, where all nodes are anchors is denoted by

$$(\mathbf{B},A) = \left(\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \{1,2,3\} \right). \tag{2.1}$$

Higher-order structures consist of at least three nodes (e.g. triangles, 4-vertex cliques) [10]. It can directly capture interaction among three or more nodes. For example, when clustering graphs, higher-order structures can be regarded as fundamental *units* and algorithms can be designed to minimise cutting them in partitioning. Clustering based on higher-order structures can help us gain new insights and significantly improve our understanding of underlying graphs. For example, triangular structures, with three reciprocated edges connecting three nodes, play important roles in brain networks [98] and social networks [45]. More importantly, higher-order structures allow for more flexible modelling. For instance, considering directions of edges, there exist 13 different third-order structures, but only two different second-order structures [1] [92]. Thus, the application can drive which third-order structures to be preserved.

2.2 Graph Representation Learning

Graphs can be naturally found in a wide diversity of real-world scenarios, such as social networks and citation networks. Effective graph analysis can benefit some graph-related tasks. For example, by analysing a constructed graph derived from users' online social behaviours (e.g., retweet/ comment/follow in Facebook), we can classify users, detect communities, and predict whether an interaction will happen between two users [15].

[1]Edges are considered as first-order structures in [9] but second-order structures in [136]. We follow the terminologies in the latter [136] so that the "order" here refers to the number of nodes involved in a particular structure.

Graph representation learning (GRL), which has attracted a lot of research interest, represents nodes by low-dimensional vectors while preserving graph topology structure, vertex content, and other side information. After we learn new node representations, graph analysis tasks can be easily and efficiently handled by applying traditional vector-based machine learning algorithms (e.g., logistic regression, *k*-means). In essence, the learned node representations remove redundant and noisy information while preserving reasonable and valuable information for downstream graph-related tasks.

In the early 2000s, graph embedding algorithms were mainly designed to reduce the high dimension of the *non-relational* data. Given a set of non-relational high-dimensional data features, a similarity graph (e.g., *k*-nearest neighbour graph) is first constructed based on the pairwise feature similarity. Then, each node in the graph is embedded into a low-dimensional space where connected nodes are closer to each other. Isomap [102] and Locally Linear Embedding [89] are representative methods. However, these methods only focus on the similarity of direct neighbours and usually require at least quadratic time complexity in terms of the number of nodes. Therefore, in recent years, loans of research works give attention to the development of effective and scalable representation learning techniques that are directly designed for graphs. Also, beyond the structure of direct neighbours, preservation of the indirect neighbour structure becomes an attractive research problem. For example, DeepWalk [82] as a pioneering work first proposes to use truncated random walks to collect connection information of nodes and apply the SkipGram model [73] to derive the embedding vectors. The problem of GRL can be formally defined as follows:

Definition 1. *(GRL) Given a graph denoted as $G = (V, E)$, GRL is a mapping function $f : v_i \rightarrow \mathbf{y}_i \in \mathbb{R}^d$,where $d \ll |V|$. The objective function is to make the similarity between $\mathbf{y}_i$ and $\mathbf{y}_j$ explicitly preserve the structural information of v_i and v_j.*

In the following, we categorise GRL methods into three groups that are matrix factorisation based, random walk based and deep neural network. In general, GRL aims to project a graph to a low dimensional space while preserving graph property information as much as possible. The main differences among different categories of GRL rely on the definition of graph property information to be preserved. Note that the below three categories of models are not completely exclusive, and it is possible to combine them for developing algorithms.

2.2.1 Matrix Factorisation based GRL

Matrix factorisation based methods encode the connections information between nodes in the form of a matrix that can be adjacency matrix, Laplacian matrix and transition probability matrix. The matrix is then factorised to obtain the embedding. Additionally, depending on the matrix properties, approaches to factorise the representative matrix is various. For example, if the obtained matrix is positive semi-definite (e.g. the Laplacian matrix), we can use the eigenvalue decomposition. Matrix factorisation based methods have been proved effective in learning node representations [122, 135] because of capturing global structure, but the scalability is a limitation that needs to be concerned. The reason is that performing factorisation on a matrix with millions of rows and columns is memory intensive and computationally expensive.

2.2.2 Random Walk based GRL

From Definition 1, graph structure preservation is essential in graph embedding space. The assumption of random Walk based GRL is that the neighbourhood structure that is local connectivity information of a node is important for GRL. The primitive representation of a node is an adjacency vector that encodes the connection information of direct neighbours. However, this vector is sparse, noisy, and high-dimensional. Specifically, for a real-world large graph with n nodes, we use a vector $\mathbf{i}$ to encode the connection information for a node v_i. This vector is sparse, which means that only few entities in this vector are non-zero due to few connected neighbours of the node v_i. This vector is noisy because the node v_i and its connected neighbours potentially have different given labels for predictions. This vector is high-dimensional because the dimension of vector $\mathbf{i}$ is $1 \times n$. Such a primitive representation is not friendly to downstream applications. DeepWalk [82] is a pioneer work in using random walks to collect a list of neighbour connection information. Based on

DeepWalk, node2vec [46] exploits a biased random walk strategy to capture more rich structural information. Random Walk based GRL methods bring the idea from the field of natural language processing because the word representation learning also suffers from sparse, noise, and high-dimensional problems. After performing random walks, it then uses word2vector [74] to reconstruct its neighbourhood connection information that is defined by co-occurrence rate. The core idea is that regarding a node as a word, we can regard a random path as a sentence, and the node neighbourhood can be identified by co-occurrence rate as in Word2Vector. The following works of DeepWalk, such as node2vec [46] and struc2vec [87], focus on the key problem about the definition of neighbourhood.

2.2.3 Deep Neural Networks based GRL

Deep neural networks have shown superior performance in a wide variety of research fields, such as computer vision and natural language processing. One important reason is that deep neural networks can effectively model the non-linear function. Essentially, by definition the core problem is to learn a function that can effectively map from the original graph space into a low-dimensional vector space. As introduced matrix factorisation based methods in Sec. 2.2.1, these methods assume the mapping function to be linear. Nevertheless, the real-world graphs are complicated and highly non-linear [128]. Due to an effective non-linear function learning model, the deep neural network can be a useful alternation to be considered. Additionally, end-to-end training is an advantage of deep neural networks for GRL. In recent years, graph convolutions networks [56] and their variations that define a convolution operator on networks become popular to learn GRL. These models iteratively aggregate the embeddings of neighbours for a central node and use a non-linear activation function of the obtained embedding. The aggregation operation only focuses on local neighbourhoods that which makes it scalable. Also, multiple iterations allow the learned embedding of a node to capture global information.

2.3 Spectral Clustering

Clustering is an important and powerful tool in analysing graph data, e.g., for community detection [85, 22, 38] and image segmentation [106]. Clustering aims to divide the data set into *clusters* (or *communities*) such that the nodes assigned to a particular cluster are similar or well connected in some predefined sense [55, 77]. It helps us reveal functional groups hidden in data. Spectral clustering is one of the most popular clustering algorithms. It has efficient solutions by standard linear algebra software, and frequently outperforms other popular clustering algorithms such as k-means [108]. As a popular clustering method, conventional spectral clustering (SC) [72, 79] encodes pairwise similarity into an adjacency matrix. Such encoding inherently restricts SC to *second-order structures* [10], such as undirected or directed edges connecting two nodes. However, in many real-world graphs, the minimal and functional structural unit of a graph is not a simple edge but a small network subgraph (a.k.a. *motif*) that involves more than two nodes [9].

Typical steps in a bi-partition SC algorithm is shown in Algorithm 1. Step 1 differentiates higher-order SC from conventional SC on whether an adjacency tensor or matrix is built first from the graph G. This adjacency matrix/tensor is then used to construct a similarity (Laplacian/transition) matrix/tensor. A three-dimensional adjacency tensor $\mathcal{T}$ encodes the connection information among three nodes. If three nodes $\{v_i, v_j, v_k\}$ form a triangle, all entries in $\mathcal{T}$ with a permutation of indices i, j, k are 1 otherwise are 0. For higher-order tensor SC, the similarity tensor is further reduced to a similarity matrix. Then, we compute the dominant eigenvector of the similarity matrix, whose entries correspond to the vertices. Next, a procedure called sweep cut is applied to this eigenvector, where the entries are sorted to construct $(n-1)$ candidate partitions. A sweep cut criterion is then used to evaluate these partitions to find the optimal one as the final output.

In this thesis, we focus on methods based on normalised graph Laplacian and random walks where a transition matrix/tensor is constructed in Step 2 and present more relevant details below.

Algorithm 1 Generic steps in bi-partition spectral clustering

```
Input: $G = (V, E)$
Output: Two node sets $S_1$, $S_2$
1: Construct an adjacency matrix/tensor from $G$.
2: Construct a similarity (Laplacian/transition) matrix/tensor from the adjacency matrix/tensor.
3: [Higher-order tensor SC only:] Reduce the similarity tensor to a similarity matrix.
4: Compute the dominant eigenvector $\mathbf{z}$ of the similarity matrix above. Each entry in $\mathbf{z}$ corresponds to a vertex.
       [Sweep cut]
5: Sort the $n$ entries in $\mathbf{z}$ from $z_1 \leq \cdots \leq z_n$ to $z_{\psi_1} \leq \cdots \leq z_{\psi_n}$, which gets the vertices $V$ sorted to $\{v_{\psi_1}, \ldots, v_{\psi_n}\}$.
6: Let $T_k = \{v_{\psi_1}, \ldots, v_{\psi_k}\}$ consist of the first $k$ sorted vertices, for $1 \leq k \leq n-1$.
7: Use a cut criterion $\beta(T_k, V \setminus T_k)$ to compute all $(n-1)$ possible binary partitions of the sorted vertices.
8: The partition $k^*$ with the optimal criterion value is the output partition $\{S_1, S_2\} \leftarrow (T_{k^*}, V \setminus T_{k^*})$.
```

2.3.1 Normalised Graph Laplacian

Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ be an unweighted adjacency matrix of G where $\mathbf{W}(i, j) = 1$ if $(v_i, v_j) \in E$, otherwise $\mathbf{W}(i, j) = 0$. The degree matrix $\mathbf{D}$ is a diagonal matrix with diagonal entries

$$\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{W}(i, j), \tag{2.2}$$

which is the *degree* of vertex v_i. We denote the *Laplacian matrix* of G as

$$\mathbf{N} = \mathbf{D} - \mathbf{W}. \tag{2.3}$$

The *normalised Laplacian* of G is defined as

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}} \mathbf{N} \mathbf{D}^{-\frac{1}{2}}. \tag{2.4}$$

Let $\mathbf{W}_T$ be *triangle adjacency matrix* of G with its entry $(i,\ j)$ being the number of triangles containing vertices i and j, which leads to a corresponding weighted graph G_T [10]. For implementation, based on [10], we formulate

$$\mathbf{W}_T = \mathbf{A} \cdot \mathbf{A} \circ \mathbf{A}, \tag{2.5}$$

where $\mathbf{A}$ is an edge adjacency matrix, '$\cdot$' is matrix multiplication, and '$\circ$' is Hadamard product. Similarly, we can define the *triangle Laplacian* as $\mathbf{N}_T = \mathbf{D}_T - \mathbf{W}_T$ and the *normalised triangle Laplacian* as

$$\mathbf{L}_T = \mathbf{D}_T^{-\frac{1}{2}} \mathbf{N}_T \mathbf{D}_T^{-\frac{1}{2}}, \tag{2.6}$$

where $\mathbf{D}_T(i,i) = \sum_{j=1}^{n} \mathbf{W}_T(i,j)$.

2.3.2 Random Walks for Structures

We define a second-order transition matrix $\mathbf{P}$ by normalising the adjacency matrix $\mathbf{W}$ to represent edge structures as [72]

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}. \tag{2.7}$$

The entry $\mathbf{P}_{ij}$ represents the probability of jumping from vertex v_i to v_j in one step. The transition matrix $\mathbf{P}$ represents a random walk process on graph G [72]. From the random walk perspective, SC can be interpreted as trying to find a partition of the graph such that the random walk stays long within the same cluster and seldom jumps between clusters [108].

Benson *et al.* [9] extend the above using a three-dimensional transition tensor to encode triangle structures. They firstly define a symmetric *adjacency tensor* $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$ such that the connectivity information for three vertices $\{\mathrm{v}_i,\ \mathrm{v}_j,\ \mathrm{v}_k\} \in V$ can be represented explicitly in this tensor. All entries in $\mathcal{T}$ with a permutation of indices i, j, k have the same value (hence symmetric). Thus, $\mathcal{T}$ encodes triangle structures in G as:

$$\mathcal{T}(i,j,k) = \begin{cases} 1 & v_i, v_j, v_k \text{ form a triangle,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Next, they form a third-order transition tensor $\mathcal{P}$ as:

$$\mathcal{P}(i,j,k) = \mathcal{T}(i,j,k) / \sum_{m=1}^{n} \mathcal{T}(i,m,k), \tag{2.9}$$

where $\sum_{m=1}^{n} \mathcal{T}(i,m,k) \neq 0$, and $1 \leq i,\ j,\ k \leq n$. For $\sum_{m=1}^{n} \mathcal{T}(i,m,k) = 0$, we follow Benson *et al.* [9] and fill in $\mathcal{P}(i,:,k)$ with a vector $\mathbf{u}$ where every value of entity is $\frac{1}{n}$, where n is the number of nodes. The vector $\mathbf{u}$ is a dangling distribution vector. Benson *et al.* [9] interpret this transition tensor $\mathcal{P}$ as a second-order random walks. The reason for this setting is that it can prevent information from getting stuck on the nodes that do not involve in any triangular motifs and eventually resulting information received by these nodes cannot be propagated. For this case, the vector $\mathbf{u}$ ensures that the information can be randomly propagated from the node that does not involve in any triangular to any other nodes with the same probability $\frac{1}{n}$.

Based on the strategies for constructing different similarity matrices in Sec. 2.3.1 and Sec. 2.3.2, the existing higher-order SC can be grouped into four approaches: 1) The first approach constructs an affinity tensor to encode higher-order structures and then reduces it to a matrix [39], followed by conventional SC [79]. These methods, such as tensor trace maximisation (TTM) [40], are developed in a closely related problem of hypergraph clustering that considers "hyperedges" connecting multiple nodes. 2) The second approach develops higher-order SC by constructing a transition tensor based on random walks model and then reduces it to a matrix for conventional SC, such as tensor spectral clustering (TSC) [9]. 3) The third approach uses a counting and reweighting scheme to capture higher-order structures and reveal clusters [105], such as motif-based SC (MSC) [10].[2] 4) The fourth approach is higher-order local clustering aiming to reduce computation cost [126], such as High-Order Structure Preserving Local Clustering (HOSPLOC) [136].

[2]We have verified that TTM and MSC are equivalent, nevertheless.

2.3.3 Sorting the Nodes

We hope to find a binary partition, i.e., a cut, that can optimise some cut criterion (as listed in Table 2.1). However, this is an NP-hard combinatorial optimization problem [109]. Fortunately, some real-valued relaxations of this problem provide a guaranteed approximation [26]. One of the most popular approach is to calculate a domain eigenvector $\mathbf{z} = \{z_1, z_2, \ldots, z_n\}$, called Fiedler vector [34], of the similarity matrix. We can sort the vertices according to the values $\{z_i\}$. Then, we consider the vertex subsets $T_k = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_k\}$, $1 \leq k \leq n-1$, i.e., T_k consists of the first k vertices in the sorted vertex list and becomes a candidate cluster, with its complement $V \setminus T_k$ as the other cluster. Next, we use a sweep cut procedure to evaluate this partition using some cut criterion $\beta(\cdot)$ to find the best one.

2.3.4 Cut Criteria

The sweep cut procedures are described in steps 5 to 8 of Algorithm 1. In the following, we describe some commonly used cut criteria (i.e., the function $\beta(\cdot)$) to be studied in this thesis. Firstly, we introduce some essential definitions.

Edge volume, associativity, and cut. The (edge) *volume* of a set of vertices S_1, denoted by $vol_2(S_1)$, is the sum of the degrees of vertices in S_1:

$$vol_2(S_1) = \sum_{v_i \in S_1, v_j \in V} \mathbf{W}_{ij}. \tag{2.10}$$

The (edge) associativity of vertices in S_1, denoted by $assoc_2(S_1)$, is the sum of the degrees of vertices in the subgraph induced by vertices in S_1, defined as

$$assoc_2(S_1) = \sum_{v_i, v_j \in S_1} \mathbf{W}_{ij}. \tag{2.11}$$

The (edge) *cut*, denoted by cut_2 (S_1, S_2), is the number of edges between S_1 and S_2

$$cut_2(S_1, S_2) \quad = \quad vol_2(S_1) - assoc_2(S_1). \tag{2.12}$$

These are conventional definitions [108].

Triangle volume, associativity, and cut. Benson *et al.* [9] extended the above definitions to 3D. The triangle *volume* of S_1, denoted by $vol_3(S_1)$, counts the number of vertices in triangles that reside in S_1:

$$vol_3(S_1) = \frac{1}{2} \sum_{v_i \in S_1; v_j, v_k \in V} \mathcal{T}(i,j,k). \tag{2.13}$$

The triangle associativity, denoted by $assoc_3(S_1)$, counts the number of vertices in triangles in the subgraph induced by vertices in S_1, defined as

$$assoc_3(S_1) = \frac{1}{2} \sum_{v_i, v_j, v_k \in S_1} \mathcal{T}(i,j,k). \tag{2.14}$$

The triangle *cut*, denoted by $cut_3(S_1,S_1)$ is the number of triangles between S_1 and S_2.

$$\begin{aligned} cut_3(S_1,S_2) \quad = \quad & \frac{1}{3}(((vol_3(S_1) - assoc_3(S_1)) \\ & +(vol_3(S_2) - assoc_3(S_2))). \end{aligned} \tag{2.15}$$

In the above, we introduce some factors $1/2$ and $1/3$ to the original definitions in [9] to avoid counting vertices in triangles more than once in an undirected graph.

Table 2.1 lists eight cut criteria based on the above definitions, to be studied in this thesis. These sweep cut criteria have two main objectives: 1) to preserve a rich set of structures in S_1 and S_2; 2) to avoid breaking (as many as possible) structures due to partitioning S_1 and S_2. Besides these sweep cut methods, k-means [70] is another popular choice in finding the final partition by clustering the first two eigenvectors into two sets.

2.3.5 Cheeger Inequalities

Given $G = (V,E)$ and a subset $S \subseteq V$, let $\bar{S}$ denote the complement of S. Let $cut_2(S;G)$ denote the *edge cut* of S, i.e, the number of edges between S and $\bar{S}$ in G. Let $vol_2(S;G)$ denote the

Table 2.1 Edge-based and triangle-based cut criteria.

	Edge-based cuts	**Triangle-based cuts**
Conductance (ϕ)	$\phi_2(S) = \frac{cut_2(S)}{\min(vol_2(S), vol_2(\bar{S}))}$ [91]	$\phi_3(S) = \frac{cut_3(S)}{\min(vol_3(S), vol_3(\bar{S}))}$ [9]
Ncut (η)	$\eta_2(S) = cut_2(S)(\frac{1}{vol_2(S)} + \frac{1}{vol_2(\bar{S})})$ [95]	$\eta_3(S) = cut_3(S)(\frac{1}{vol_3(S)} + \frac{1}{vol_3(\bar{S})})$ [63]
Nassoc (ξ)	$\xi_2(S) = \frac{assoc_2(S)}{vol_2(S)} + \frac{assoc_2(\bar{S})}{vol_2(\bar{S})}$[95]	$\xi_3(S) = \frac{assoc_3(S)}{vol_3(S)} + \frac{assoc_3(\bar{S})}{vol_3(\bar{S})}$[40]
Expansion (α)	$\alpha_2(S) = \frac{cut_2(S)}{\min(\lvert S\rvert, \lvert\bar{S}\rvert)}$ [35]	$\alpha_3(S) = \frac{cut_3(S)}{\min(\lvert S\rvert, \lvert\bar{S}\rvert)}$ [9]

edge volume of S, i.e. the total degrees of vertices in S, and $assoc_2(S;G)$ is the total degrees in the subgrapgh induced by vertices in S. The *edge conductance* of S is defined as

$$\phi_2(S;G) = \frac{cut_2(S;G)}{\min\{vol_2(S;G), vol_2(\bar{S};G)\}}. \tag{2.16}$$

Other popular edge-based cut criteria are shown in Table 2.1 (left column). The classical Cheeger inequality below relates the conductance of the sweep cut of SC to the minimum conductance value of the graph [34].

Lemma 1 (Second-Order Cheeger Inequality [34]). *Let* $\mathbf{v}$ *be the second smallest eigenvector of* $\mathbf{L}$. *Let* T^* *be the sweep cut of* $\mathbf{D}^{-1/2}\mathbf{v}$ *w.r.t. cut criterion* $\phi_2(\cdot;G)$. *It holds that*

$$\phi_2(T^*;G) \leq 2\sqrt{\phi_2^*}, \tag{2.17}$$

where $\phi_2^* = \min_{\mathrm{S}\subset\mathrm{V}}\ \phi_2(\mathrm{S};G)$ *is the minimum conductance over any set of vertices.*

Let $cut_3(S;G)$ denote the *triangle cut* of S, i.e. the number of triangles that have at least one endpoint in S and at least one endpoint in $\bar{S}$, and let $assoc_3(S;G)$ count the number of vertices in triangles in the subgraph induced by vertices in S. Let $vol_3(S;G)$ denote the *triangle volume* of S, i.e. the number of triangle endpoints in S. The *triangle conductance* [9] of S is defined as

$$\phi_3(S;G) = \frac{cut_3(S;G)}{\min\{vol_3(S;G), vol_3(\bar{S};G)\}}. \tag{2.18}$$

It is further proved in [10] that for any $S \subset V$, $\phi_3(S;G) = \phi_2(S;G_T)$, which leads to the following third-order Cheeger inequality. Other popular triangle-based cut criteria are summarised in Table 2.1 (right column).

Lemma 2 (Third-order Cheeger Inequality [10]). *Let* $\mathbf{v}$ *be the second smallest eigenvector of* $\mathbf{L}_T$. *Let* T^* *denote the sweep cut of* $\mathbf{D}_T^{-1/2}\mathbf{v}$ *w.r.t. cut criteria* $\phi_2(\cdot;G_T)$. *It holds that*

$$\phi_3(T^*;G) \leq 4\sqrt{\phi_3^*}, \tag{2.19}$$

where $\phi_3^* = min_{\mathrm{S}\subset\mathrm{V}}\ \phi_3(\mathrm{S};G)$.

2.4 Proximity in Graph Representation Learning

2.4.1 Higher-order Proximity

Graph embedding (a.k.a. network embedding) learns low-dimensional latent representations of nodes while preserving the structure and inherent properties of the graph. It has been successfully applied in node classification [56, 107, 37], link prediction [135, 100], and community detection [114, 20].

Graph embedding aims to learn latent, low-dimensional representation of nodes while preserving graph topology [128]. Prior works have demonstrated that the higher-order proximities between nodes are of tremendous importance in capturing the underlying structure of the graph [17, 80, 123, 33]. The adjacency matrix $\mathbf{A}$ can be treated as the first-order proximity, which captures the pairwise proximity between nodes. However, the first-order proximity is very sparse and insufficient to fully model the relationships between nodes in most cases. In order to characterise the connections between nodes better, HOP is widely studied. Given **A**, the HOP can be defined as a polynomial function of **A** [135]:

$$\mathbf{S} = w_1\mathbf{A} + w_2\mathbf{A}^2 + \ldots + w_l\mathbf{A}^l, \tag{2.20}$$

where l is the order, and $w_1, ..., w_l$ are the weights for each term. The $\{\ \mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_l\ \}$ are hyperparameters, and users can define them. In the paper [135], the authors recommend a setting $\mathbf{w}_i = 0.1^i$. Matrix $\mathbf{A}^l$ denotes the lth-order proximity matrix, with a multiplication of l matrices **A**. The lth-order proximity value between nodes v_i and v_j is denoted as $a_{ij}^{(l)}$.

2.4.2 Homophily and Heterophily Proximity

Preserving higher-order proximity (HOP) instead of only considering direct neighbourhood relationship (e.g., adjacency matrix) has been shown to be effective for graph embedding since it can capture rich underlying structures of graphs [17, 101, 135]. In the context of the graph embedding, homophily is an assumption that highly inter- connected nodes that are resident in the same community should be placed tightly in the embedding space [71]. To encode homophily information, we use a homophily proximity **K** that is defined as:

Definition 2. *(**Homophily Proximity**) Given a graph denoted as $G = (V, E)$, homophily proximity is a square matrix* **K** *of G such that an entry* $\mathbf{K}(i, j)$ *indicates a level that node v_i and node v_j are resident in the same community.*

Most graph embedding methods share the homophily assumption. However, heterophily is also an important assumption that places distant but structurally similar nodes (e.g., bridge [6]) together after embedding. It can benefit the role classification task. Here we use heterophily proximity **N** to encode heterophily information of a graph G, and it is defined as:

Definition 3. *(**Heterophily Proximity**) Given a graph denoted as $G = (V, E)$, heterophily proximity is a square matrix* **N** *of G such that an entry* $\mathbf{N}(i, j)$ *indicates a level of an edge e_{ij} to connect nodes acting as roles and to connect different communities.*

In Definition 3, the level of an entry $\mathbf{N}(i, j)$ is the weight of an edge e_{ij}. This weight indicates how likely this edge to bridge different communities. To preserve HOP under the homophily assumption, DeepWalk [82] employs random walks to generate node sequences analogous to word sentences, and then the HOP is approximately captured by a Skip-gram model [74]. Graph convolutional network (GCN) [56] aggregates the features of local neighbours for central node representation so that feature vectors of nodes within the same community are more similar than those in different communities. Arbitrary-order proximity embedding (AROPE) [135] effectively and accurately preserves arbitrary-order proximity by reweighting the eigen-decompostion. To preserve HOP under the heterophily assumption, struc2vec [87] first encodes the node structural similarity into a multi-layer graph, and then DeepWalk is performed on this multi-layer graph to learn node representations. A recent transformation model graph diffusion convolution (GDC) [57] generates a new graph by constructing a diffusion graph obtained by a polynomial function, and then sparsify this diffusion graph by setting a threshold, but still overlooks the heterophily assumption.

2.5 Knowledge Graph Representation for Recommendation

2.5.1 Knowledge Graph Representation

In this thesis, we investigate the GRL for homogeneous graphs that all nodes belong to a single type and all edges belong to one single type, such as the Zachary Karate Club network in Fig. 1.1. Additionally, we study the GRL for heterogeneous graphs that both node and edge types are larger than one. Knowledge graphs (KGs) contain rich knowledge in the form of heterogeneous graphs where nodes correspond entities and edges correspond to relations. Knowledge in KGs is presented as in the form of the triple (head entity, relation, tail entity) [113]. For example, in Fig. 2.1, (Tom Hanks, Star, Forrest Gump) indicates that Tom Hanks is a star of Forrest Gump.

KGR is used to embed entities and relations into low-dimensional vectors while preserving the semantic and structural information [53]. Translational models are popular to exploit distance-based energy functions and a relation is regarded as a translation in the embedding

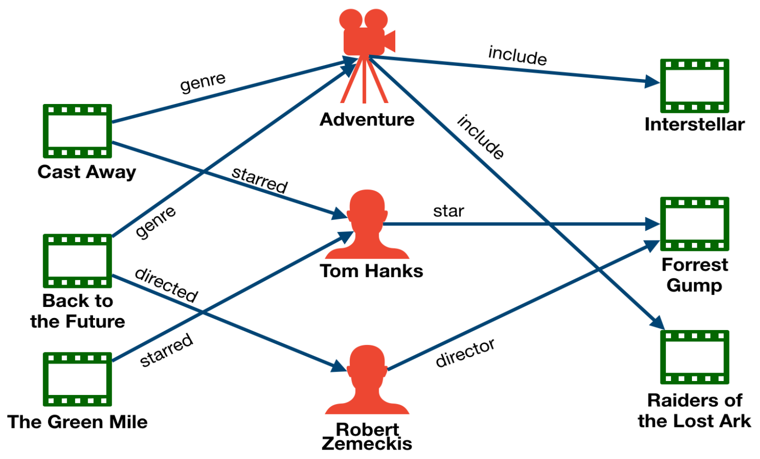


Fig. 2.1 An example of a knowledge graph that contains entities and relations.

space. TransE [13] follows an assumption that e_h and e_t are connected by r with low error if a triple (e_h, r, e_t) holds, and thus formulates an energy function

$$g_E = ||\mathbf{e}_h + \mathbf{r} - \mathbf{e}_t||. \tag{2.21}$$

However, TransE has flaws when dealing with 1-to-N, N-to-1 and N-to-N relations. To address these issues, TransH [115] introduces relation specific hyperplanes, which each relation r as a vector $\mathbf{r}$ on a hyperplane with $\mathbf{w}_r$ The embeddings $\mathbf{e}_h$ and $\mathbf{e}_t$ are first projected to the hyperplane of relation r to obtain vectors. They are defined as

$$\mathbf{e}_h^{\perp} = \mathbf{e}_h - \mathbf{w}_r^{\perp}\mathbf{e}_h\mathbf{w}_r \;,\; \mathbf{e}_t^{\perp} = \mathbf{e}_t - \mathbf{w}_r^{\perp}\mathbf{e}_t\mathbf{w}_r. \tag{2.22}$$

We then calculate $\mathbf{e}_h^{\perp} + \mathbf{r} \approx \mathbf{e}_t^{\perp}$. For TransE and TransH, the embeddings of entities and relations are in the same space. However, entities and relations are different types objects, it is insufficient to model them in the same space. To address this issue, In TransR [66], e_h and e_t are projected to a new space so that the relation r focuses on through a matrix $\mathbf{M}_r$ and then

$$g_R = ||\mathbf{M}_r\mathbf{e}_h + \mathbf{r} - \mathbf{M}_r\mathbf{e}_t||. \tag{2.23}$$

TransD [52] constructs dynamic mapping matrices as

$$\mathbf{M}_{rh} = \mathbf{r}_p\mathbf{h}_p + \mathbf{I}\ ,\ \mathbf{M}_{rt} = \mathbf{r}_p\mathbf{t}_p + \mathbf{I}. \tag{2.24}$$

We then construct projection vectors $\mathbf{h}_p, \mathbf{t}_p, \mathbf{r}_p \in \mathbb{R}^n$ and an identity matrix $\mathbf{I} \in \mathbb{R}^{n\times n}$, with the formulation as

$$g_D = || \left(\mathbf{r}_p\mathbf{h}_p^\top + \mathbf{I}\right)\mathbf{h} + \mathbf{r} - \left(\mathbf{r}_p\mathbf{t}_p^\top + \mathbf{I}\right)\mathbf{t}||. \tag{2.25}$$

2.5.2 Knowledge Graphs for Recommender Systems

The explosive growth of media services has provided overwhelming choices for users, such as movies, music and series. Recommender systems (RS) aim to ease information explosion and largely reduce users' effort in finding items of interest. Collaborative filtering (CF) is popular for recommender systems, which assumes that behaviourally similar users would have a similar preference on items [51]. To build RS, CF models users' preference on items based on historical interactions. CF recommends items to users by learning user/item similarities from existing ratings. MF [58] is a popular technique, which decomposes a rating matrix into two-factor matrices, representing latent factors for users and items respectively. It is formulated as

$$\hat{y}_{ui} = \mathbf{p}_u^T\mathbf{q}_i, \tag{2.26}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ denote the latent vector for user u and item i. Bayesian personalised ranking (BPR) optimises the above equation with a pairwise ranking loss [86]

$$\mathcal{L}_r = \sum_{(u,i)\in\mathbf{Y},(u,i')\in\mathbf{Y}'} -\log\delta(\hat{y}_{ui} - \hat{y}_{ui'}), \tag{2.27}$$

where $\delta(\cdot)$ is a sigmoid function and $\mathbf{Y}'$ contains negative interactions by randomly corrupting an interacted item to a non-interacted one for each user. It encourages that the interacted items are smaller than random ones for each user through BPR loss function MF associates each user and item with a real-valued vector of latent features. Conventional RS, based on collaborative filtering (CF) [58], usually suffer from the sparsity of interactions and the cold-start problem. To address these issues, CF-based methods usually suffer from the cold-start problem and have troubles in recommending brand new items that have not yet been heavily explored by users. The sparsity problem can be addressed by incorporate auxiliary sources as side information, such as social networks [50] and images [129]. In this thesis, we incorporate KGs as an auxiliary source into CF-based methods to address the above two issues. Recently, some KGs (e.g., Freebase [12]) are successfully applied to many applications such as question answering [31], text classification [113].

Inspired by the success of applying KGs in a variety of tasks, some recent works incorporate KGs into RS. The usage of KGs within the context of RS can alleviate the item cold-start and sparsity problem of CF. The reason is twofold: (1) KGs introduce extra semantic connections among items, which can provide new items with more interactions to recommendations; (2) KGs consist of a variety of relation types, which helps extend a user's interests reasonably and increasing the diversity of recommended items. Moreover, KG can bring explainability to recommender systems since KG connects a user's historical records and the recommended ones based on relations in the KG.

To construct RS with KGs, we have a knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$, which is comprised of massive entity-relation-entity triples (e_h, r, e_h), in which h $\in \mathcal{E}$, r $\in \mathcal{R}$, and t $\in \mathcal{E}$ denote the head, relation, and tail of a knowledge triple, $\mathcal{E}$ and $\mathcal{R}$ are the set of entities and relations in the knowledge graph, respectively. For example, the triple (*Forrest Gump, film.film.director, Robert Zemeckis*) states the fact that Robert Zemeckis is the director of the film "Forrest Gump". In many recommendation scenarios, an item $v \in \mathcal{V}$ corresponds to an entity e $\in \mathcal{E}$ (e.g., item "Forrest Gump" in MovieLens also appears in the knowledge graph as an entity). The set of entities $\mathcal{E}$ is composed from items $\mathcal{V}$ ($\mathcal{V} \in \mathcal{E}$) and non-items $\mathcal{E}/\mathcal{V}$ (e.g., entities corresponding to item properties). We construct a directed graph G from a KG $\mathcal{G}$. Each entity $e \in \mathcal{E}$ is abstracted into a node. If there is a relation from the entities e_1 to e_2, a directed edge will exist from node e_1 to e_2. Therefore, a KG with n entities can be mapped as a directed graph G with n nodes. Given user-item interaction matrix $\mathbf{Y}$ and knowledge graph $\mathcal{G}$, our task is to predict whether user u has a potential interest in item v with which this user has not engaged before. Specifically, we aim to learn a prediction function

$$\hat{y}_{uv} = \mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathcal{G}), \tag{2.28}$$

where $\hat{y}_{uv}$ denotes the probability that user u will engage with item v, and Θ are model parameters of the function $\mathcal{F}$.

Therefore, collaborative knowledge-based embedding (CKE) [129] combines CF with KG embedding in a unified Bayesian framework. CFKG [133] combines user-item interactions and KG into one graph. It then uses an existing KGR method and CF-based method to learn user and item representations. Knowledge translation-based user preference model [18] transfers relation information from a KG to recommendation for better understanding the reasons that a user likes an item. Knowledge-aware graph neural network (GNN) with label smoothness regularisation [111] applies GNN architecture to KGs by using a user-specific relation score function and aggregating neighbourhood information with different weights.

2.5.3 Trustworthiness in Knowledge Graphs

Most traditional knowledge graph construction methods usually involve huge human supervision or expert annotation, which are extremely labour-intensive and time-consuming [119]. Recently, large-scale knowledge graphs (e.g., DBpedia [8], Freebase [12]) are productively and automatically constructed from unstructured web text (e.g., NELL [19]). However, some noises and errors are inevitably introduced in the process of automation due to limited labour supervision [49, 65].

Existing KG-based tasks (e.g., knowledge completion [64]) or applications (e.g., question answering [68]) assume knowledge in the existing KG is completely correct. To model errors in KGs, Xie *et al.* [119] proposed a triple confidence awareness knowledge representation learning framework, which detects possible noises in KGs while learning knowledge representations with confidence simultaneously. They introduced the triple confidence to conventional translation-based methods for knowledge representation learning. Jia *et al.* [54] synthetically extracted trustworthiness of the triples from knowledge graph embedding, entity resource and path information of the knowledge graph. Most KGs representations consider deterministic KGs (e.g., Freebase) that consist of deterministic facts. Chen *et al.* [24] proposed a KGs embedding model on uncertain KGs that associate every fact with a confidence score. Dong *et al.* [32] built a large-scale uncertain knowledge graph, and fused multiple extraction sources with prior knowledge derived from an existing knowledge base.

2.6 Summary

In this chapter, we conduct a review of the literature in the motifs and GRL. We provide a formal definition to the problem of GRL and introduce some basic concepts. Additionally, we present state-of-the-art GRL algorithms in the data mining and machine learning field, which will help to introduce our proposed algorithms in the following chapters.

Chapter 3

Mixed-order Spectral Clustering for Graphs

3.1 Introduction

As discussed the importance of motif structures in the Sec. 1.1, there are emerging interests in directly modelling higher-order structures in graph clustering. However, it should be noted that most graphs have both second-order and higher-order structures, and both can be important. Existing conventional and third-order SC methods model only either second-order or third-order structures, but *not both*. Second-order SC does not take triangles into consideration, while third-order SC loses information of some edges, in particular, those that do not belong to any triangle. A simple example is given by the graph in Fig. 3.1a, which contains both edges and triangles. In Figs. 3.1b and 3.1c, which correspond to the representations used by second-order and third-order SC, respectively, each entry indicates the number of edges and triangles involving two nodes of Fig. 3.1a. As the figures show: second-order SC fails to capture the importance between nodes 2 and 3, but they participate in more triangles than any other two adjacent nodes (say 1 and 2), which is not reflected in Fig. 3.1b; Third-order SC fails to model the importance of the relation between nodes 4 and 5, but there does exist an edge between them (and thus is more important than any two non-adjacent nodes, say nodes 2 and 5), which was missed in Fig. 3.1c.

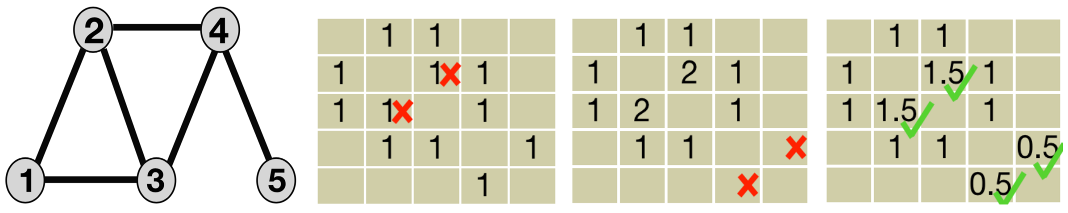


Fig. 3.1 Motivation of mixed-order structures: the second and third order structures in (a) can not be fully captured by edge/triangle adjacency matrix in (b) or (c). A mixed adjacency matrix in (d) can capture both.

In this thesis, we propose a novel *Mixed-Order Spectral Clustering* (MOSC) framework to preserve structures of different orders simultaneously, as in Fig. 3.1d. For clear and compact presentation, we focus on two *undirected unweighted* structures: edges (second-order structures) and triangles (third-order structures). Further extensions can be developed for mixing more than two orders, and/or orders higher than three.

The MOSC framework can be decomposed into three blocks, and we summarise our contributions of this chapter based on building blocks as follows:

1. **Mixed-order structure**. We propose two mixed-order structure approaches: one based on Graph Laplacian (GL) and the other based on Random Walks (RW). Based on these approaches, we develop two new algorithms MOSC-GL and MOSC-RW. MOSC-GL combines edge and triangle adjacency matrices to define a mixed-order Laplacian, with its theoretical performance guarantee derived by proving a mixed-order Cheeger inequality. MOSC-RW combines first-order and second-order RW models for a probabilistic interpretation. Besides, mixing parameter (ranging from 0 to 1) can be automatically decided based on cut criteria and triangle density.

2. **Mixed-order cut criterion**. To enable existing single-order SC methods [95, 10] to preserve mixed-order structures, we consider cut criteria of different orders from the order used to encode a structure (e.g. second-order SC with a third-order cut criterion). We then empirically study the effectiveness of mixed-order cut criterion, finding that this strategy can enhance the performance of conventional SC methods.

3. **Mixed-order evaluation metric**. Given ground truth, existing works only consider the number of error nodes to evaluate the quality of output clusters [126, 40]. However, it may fail to reflect the errors in structures. To address this issue, we propose structure-aware error metrics to evaluate performance at the level of structures. Additionally, we design mixed-order evaluation metrics by further utilising proposed metrics of different orders from the order used to encode a structure (e.g. evaluate second-order SC with a third-order-aware error metric), which aims to gain new insights on the quality of structure preservation.

3.2 Methodology

To model both edge and triangle structures simultaneously, we introduce a new Mixed-Order SC (MOSC) approach, with two methods based on Graph Laplacian (GL) and Random Walks (RW). MOSC-GL combines the edge and triangle adjacency matrices, which leads to a mixed-order Cheeger inequality to provide a theoretical performance guarantee. MOSC-RW is developed under the random walks framework to combine the first and second order random walks, providing a probabilistic interpretation. Moreover, we define new mixed-order cut criteria to enable existing single-order SC methods to preserve mixed-order structures, and propose mixed-order evaluation metrics to evaluate clustering methods at the level of structures. The proposed MOSC framework is illustrated in Fig. 3.2.

3.2.1 MOSC Based on Graph Laplacian (MOSC-GL)

MOSC-GL introduces a *mixed-order adjacency matrix* $\mathbf{W}_X$ that linearly combines the edge adjacency matrix $\mathbf{W}$ and the triangle adjacency matrix $\mathbf{W}_T$, with a *mixing parameter* $\lambda \in [0,1]$. $\mathbf{W}_X$ can be seen as a weighted adjacency matrix of a weighted graph G_X, on which we can apply conventional SC (Algorithm 1). Specifically, we first construct the matrix $\mathbf{W}_X$ and the corresponding diagonal degree matrix $\mathbf{D}_X$ as

$$\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}, \tag{3.1}$$

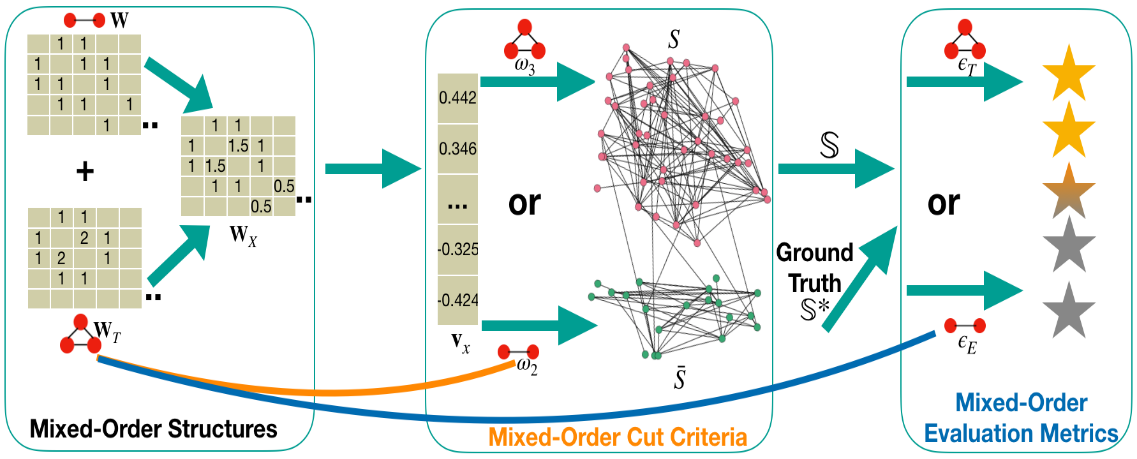


Fig. 3.2 Illustration of mixed-order structures (i.e. MOSC-GL), cut criteria and evaluation metrics. The left shows a mixed-order adjacency matrix $\mathbf{W}_X$ that linearly combines an edge-based matrix ($\mathbf{W}$) and a triangle-based matrix ($\mathbf{W}_T$). The middle shows output clusters $\mathbb{S} = \{S, \bar{S}\}$ generated by either a second-order (ω_2) or third-order cut criterion (ω_3). An orange line from the left to the middle indicates an instance of a mixed-order cut criterion where a sorted dominant eigenvector ($\mathbf{v}_X$) derived from only triangle-based adjacency matrix is split by an edge-based cut criterion. The right shows that given the ground-truth, we evaluate the quality of output clusters $\mathbb{S} = \{S, \bar{S}\}$ by either a second-order (ε_E) or a third-order structural error metric (ε_T). A blue line from the left to right indicates an instance of a mixed-order evaluation metric where output clusters $\mathbb{S}$ derived from only a triangle-based adjacency matrix ($\mathbf{W}_T$) are evaluated by an edge-based error metric (ε_E).

$$\mathbf{D}_X = (1-\lambda)\mathbf{D}_T + \lambda\mathbf{D}. \tag{3.2}$$

Let G_X denote an undirected weighted graph with adjacency matrix $\mathbf{W}_X$, as illustrated in Fig. 3.2 (the left block). We can define a mixed-order Laplacian $\mathbf{N}_X$ and its normalised version $\mathbf{L}_X$ as

$$\mathbf{N}_X = \mathbf{D}_X - \mathbf{W}_X = (1-\lambda)\mathbf{N}_T + \lambda\mathbf{N},$$
$$\mathbf{L}_X = \mathbf{D}_X^{-\frac{1}{2}}\mathbf{N}_X\mathbf{D}_X^{-\frac{1}{2}}. \tag{3.3}$$

Then, we compute the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}_X$ and perform the sweep cut to find the partition with the smallest edge conductance in G_X. The MOSC-GL algorithm is summarised in Algorithm 2.

When $\lambda = 1$, MOSC-GL is equivalent to SC by Ng *et al.* [79] and only considers second-order structures. When $\lambda = 0$, MOSC-GL is equivalent to motif-based SC [10]. MOSC-GL

Algorithm 2 MOSC-GL

```
Input: $G = (V, E)$, a mixing parameter $\lambda$
Output: Two node sets $\{S, \bar{S}\}$
 1: Construct the edge adjacency matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$.
 2: Construct the triangle adjacency matrix $\mathbf{W}_T \in \mathbb{R}^{n\times n}$.
 3: Let $\mathbf{D}$ be diagonal with $\mathbf{D}(i,i) = \sum_i^n \mathbf{W}(i,j)$.
 4: Let $\mathbf{D}_T$ be diagonal with $\mathbf{D}_T(i,i) = \sum_i^n \mathbf{W}_T(i,j)$.
 5: $\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}$.
 6: $\mathbf{D}_X = (1-\lambda)\mathbf{D}_T + \lambda\mathbf{D}$.
 7: $\mathbf{N}_X = \mathbf{D}_X - \mathbf{W}_X = (1-\lambda)\mathbf{N}_T + \lambda\mathbf{N}$.
 8: $\mathbf{L}_X = \mathbf{D}_X^{-\frac{1}{2}}\mathbf{N}_X\mathbf{D}_X^{-\frac{1}{2}}$.
 9: Compute the second smallest eigenvector $\mathbf{v}_X$ of $\mathbf{L}_X$.
10: $\mathbf{v}_X \leftarrow$ Sort entries of $\mathbf{D}_X^{-\frac{1}{2}}\mathbf{v}_X$.
11: $\{S, \bar{S}\} \leftarrow$ Sweep cut on $\mathbf{v}_X$ w.r.t. some cut criteria.
```

maintains the advantages of traditional SC: computational efficiency, ease of implementation and mathematical guarantee on the near-optimality of resulting clusters, which we formalise and prove in the following.

Performance Guarantee. Given a graph G and a vertex set S, we define its mixed-order cut and volume as

$$cut_X(S;G) = (1-\lambda)cut_3(S;G) + \lambda cut_2(S;G), \tag{3.4}$$

and

$$vol_X(S;G) = (1-\lambda)vol_3(S;G) + \lambda vol_2(S;G), \tag{3.5}$$

respectively. Then, we define the *mixed-order conductance* of S as:

$$\phi_X(S;G) = \frac{cut_X(S;G)}{\min(vol_X(S;G), vol_X(\bar{S};G))}, \tag{3.6}$$

which generalises edge and triangle conductance. A partition with small $\phi_X(S;G)$ corresponds to clusters with rich edge and triangle structures within the same cluster while few both structures crossing clusters. Finding the exact set of nodes S with the smallest ϕ_X is computationally intractable. Nevertheless, we can derive a performance guarantee for MOSC-GL to show that the output set obtained from Algorithm 2 is a good approximation. To prove Theorem 1, we need the following Lemma.

Lemma 3 (Lemma 4 and 1 in [10]). *Let G = (V, E) be an undirected, unweighted graph and let G_T be the weighted graph for the triangle adjacency matrix. Then for any $S \subset V$, it holds that*

$$cut_3(S;G) = \frac{1}{2} cut_2(S;G_T), \tag{3.7}$$

$$vol_3(S;G) = \frac{1}{2} vol_2(S;G_T). \tag{3.8}$$

Theorem 1 (Mixed-order Cheeger Inequality). *Given an undirected graph G, let T^* denote the set outputted by MOSC-GL w.r.t. the cut criterion $\phi_2(\cdot;G_X)$. Let $\phi_X^* = \min_{S\subseteq V} \phi_X(S;G)$ be the minimum mixed-order conductance over any set of vertices. Then it holds that*

$$\phi_X(T^*;G) \leq 4\sqrt{2\phi_X^*}. \tag{3.9}$$

Proof. It suffices for us to prove that for any set S,

$$\frac{1}{2}\phi_2(S;G_X) \leq \phi_X(S;G) \leq 2\phi_2(S;G_X). \tag{3.10}$$

Assume for now that the above inequality (3.10) holds. By Lemma 1, the set T^* satisfies

$$\phi_2(T^*;G_X) \leq 2\sqrt{\psi^*}, \tag{3.11}$$

where $\psi^* = \min_{S\subseteq V} \phi_2(S;G_X)$. Let R be the set with $\phi_X(R;G) = \phi_X^* = \min_{S\subseteq V} \phi_X(S;G)$. Then by inequality (3.10), we have

$$\phi_X(T^*;G) \leq 2\phi_2(T^*;G_X) \leq 4\sqrt{\psi^*} \leq 4\sqrt{\phi_2(R;G_X)} \leq 4\sqrt{2\phi_X(R;G)} = 4\sqrt{2\phi_X^*}. \tag{3.12}$$

This will then finish the proof. Therefore, we only need to prove the inequality (3.10). We will make use of the Lemma 3 from [10].

By Lemma 3, we have

$$cut_X(S;G) = (1-\lambda)cut_3(S;G) + \lambda cut_2(S;G) = (1-\lambda)\frac{1}{2}cut_2(S;G_T) + \lambda cut_2(S;G),$$

$$vol_X(S;G) = (1-\lambda)vol_3(S;G) + \lambda vol_2(S;G) = (1-\lambda)\frac{1}{2}vol_2(S;G_T) + \lambda vol_2(S;G).$$

Since the adjacency matrix of G_X is a linear combination of the adjacency matrix of G_T and the adjacency matrix of G, i.e.

$$\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}, \tag{3.13}$$

we have

$$cut_2(S;G_X) = (1-\lambda)cut_2(S;G_T) + \lambda cut_2(S;G), \tag{3.14}$$

$$vol_2(S;G_X) = (1-\lambda)vol_2(S;G_T) + \lambda vol_2(S;G). \tag{3.15}$$

The above equations imply that for any set S,

$$\frac{1}{2}cut_2(S;G_X) \le cut_X(S;G) \le cut_2(S;G_X), \tag{3.16}$$

$$\frac{1}{2}vol_2(S;G_X) \le vol_X(S;G) \le vol_2(S;G_X). \tag{3.17}$$

The last inequality also implies that for any S,

$$\frac{1}{2}vol_2(\bar{S};G_X) \le vol_X(\bar{S};G) \le vol_2(\bar{S};G_X). \tag{3.18}$$

Therefore, by the definition of $\phi_X(S;G)$, we have

$$\begin{aligned}\phi_X(S;G) &\leq \frac{cut_2(S;G_X)}{\min(\frac{1}{2}vol_2(S;G_X),\frac{1}{2}vol_2(\bar{S};G_X))}\\ &= 2\phi_2(S;G_X),\\ \phi_X(S;G) &\geq \frac{\frac{1}{2}cut_2(S;G_X)}{\min(vol_2(S;G_X),vol_2(\bar{S};G_X))}\\ &= \frac{1}{2}\phi_2(S;G_X).\end{aligned}$$

This completes the proof of the inequality (3.10). □

Complexity Analysis. The computational time of MOSC-GL is dominated by the time to form $\mathbf{W}_X$ and compute the second eigenvector of $\mathbf{L}_X$. The former requires finding all triangles in the graph, which can be as large as $O(n^3)$ for a complete graph. While most real graphs are far from complete so the actual complexity is much lower than $O(n^3)$. In general, for a graph with n nodes and m edges, building a triangle adjacency matrix $\mathbf{W}_T$ is at least as hard as the problem of triangle detection (i.e. to check if a graph contains a triangle or not), which in turn is conjectured to require $m^{1+\delta+o(1)}$ time, for some constant $\delta > 0$ [1]. In this thesis, we build $\mathbf{W}_T$ by checking each edge and then finding all possible common neighbours, which requires $O(mn)$ time. For the calculation of the second eigenvector of $\mathbf{L}_X$, it suffices to use power iteration to find an approximate eigenvector, with each iteration at $\tilde{O}(g)$, where g denotes the number of non-zero entries in $\mathbf{L}_X$.

3.2.2 MOSC Based on Random Walks (MOSC-RW)

Alternatively, we can develop MOSC under the random walks framework. Edge/triangle conductance can be viewed as a probability corresponding to the Markov chain. For a set S with edge volume at most half of the total graph edge volume, the edge conductance of S is the probability that a random walk will leave S conditioned upon being inside S, where the transition probabilities of the walk are defined by edge connections [108]. Similarly, for a set S with triangle volume at most half of the total graph triangle volume, the triangle conductance of S is the probability that a random walk will leave S conditioned upon being inside S, where the transition probabilities of the walk are defined by the triangle connections [9]. This motives us to directly combine random walks from edge and triangle connections to perform MOSC. Therefore, we propose MOSC-RW to consider both edge and triangle structures via the respective probability transition matrix and tensor, under the random walks framework.

Specifically, starting with the third-order adjacency tensor $\mathcal{T}$, we define a third-order transition tensor $\mathcal{P}$ as Eq. (2.9). Each entry of $\mathcal{P}$ represents the transition probability of a random walk such that the probability of jumping to a state j depends on the last two states i and k [120]. In the case $\sum_{m=1}^{n} \mathcal{T}(i,m,k) = 0$, we set $\mathcal{P}(i,j,k)$ with 0.

Let $\mathbf{T}_k \in \mathbb{R}^{n\times n}$ denote the kth $n \times n$ block of $\mathcal{P}$, i.e.

$$\mathbf{T}_k = \mathcal{P}(:,:,k). \tag{3.19}$$

Next, we average $\{\mathbf{T}_k, k = 1,...,n\}$ to reduce $\mathcal{P}$ to a similarity matrix $\mathbf{A}$:

$$\mathbf{A} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{T}_k. \tag{3.20}$$

Now recall that $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ denotes the probability transition matrix of random walks on the input graph. We construct a mixed-order similarity matrix $\mathbf{H}$ by a weighted sum of $\mathbf{A}$ and $\mathbf{P}$ via a mixing parameter $\lambda \in [0,1]$ as

$$\mathbf{H} = (1-\lambda)\mathbf{A} + \lambda\mathbf{P}. \tag{3.21}$$

Thus, we obtain the MOSC-RW algorithm with standard SC steps on $\mathbf{H}$, as summarised in Algorithm 3.

When $\lambda = 1$, MOSC-RW is equivalent to conventional SC by Shi and Malik [95] and considers only second-order structures. MOSC-RW with $\lambda = 0$ considers only third-order structures, which is a simplified (unweighted) version of tensor SC (TSC) by Benson *et al.* [9], so we name it as simplified TSC (STSC). In the intermediate case, λ controls the trade-off.

Algorithm 3 MOSC-RW

```
Input: $G = (V, E)$, a mixing parameter $\lambda$
Output: Two node sets $\{S, \bar{S}\}$
 1: Construct the adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$.
 2: Construct the adjacency tensor $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$.
 3: for $1 \leqslant i, j, k \leqslant n$ do
 4:     if $\sum_{m=1}^{n} \mathcal{T}(i,m,k) \neq 0$ then
 5:         $\mathcal{P}(i,j,k) = \mathcal{T}(i,j,k) \,/ \sum_{m=1}^{n} \mathcal{T}(i,m,k)$.
 6:     else
 7:         $\mathcal{P}(i,j,k) = 0$.
 8:     end if
 9: end for
10: $\mathbf{T}_k \leftarrow \mathcal{P}(:,:,k)$ for $k = 1, \cdots, n$.
11: Compute the reduced similarity matrix $\mathbf{A}$.
12: Let $\mathbf{D}$ be diagonal with $\mathbf{D}_{ii} = \sum_{i}^{n} \mathbf{W}(i, j)$.
13: $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$.
14: $\mathbf{H} = (1 - \lambda)\mathbf{A} + \lambda\mathbf{P}$.
15: Compute the second largest eigenvector $\mathbf{v}$ of $\mathbf{H}$.
16: $\mathbf{v} \leftarrow$ Sorting entries of $\mathbf{v}$.
17: $\{S, \bar{S}\} \leftarrow$ Sweep cut on $\mathbf{v}$ w.r.t. some cut criteria.
```

Interpretation. Now we interpret the model as a mixed-order random walk process. At every step, the random walker chooses either a first-order (with probability λ) or a second-order (with probability $(1-\lambda)$) random walk. For the first-order random walk, the walker jumps from the current node i to a neighbour j with probability $\mathbf{P}(i, j) = \frac{1}{\mathbf{D}(i,i)}$. For the second-order random walk in $\mathbf{A}$, $\mathbf{A}(i, j)$ is the probability of the following random process: supposing the walker is at vertex i, it first samples a vertex k with probability $\frac{1}{n}$, then in the case that some neighbour k of i is sampled and i, j, k forms a triangle, the walker jumps from i to j with probability $1/\mathbf{W}_T(i,k)$, where $\mathbf{W}_T(i,k)$ is the number of triangles containing both i and k.

Complexity Analysis. The running time of MOSC-RW is again dominated by the time of finding all the triangles and the approximate eigenvector, and thus asymptotically the same as the running time of MOSC-GL. However, since MOSC-RW involves tensor construction, normalisation and averaging, it is more complex than MOSC-GL in implementation.

The computation of the second largest eigenvector of $\mathbf{H}$ in step 15 is another costly procedure, and its complexity depends on the sparsity of $\mathcal{T}$ and $\mathbf{P}$. Let a and p be the number of non-zeros entries in $\mathcal{T}$ and $\mathbf{P}$, respectively. In theory, the number of non-zero entries of $\mathbf{H}$ can be $O(a+p)$, and an eigenvector can be computed via power iterations, and the running time for each iteration is $O(a+p)$.

Multiple Clusters and Higher-order Cheeger Inequalities of MOSC. To cluster a graph into $k>2$ clusters based on MOSC-GL and MOSC-RW, we follow the conventional SC [108]. Specifically, MOSC-GL treats the first k row-normalised eigenvectors of $\mathbf{L}_X$ as the embedding of nodes that can be clustered by k-means. Similarly, MOSC-RW uses the first k eigenvectors of $\mathbf{H}$ as the node embedding to perform k-means. Regarding performance guarantee, following [10] and [60], MOSC-GL and MOSC-RW do not have performance guarantee with respect to higher-order Cheeger inequalities.

3.2.3 Automatic Determination of λ

The mixing parameter λ is the only hyperparameter in MOSC. To improve the usability, we design schemes to automatically determine its optimal value λ^* from a set Λ based on the quality of output clusters [62, 21, 124]. For bi-partitioning graphs, the cut criterion used to obtain output clusters can help to determine the best λ^* from Λ. For multiple partitioning graphs, we can use the sum of triangle densities of the individual cluster to determine the best λ^* from Λ.

Specifically, for each $\lambda' \in \Lambda$, let $\{S_{\lambda'}, \overline{S_{\lambda'}}\}$ denote the MOSC bi-partitioning clusters obtained with $\lambda = \lambda'$. For a specific minimisation or maximisation cut criterion τ (e.g. edge conductance ϕ_2), we choose λ to be the one that optimises τ, i.e. $\lambda^* = \arg\min_{\lambda' \in \Lambda} \tau(S_{\lambda'})$ or $\lambda^* = \arg\max_{\lambda' \in \Lambda} \tau(S_{\lambda'})$, respectively.

For the case of multiple partitions, we propose a triangle-density-based scheme to determine λ as follows:

$$\lambda^* = \arg\max_{\lambda' \in \Lambda} \sum_{c=1}^{k} \frac{\sum_{v_i, v_j, v_k \in S_c(\lambda')} \mathcal{T}(i,j,k)}{6|S_c(\lambda')|}, \tag{3.22}$$

where $S_c(\lambda')$ denotes the c-th cluster resulted from λ', and the factor 1/6 is used to avoid repeated count of triangles in an undirected graph.

3.2.4 Mixed-order Cut Criteria

A cut criterion measures the quality of output clusters when performing the sweep cut procedure. However, conventional SC is limited to use edge-based cut criteria [95, 79], while triangle-based SC is limited to use triangle-based cut criteria [10, 9]. Thus, to enable existing single-order SC methods [95, 10] to preserve mixed-order structures, we consider cut criteria of different orders from the order used to encode a structure (e.g. second-order SC with a third-order cut criterion). Therefore, we formally define a mixed-order cut criterion as follows:

Definition 4. *(Mixed-order Cut Criterion) Given a graph $G = (V, E)$, a mixed-order cut criterion performs a triangle-based cut criterion $\omega_3 \in \{\phi_3, \eta_3, \xi_3, \alpha_3\}$ on a sorted dominant eigenvector $\mathbf{v}_e$ derived from an edge-based similarity matrix $\mathbf{B}$ of G, or performs an edge-based cut criterion $\omega_2 \in \{\phi_2, \eta_2, \xi_2, \alpha_2\}$ on a sorted dominant eigenvector $\mathbf{v}_t$ derived from a triangle-based similarity matrix* $\mathbf{T}$.

Mixed-order cut criteria are illustrated in the middle block of Fig.3.2. In this thesis, we will empirically study the effectiveness of eight cut criteria including edge- and triangle-based (Table 2.1) ones on representative edge- and triangle-based SC algorithms. Additionally, we discuss a relation between $\xi_3(S)$ and $\eta_3(S)$. According to $\eta_3(S)$ [63] and $cut_3(S, \bar{S})$ [63], we have

$$\eta_3(S) = \frac{2}{3} - \frac{1}{3}\left(\frac{assoc_3(S)}{vol_3(S)} + \frac{assoc_3(\bar{S})}{vol_3(\bar{S})}\right) + \frac{1}{3}\left(\frac{vol_3(\bar{S}) - assoc_3(\bar{S})}{vol_3(S)} + \frac{vol_3(S) - assoc_3(S)}{vol_3(\bar{S})}\right).$$

Then based on $\xi_3(S) = \frac{assoc_3(S)}{vol_3(S)} + \frac{assoc_3(\bar{S})}{vol_3(\bar{S})}$, we can see a relation between $\xi_3(S)$ and $\eta_3(S)$ as below:

$$\eta_3(S) = \frac{2}{3} - \frac{1}{3}\xi_3(S) + \frac{1}{3}(\frac{vol_3(\bar{S}) - assoc_3(\bar{S})}{vol_3(S)} + \frac{vol_3(S) - assoc_3(S)}{vol_3(\bar{S})}). \tag{3.23}$$

In the above Eq. (3.23), the term

$$\omega(S) = \frac{vol_3(\bar{S}) - assoc_3(\bar{S})}{vol_3(S)} + \frac{vol_3(S) - assoc_3(S)}{vol_3(\bar{S})} \tag{3.24}$$

is not a constant function due to the definition of $vol_3(\cdot)$ and $assoc_3(\cdot)$ from Eq. (2.13) and Eq. (2.14) in the thesis respectively.

3.2.5 Mixed-order Evaluation Metrics

If we have ground-truth clusters available, we can use them to measure performance of clustering algorithms. Existing works commonly use mis-clustered nodes [40] or related metrics (e.,g. NMI) [7]. We denote the ground-truth partition of G with k clusters as $\mathbb{S}^* = \{S_1^*, S_2^*, \ldots, S_k^*\}$ and a candidate partition to be evaluated as $\mathbb{S} = \{S_1, S_2, \ldots, S_k\}$. The mis-clustered node metric is defined as

$$\varepsilon_N(\mathbb{S}^*, \mathbb{S}) = \min_{\sigma} \sum_{c=1}^{k} |S_c^* \oplus S_{\sigma(c)}|, \tag{3.25}$$

which measures the difference between two partitions $\mathbb{S}^*$ and $\mathbb{S}$, where σ indicates all possible permutations of $\{1, 2, \ldots, k\}$ and $\oplus$ denotes the symmetric difference between the two corresponding sets. A smaller ε_N indicates a more accurate partition.

To break existing rigid evaluation scenarios based on the level of nodes, we design a flexible way to evaluate the quality of communities by leveraging the level of structures (e.g. triangles). In this scenario, the level of nodes means that popular graph clustering metrics only consider the node information rather than edge/triangle information. For example, the metric mis-clustered node measures the difference between two node sets. Depending on diverse application scenarios, we can flexibly choose one to evaluate communities. For example, if the triangle structure plays an important role in communities (e.g. in social networks), we can evaluate communities in terms of triangles to truly reflect the quality of communities. A limitation of the above metric is that it fails to truly reflect the errors at the level of structures. Also, our studies show that mis-clustered nodes do not have a *monotonic* relationship with mis-clustered edges or triangles. That is, a smaller number of mis-clusterd nodes does not imply smaller number of mis-clustered edges or triangles, and vice versa. This motivates us to propose structure-aware error metrics to measure the quantity of *mis-clustered edges* (ε_E) and *triangles* (ε_T), respectively. Specifically, we define ε_E as

$$\varepsilon_E(\mathbb{S}^*,\mathbb{S}) = \sum_{c=1}^{k} E_N(S_c^*) - \max_{\sigma} \sum_{c=1}^{k} E_N(S_c^* \cap S_{\sigma(c)}), \tag{3.26}$$

where $E_N(S)$ is the number of edges in S. We can define ε_T similarly by replacing $E_N(S)$ with $T_N(S)$, where $T_N(S)$ is the number of triangles in S.

Based on the proposed structure-aware error metrics, we define a *mixed-order evaluation metrics* that can give us new insights on the preservation of edges and triangles for existing single-order SC. It is illustrated in the right block of Fig. 3.2.

Definition 5. *(Mixed-order Evaluation Metrics) Given a graph $G = (V,E)$ and ground-truth $\mathbb{S}^*$, a mixed-order evaluation metric evaluates a candidate partition $\mathbb{S}$ derived from an edge-based similarity matrix $\mathbf{B}$ of G by the triangle-aware error metric ε_T, or evaluates a candidate partition $\mathbb{S}$ derived from a triangle-based similarity matrix $\mathbf{T}$ of G by an edge-aware error metric ε_E.*

3.3 Experiments

This section aims to evaluate MOSC against existing SC methods in two applications: community detection and superpixel segmentation. Furthermore, we will explicitly study the effect of mixed-order cut criteria, and gain insights from the newly designed mixed-order evaluation metrics for the community detection task.

Table 3.1 Statistics of the 2,005 graphs. The number in parentheses is the median for each range.

Graph	$\|V\|$	$\|E\|$	Size	Triangle density	#Interaction edges	#Clusters/graph	#Graph(s)
DBLP	317K	1.05M	14∼303 (22)	7.4∼167.9 (15.4)	1∼278 (15)	2	500
YouTube	1.13M	2.99M	6∼389 (91)	1∼22.9 (3.73)	1 ∼1054 (89)	2	500
Orkut	3.07M	117M	88∼379 (206)	213.7∼1526 (452.6)	37∼10470 (2411)	2	500
LJ	4.00M	34.7M	33∼193 (98)	116.3∼2968 (422.4)	1∼9179 (1489)	2	500
Zachary	34	78	34	1.32	11	2	1
Dolphin	62	159	62	1.53	6	2	1
Polbooks	105	441	105	5.33	70	3	1
Football	115	613	115	7.04	219	12	1
PBlogs	1490	16716	1490	67.8	1576	2	1
Facebook	22,470	170,912	22,470	35.50	19,590	4	1

3.3.1 Datasets

The experiments were conducted on two popular groups of graphs with very different triangle densities: The first group consists of five full real-world graphs: Zachary's karate club (Zachary) [127], Dolphin social network (Dolphin) [69], American college football (Football) [78], U.S. politics books (Polbooks) [78], Political blogs (PBlogs) [4] and Facebook[1] [90]. Full graphs means that the graph size referring the number of nodes and edges is not tailored. Statistics of networks are shown in Table 3.1. The description of these networks is shown as below:

1. Zachary: It is an university karate club in which the nodes represent club members and edges represent the members who interact outside the club. This network is split into two communities since an internal conflict between the administrator and instructors.

2. Dolphin: This is a dolphin social network. The nodes represent dolphins and the edge will appear if a pair of dolphins interact frequently. Two communities are formed because of one temporarily disappeared dolphin.

3. Polbooks: This is a politics book network. The nodes represent authors and a link between two authors exists if they jointly publish at least one paper. Three communities denote categories of books "liberal", "neutral" and "conservative".

[1]https://archive.ics.uci.edu/ml/datasets/Facebook+Large+Page-Page+Network

4. Football: This is a U.S college football network in which the nodes represent teams, and the edges represent regular-season games between the two teams. The network Football has twelve communities. Teams that are in the same conference consist of a community. Teams within the same conference have more frequent games than that of teams in different conferences.

5. PBlogs: This is a political blogosphere about 2004 U.S election network. The nodes represent a blog, and the link is automatically generated from a crawl. The two communities are “liberal” and “conservative”.

6. Facebook: This graph is collected by using the Facebook Graph API and restricted to pages from 4 categories which are defined by Facebook. These categories are: politicians, governmental organisations, television shows and companies.

The second group consists of four complex real-world graphs: DBLP, YouTube, Orkut, and LiveJournal (LJ) from the Stanford Network Analysis Platform (SNAP) [124].[2] All graphs have ground-truth communities available. We describe them as below:

1. DBLP: This is a co-authorship network. Nodes represent authors and a link between two authors exists if they jointly publish at least one paper. The community is formed since authors publish in the same journal or conference.

2. Youtube: This is a video-sharing network. The nodes represent users and the links represent friendship between users. Communities are formed due to created groups that other users join.

3. Orkut: This is an online social network. Nodes represent users and links indicate their friendship. User-defined groups are viewed as communities.

4. LJ: This is a content-sharing online network, such as culture, entertainment. People are the nodes and the links represent the friendship. Communities are created with user-defined groups that can share content.

[2]https://snap.stanford.edu/data/index.html

For the four SNAP graphs, we extract paired communities to focus on bi-partitioning problems with the following procedures: 1) For each graph, we select communities with the top 500 highest triangle densities, among those communities having no more than 200 nodes (for DBLP, YouTube, and Orkut) or 100 nodes (for LJ) because it has high density; 2) For every community in the top list, we choose another community having the most connections with it, among all the other communities in the respective graph (without limiting the community node size). These two communities form a bi-partitioning graph. In this way, we extracted 2,000 graphs from SNAP.

Besides the aforementioned graph datasets, we perform experiments on the test set of Berkeley Segmentation Dataset[3] including two hundred images (size 481×321, equivalent to 154,401 nodes in graphs) with human-labelled ground-truth segmentations.

3.3.2 Baselines

We evaluate MOSC-GL and MOSC-RW against the following seven state-of-the-art methods, including both edge-based SC and triangle-based SC, and both global and local methods:

1. SC-Shi [95]: Shi and Malik developed a method aiming to minimise Ncut_2 criterion via a generalised eigenvalue problem of Eq. (2.7).

2. SC-Ng [79]: Ng *et al.* designed a method built upon [95]. Instead of using one dominant eigenvector, it used the first k eigenvectors of $\mathbf{L}$ for performing k partitions and then an additional row normalisation step before k-means.

3. Tensor Spectral Clustering (TSC) [9]: TSC is a higher-order spectral clustering method developed by Benson *et al.* They constructed a transition tensor $\mathcal{P}$ as in Eq. (2.9) and used an expensive multilinear PageRank algorithm [42] to produce a vector as the weight for reducing the tensor to a matrix via weighted average, followed by conventional SC.

[3]https://www2.eecs.Berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

4. Higher-order SVD (HOSVD) [39]: To address the hypergraph clustering problem, this method used an adjacency tensor $\mathcal{T}$ to encode hyperedges, which is equivalent to the adjacency tensor definition in Eq. (2.8). $\mathcal{T}$ is then reduced to a matrix via computing a modelwise covariance matrix, followed by conventional SC.

5. Motif-based SC (MSC) [10] / Tensor Trace Maximisation (TTM) [40]: MSC is a general higher-order spectral clustering method via re-weighting edges according to the number of motifs containing corresponding edges, followed by conventional SC. TTM is independently proposed but equivalent to MSC, which we have verified both analytically and experimentally.

6. HOSPLOC [136]: This is a higher-order local clustering method aiming for more efficient processing while taking higher-order graph structures into account.

7. DeepWalk [82]: DeepWalk adopts an unsupervised Skip-Gram [6] neural network model to learn the embedding of each node. This approach samples random walks from each node, and then maximises the co-occurrence probability among the nodes that appear as neighbours. Following [17], we employ the learned embedding of nodes in a *k*-means to conduct communities.

For superpixel segmentation, besides the above baselines, we compare MOSC with two baselines that are specifically designed to the superpixel segmentation task:

1. Simple Linear Iterative Clustering (SLIC) [3]: It adapts a variant of *k*-means clustering approach to efficiently generate superpixels by a weighted distance measure combining colour and spatial proximity.

2. Linear Spectral Clustering (LSC)[23]. LSC relates the optimisation objectives between *k*-means and normalised cuts by introducing a elaborately designed high dimensional space.

We study two versions for each MOSC: MOSC ($\lambda = 0.5$): MOSC with a fixed (recommended) λ value of 0.5; MOSC (Auto-λ): MOSC with automatically determined λ; Additionally, for MOSC-RW, we study simplified TSC (STSC) when $\lambda = 0$.

We implemented algorithms using Matlab code released by the authors of MSC,[4] HOSVD,[5] HOSPLOC,[6], SLIC[7], LSC[8], and TSC via multilinear PageRank.[9] We followed guidance from the original papers to set their hyperparameters. All experiments were performed on a Linux machine with one 2.4GHz Intel Core and 16G memory. We have released the Matlab code for MOSC.[10]

3.3.3 Evaluation metrics

For the community detection task, we use the proposed structure-aware metrics, mis-clustered edges (ε_E) and triangles (ε_T). We also use two popular metrics, mis-clustered nodes (Eq. (3.25)) and normalised mutual information (NMI) [7, 22]. For the SNAP graphs, we show the average results of the 500 bi-partitioning graphs.

To define NMI, we need the Shannon entropy for $\mathbb{S}$ that can be defined as

$$H(\mathbb{S}) = -\sum_{c=1}^{k} (n_{S_c}/n) \log(n_{S_c}/n), \tag{3.27}$$

where n_{S_c} is the number of vertices in community S_c. The mutual information between $\mathbb{S}$ and $\mathbb{S}^*$ can be expressed as

$$I(\mathbb{S},\mathbb{S}^*) = \sum_{c=1}^{k}\sum_{d=1}^{k} \frac{n_{S_c S_d^*}}{n} \log\left(\frac{n_{S_c S_d^*}/n}{(n_{S_c}/n)\times(n_{S_d^*}/n)}\right), \tag{3.28}$$

where $n_{S_c S_d^*}$ is the number of vertices shared by communities S_c and S_d^*. The NMI between two partitions $\mathbb{S}$ and $\mathbb{S}^*$ is defined as

$$\mathrm{NMI}(\mathbb{S},\mathbb{S}^*) = \frac{2I(\mathbb{S},\mathbb{S}^*)}{H(\mathbb{S})+H(\mathbb{S}^*)}. \tag{3.29}$$

[4]https://github.com/arbenson/higher-order-organization-matlab
[5]http://sml.csa.iisc.ernet.in/SML/code/Feb16TensorTraceMax.zip
[6]http://www.public.asu.edu/~dzhou23/Code/HOSPLOC.zip
[7]https://www.mathworks.com/help/images/ref/superpixels.html
[8]https://github.com/neuwangmeng/
[9]https://github.com/dgleich/mlpagerank
[10]https://bitbucket.org/Yan_Sheffield/mosc/

If $\mathbb{S}$ and $\mathbb{S}^*$ are identical, NMI$(\mathbb{S},\mathbb{S}^*)$ = 1. If $\mathbb{S}$ and $\mathbb{S}^*$ are independent, NMI$(\mathbb{S},\mathbb{S}^*)$ = 0.

For the superpixel segmentation task, we quantitatively evaluate our algorithm by two popular metrics that are 1) undersegmentation error (UE) [93], 2) achievable segmentation accuracy (ASA) [67]. To compute the above metrics, we use $\mathbb{C} = \{C_j\}_{j=1}^{K}$ to denote the K segmentations of a ground-truth image, and $\mathbb{T} = \{T_i\}_{i=1}^{L}$ to represent the L segmentations by the superpixel algorithm. where K and L are the number of superpixel in a ground-truth and predicted output image.

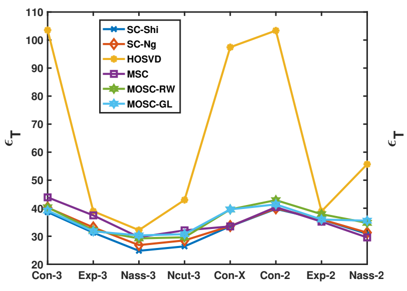
1. Undersegmentation Error [93]. A good superpixel segmentation should ensure that a superpixel only overlaps with one object. This evaluation measurement considers the deducted area by the superpixel that overlaps with the given ground-truth. We formulate it as
$$\mathrm{UE} = \frac{\sum_i \sum_{k:C_k \cap T_i \neq \emptyset} |C_k - T_i|}{\sum_i |T_i|}. \tag{3.30}$$

2. Achievable Segmentation Accuracy [67]. It calculates the highest achievable accuracy by labelling each superpixel with the label of ground truth segmentation that has the biggest overlap area. The metric is defined as
$$\mathrm{ASA} = \frac{\sum_k \max_i |C_k \cap T_i|}{\sum_i |T_i|}. \tag{3.31}$$

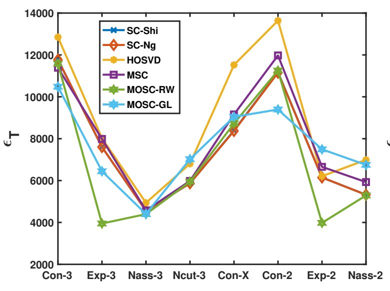
3.3.4 Effectiveness of Mixed-order Cut Criteria

Firstly, we study the effect of mixed-order cut criteria on clustering performance. We have seven existing cut criteria from Table 2.1 and the proposed mixed-order conductance (ϕ_X). We study their effect on SC-Shi, SC-Ng, MSC, HOSVD, MOSC-GL ($\lambda = 0.5$) and MOSC-RW ($\lambda = 0.5$) on DBLP, Orkut and LJ w.r.t ε_T, ε_E, ε_N. Note that we omit experimental results about ε_E on DBLP, Orkut and LJ since they show the same scenarios with ε_T on these three datasets. From Fig. 3.3, we have the following observations:

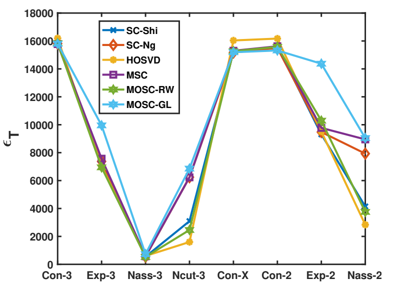
1. Mixed-order cut criteria have a greater impact on graphs with dense triangles (e.g. Orkut and LJ) than graphs with the sparse triangles (e.g. DBLP). The reason is that one



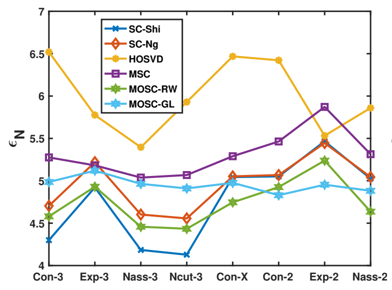
(a) ε_T on DBLP



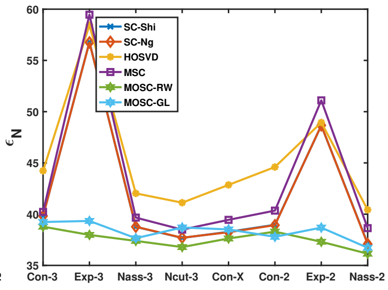
(b) ε_T on Orkut



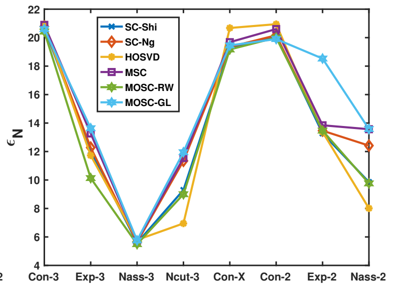
(c) ε_T on LJ



(d) ε_N on DBLP



(e) ε_N on Orkut



(f) ε_N on LJ

Fig. 3.3 Mixed-order cut criteria analysis on SC-Shi, SC-Ng, HOSVD, MSC, MOSC-GL ($\lambda = 0.5$) and MOSC-RW ($\lambda = 0.5$) w.r.t mis-clustered triangles (ε_T) and nodes (ε_N) for eight cut criteria: Conduct$_3$ (Con-3), Exp$_3$, Nassoc$_3$ (Nass-3), Ncut$_3$, Conduct-X (Con-X), Conduct$_2$ (Con-2), Exp$_2$, Nassoc$_2$ (Nass-2). In general the cut criteria Nassoc$_3$ is the best criteria. The correlations between criteria and errors are that triangle-based criteria can lead to a low number of errors and depend on the properties of graphs.

error node in a dense triangle graph, in comparison to in a sparse graph, is normally shared by more error triangles.

2. Some optimised cut criteria do not truly reflect the quality of output communities w.r.t ε_T when comparing with ground-truth communities. In particular, in DBLP the optimised Nassoc$_2$ of SC-Shi still has great quantity of error triangles ε_T when comparing with some other optimised cut criteria (e.g. Ncut$_3$).

3. Mixed-order cut criteria can improve the performance of SC-Shi and SC- Ng that are conventional SC. In general, the cut criteria Nassoc$_3$ is the best criteria. It consistently gives the lowest number of error triangle ε_T and edge ε_E over three datasets than other seven cut criteria except for ε_E of MOSC-RW on Orkut. For error nodes, Nassoc$_3$ has the lowest number on LJ and second lowest on DBLP and Orkut.

4. Triangle-based criteria can lead to the less errors than edge-based criteria. In particular, Nassoc_3 can achieve a very low number of errors. In general edge-based cut criteria (e.g. Nassoc_2) do not show such low number of errors. From Table 2.1, Nassoc_3 is the only one that considers the number of triangles within communities. Therefore, the maximisation of the number of triangles within communities is effective to reduce errors in output communities.

5. The correlation between criteria and errors depends on the properties of datasets. For LJ with small graphs and dense triangles, the criterion Nassoc_3 can significantly reduce errors in terms of ε_N, ε_E and ε_T when comparing with other criteria. By contrast, for DBLP with big graphs and sparse triangles, the performance of criterion Nassoc_3 is weakened so that the performance of criteria Ncut_3 is closed to it in terms of ε_E and ε_T.

Based on the above observations, we recommend the criterion Nassoc_3 to achieve a low number of error edges and triangles. We recommend Ncut_3 to reduce the number of error nodes for big and sparse graphs.

3.3.5 Performance for Community Detection

We study the results of all algorithms in combination of all eight criteria and k-means (KM). Fig. 3.3 shows that cut criteria can affect the performance of all algorithms. Therefore, for fair comparison, we report the clustering results conducted by the best criteria for each algorithm. The top two results are in bold (best) or underlined (second best).

Results on SNAP Graphs. We show the performance of all clustering algorithms with the best cut criteria in terms of NMI, ε_N, ε_E, and ε_T on SNAP graphs in Table 3.2. The results for some settings of TSC and HOSPLOC are not available either due to long running time (not finished within 40 hours) or out of memory. In particular, the multilinear PageRank algorithm in TSC is very expensive.

We have four observations:

1. MOSC-RW ($\lambda = 0.5$) achieves the best in 10 out of 16 settings.

Table 3.2 Performance of clustering algorithms with the best cut criteria on SNAP graphs. The best is in **bold** and the second best is underlined. A larger NMI indicates a better result, while a smaller $\varepsilon_N/\varepsilon_E/\varepsilon_T$ indicates a better result.

		Second order			Third order					MOSC-RW		MOSC-GL	
Method		SC-Shi	SC-Ng	DeepWalk	HOSVD	MSC	TSC	HOSPLOC	STSC	$\lambda = 0.5$	Auto-λ	$\lambda = 0.5$	Auto-λ
DBLP	NMI	0.650	**0.656**	0.614	0.550	0.620	0.648	0.286	0.628	0.654	0.646	0.648	0.645
	ε_N	**4.13**	4.56	4.99	5.40	4.65	4.30	17.28	4.24	4.43	4.76	4.82	4.87
	ε_E	**13.36**	14.58	17.46	17.99	15.98	18.72	70.49	18.72	14.27	15.68	15.87	16.67
	ε_T	**24.81**	26.84	35.15	32.18	29.57	37.93	236.65	45.46	29.23	28.46	30.30	32.01
YouTube	NMI	0.248	0.270	0.258	0.124	0.184	-	-	0.150	**0.284**	0.260	0.275	0.263
	ε_N	22.48	23.31	26.63	25.69	24.41	-	-	23.83	**22.18**	23.46	23.44	23.83
	ε_E	44.42	46.46	61.43	50.61	47.18	-	-	63.12	**44.28**	46.74	52.58	47.96
	ε_T	**27.70**	29.49	47.36	30.78	29.1	-	-	59.78	28.90	29.29	38.63	29.46
Orkut	NMI	0.397	0.397	0.399	0.3618	0.390	-	-	0.387	**0.410**	0.393	0.397	0.394
	ε_N	37.13	37.09	38.06	40.43	38.49	-	-	37.60	**36.05**	37.02	36.72	36.93
	ε_E	574.6	574.6	635.5	624.7	582.3	-	-	569.2	**521.6**	571.8	550.4	574.9
	ε_T	4557	4557	5703	4937	4575	-	-	5104	**3949**	4541	4405	4614
LJ	NMI	0.226	0.224	0.156	0.218	0.224	0.214	-	0.201	**0.229**	0.221	0.208	0.212
	ε_N	5.58	5.63	23.08	5.79	5.74	5.52	-	**5.15**	5.49	5.66	5.76	5.64
	ε_E	49.83	50.01	1134	55.09	52.54	58.19	-	57.06	**47.88**	51.01	58.64	54.17
	ε_T	546.1	547.6	3404	600.6	574.5	737.7	-	773.0	**530.6**	556.4	730.3	617.8

2. MOSC-RW outperforms MOSC-GL, although MOSC-GL achieves top two results in 4 settings. We will give a detailed discussion about it at Sec. 3.3.7.

3. Both MOSC-RW($\lambda = 0.5$) and MOSC-GL($\lambda = 0.5$) have better results than MOSC-RW(Auto-λ) and MOSC-GL(Auto-λ). This demonstrates that a fixed mixing parameter is effective, but it also shows the automatic schemes are not effective in these settings.

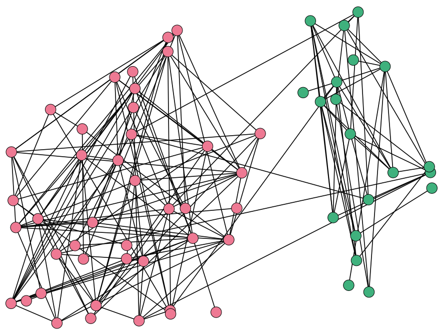
4. Mixed-order evaluation metrics can give insights on the quality of structure preservation for single-order SC. Specifically, SC-Shi preserves the most number of nodes, edges and triangles in DBLP than others. Additionally, although SC-Shi does not preserve the most number of nodes in YouTube, it still can preserve the most number of triangles than others.

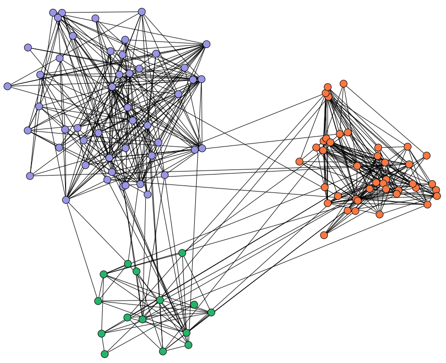
Table 3.3 Clustering performance of algorithms with the best cut criteria. The best is in **bold** and the second best is $\underline{\text{underlined}}$. A larger NMI indicates a better result, while a smaller $\varepsilon_N/\varepsilon_E/\varepsilon_T$ indicates a better result. Note that there are ties.

		Second order			Third order				MOSC-RW		MOSC-GL	
Method		SC-Shi	SC-Ng	DeepWalk	HOSVD	MSC	TSC	STSC	$\lambda = 0.5$	Auto-λ	$\lambda = 0.5$	Auto-λ
Zachary	NMI	**0.837**	**0.837**	0.732	0.069	0.732	0.677	0.325	**0.837**	**0.837**	**0.837**	**0.837**
	ε_N	**1**	**1**	2	14	2	2	8	**1**	**1**	**1**	**1**
	ε_E	**2**	**2**	7	34	3	3	24	**2**	**2**	**2**	**2**
	ε_T	**1**	1	10	16	**1**	**1**	14	**1**	**1**	**1**	**1**
Football	NMI	0.883	0.904	0.529	0.896	0.924	0.866	0.862	0.924	$\underline{0.924}$	0.9	**0.931**
	ε_N	23	15	65	16	$\underline{10}$	26	26	$\underline{10}$	$\underline{10}$	15	**9**
	ε_E	63	37	291	36	**7**	70	72	**7**	**7**	36	**7**
	ε_T	99	50	584	39	**2**	110	114	**2**	**2**	39	**2**
Polbooks	NMI	0.575	0.542	**0.615**	0.092	0.542	0.180	0.103	0.575	0.575	0.563	$\underline{0.589}$
	ε_N	**17**	18	**17**	56	18	55	51	**17**	**17**	**17**	**17**
	ε_E	$\underline{27}$	33	39	185	34	281	172	$\underline{27}$	$\underline{27}$	28	**21**
	ε_T	$\underline{7}$	10	19	234	8	384	227	$\underline{7}$	$\underline{7}$	$\underline{7}$	**1**
Dolphin	NMI	$\underline{0.889}$	$\underline{0.889}$	$\underline{0.889}$	0.081	0.536	0.582	0.631	$\underline{0.889}$	$\underline{0.889}$	$\underline{0.889}$	**1**
	ε_N	$\underline{1}$	$\underline{1}$	$\underline{1}$	19	7	6	5	$\underline{1}$	$\underline{1}$	$\underline{1}$	**0**
	ε_E	$\underline{1}$	$\underline{1}$	$\underline{1}$	43	10	8	6	$\underline{1}$	$\underline{1}$	$\underline{1}$	**0**
	ε_T	**0**	**0**	**0**	29	**0**	1	**0**	**0**	**0**	**0**	**0**
PBlogs	NMI	0.007	0.007	**0.740**	0.014	0.023	-	0.430	0.012	$\underline{0.458}$	0.098	0.016
	ε_N	671	732	**54**	677	614	-	$\underline{204}$	659	230	478	647
	ε_E	7,302	7,302	**159**	7,307	7,260	-	362	7,302	$\underline{184}$	7,302	7,301
	ε_T	36,401	36,402	**423**	36,400	36,400	-	631	36,401	$\underline{456}$	36,402	36,400
Facebook	NMI	0.023	$\underline{0.255}$	0.163	0.177	0.055	-	0.0228	0.023	0.139	0.032	**0.258**
	ε_N	15,553	$\underline{11,171}$	13,062	12,234	14,206	-	15,636	15,553	12,955	15,517	**11,166**
	ε_E	69,789	**57,630**	64,195	60,853	69,889	-	81,438	69,789	67,973	69,740	$\underline{57,635}$
	ε_T	244,499	190,424	220,356	**182,274**	243,998	-	381,685	244,499	244,498	244,487	$\underline{190,335}$

Results on Full Graphs. We show the performance of all clustering algorithms with the best cut criteria for five full graphs in terms of NMI, ε_N, ε_E, and ε_T in Table 3.3, except HOSPLOC, for which we were not able to obtain comparable results. We have four observations:

1. MOSC-GL (Auto-λ) achieves the best performance in 17 out of 24 settings, demonstrating that automatic determination of λ is effective in these settings. Specifically, in Dolphin MOSC-GL (Auto-λ) produces perfect results in all metrics. For graphs with multiple clusters (Polbooks, Football), MOSC-GL (Auto-λ) is also superior to others. We visualise the output clusters of Dolphin, Polbooks and Football in Fig. 3.4a, 3.4b and 3.4c respectively.


(a) Dolphin


(b) Polbooks

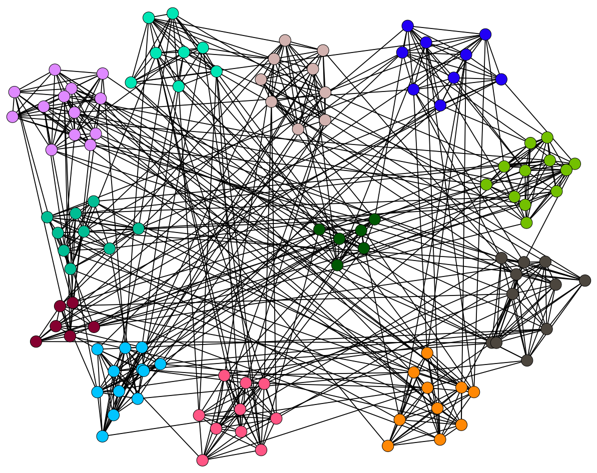

(c) Football (12 clusters)

Fig. 3.4 Clusters in Polbooks, Dolphin and Football graphs discovered by MOSC-GL (Auto-λ).

2. MOSC-GL (Auto-λ) outperforms MOSC-RW (Auto-λ), although MOSC-RW (Auto-λ) achieves the best results in 10 settings, which is still better than all existing SC algorithms (Note that there are ties).

3. Mixed-order evaluation metrics can gain insights on the quality of structure preservation. In Football MSC achieves the best ε_E and ε_T but not for ε_N, which also indicates existing mis-clustered node cannot reflect errors in structures.

4. In Facebook, MOSC-GL (Auto-λ) achieves the best performance w.r.t NMI and ε_N, and achieves the second best w.r.t ε_E and ε_T. Also, the results of SC-Ng are closed to

Table 3.4 The best is in **bold** and the second best is <u>underlined</u>. The down arrow means that a small number indicates good performance. The up arrow means that a large number indicates good performance. MOSC-GL uses λ = 0.5. Comparing with LSC, MOSC-GL achieves competitive results especially for ASA.

	SLIC	SC-Shi	SC-Ng	MSC	DeepWalk	LSC	MOSC-GL
UE $\downarrow$	0.123	0.256	0.112	0.110	0.697	**0.092**$\pm$0.046	<u>0.108</u>$\pm$0.046
ASA $\uparrow$	0.867	0.767	0.880	0.882	0.438	**0.894**$\pm$0.045	<u>0.883</u>$\pm$0.043

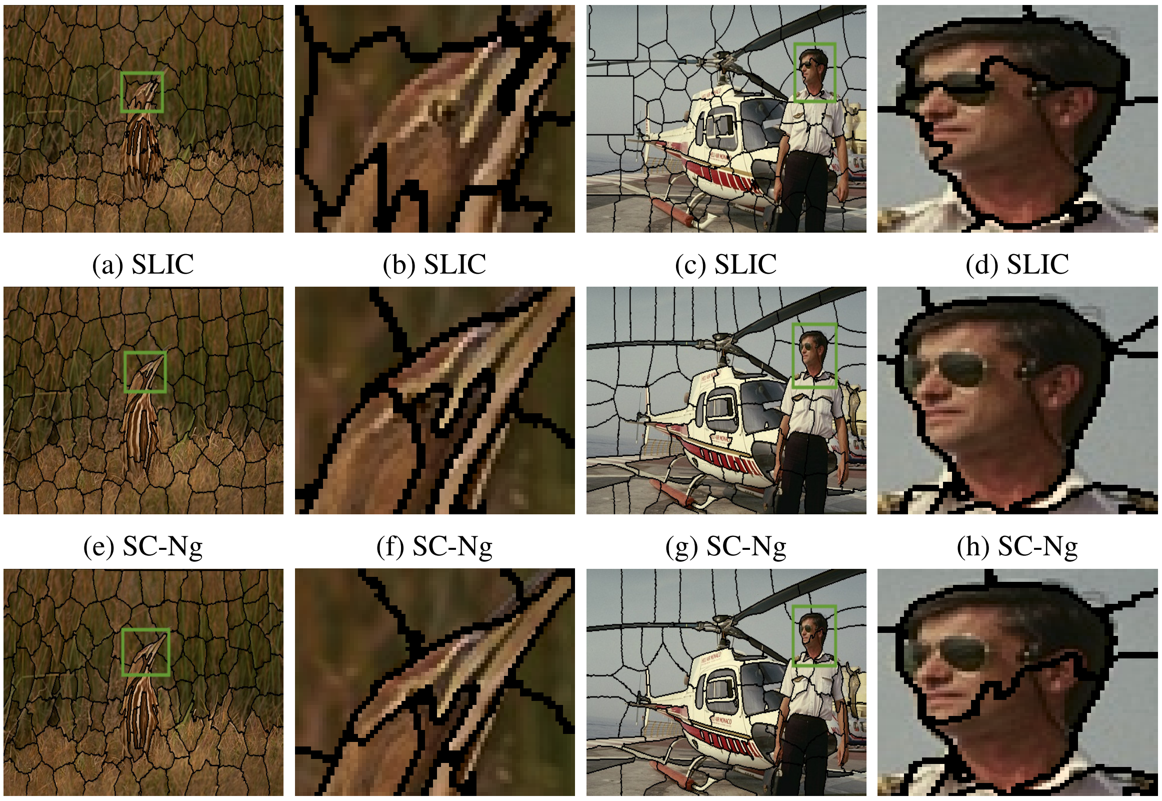
the results of MOSC-GL (Auto-λ) since MOSC-GL(Auto-λ) is the generalisation of SC-Ng.

In Table 3.2, our auto-learning strategy that is based on cut criteria do not show superior performance for bi-partitioning and dense graphs extracted from large graphs. The reason is that optimised cut criteria do not truly reflect the quality of output communities w.r.t NMI, ε_N, ε_E, ε_T when comparing with ground-truth communities. However, in Table 3.3, our auto-learning strategy that is based on triangle density is effective to multi-cluster graphs. Triangle density of communities can better reflect the quality of communities.

3.3.6 Performance for Superpixel Segmentation

A superpixel is a group of similar pixels in colour or other low-level properties [99]. Superpixel segmentation as a preprocessing technique is increasingly popular in many computer vision tasks such as object tracking [117]. The main merit of superpixel is to provide a more natural and perceptually meaningful representation of the input image [93]. Therefore, compared with the traditional pixel representation of images, the superpixel representation greatly reduces the number of image primitives and improves the representative efficiency [93].

We exclude the comparison with tensor-based method, e.g. MOSC-RW, HOSVD and TSC due to their large space consumption (still out of memory using 100G memory to



(a) SLIC (b) SLIC (c) SLIC (d) SLIC

(e) SC-Ng (f) SC-Ng (g) SC-Ng (h) SC-Ng

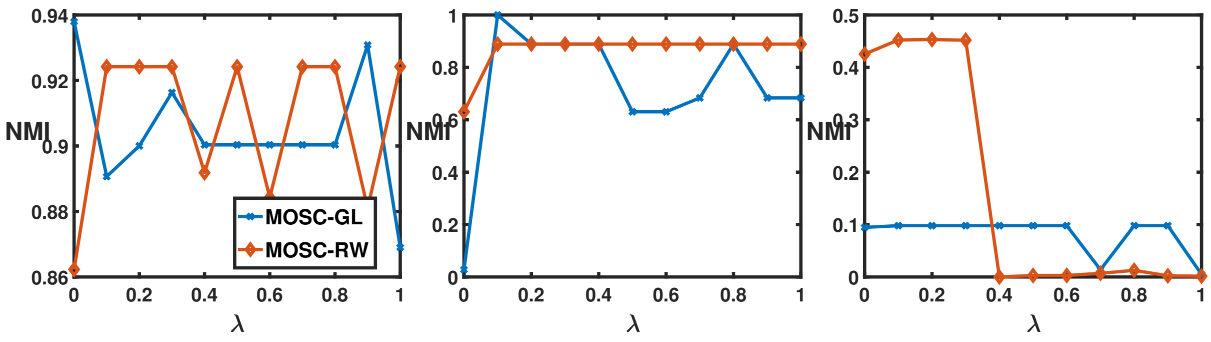
(i) MOSC (λ=0.5) (j) MOSC (λ=0.5) (k) MOSC (λ=0.5) (l) MOSC (λ=0.5)

Fig. 3.5 Visual comparison among SLIC, SC-Ng and MOSC (λ=0.5). We observe that superpixels obtained by MOSC ($\lambda = 0.5$) can adhere well to boundaries of the head of a bird, and also can fit to the edges between a man's neck and head. SLIC generates the irregular and sharp superpixels.

construct the adjacency tensor). For all compared algorithms, the total number of superpixel is set to 100 for each image.

For quantitative evaluation, from Table 3.4, we observe that comparing with LSC, MOSC-GL achieves competitive results especially for ASA. Although our MOSC-GL is not motivated to handle superpixel segmentation, superpixel segmentation still turns out to be an application that is worthy to be applied. Furthermore, LSC is not applicable to the community detection task in graphs since it cannot form graphs.

For qualitative evaluation, Fig. 3.5 shows two example segmentations generated by MOSC (λ = 0.5), SC-Ng and SLIC. At the first column in Fig. 3.5, bounding boxes aim to highlight important parts of images such as a bird/human head in this case. At the



(a) Football (b) Dolphin (c) PBlogs

Fig. 3.6 Sensitivity analysis of λ on Football, Dolphin and PBlogs w.r.t NMI.

second column, you can see enlarged parts within bounding boxes to visually compare image segmentation performance. We observe that the superpixel boundaries obtained by proposed MOSC ($\lambda = 0.5$) can fit the object edges better than others. For example, MOSC ($\lambda = 0.5$) can adhere well to boundaries of the head of a bird, and also can fit to the edges between a man's neck and head. Additionally, compared with graph-based methods, superpixels obtained by SLIC are irregular and sharp.

3.3.7 Sensitivity Analysis

The mixing parameter λ is the only hyperparamter in MOSC. To gain insight of MOSC, we conduct sensitivity analysis on λ as shown in Fig. 3.6 w.r.t. NMI. We can see that the choice of λ can significantly affect the performance while there are large regions of stable performance as well. This was the motivation of developing schemes to automatically determine the best λ. For PBlogs, Table 3.3 shows that MOSC-RW achieves significantly better performance than the others. From Fig. 3.6c, MOSC-RW does not have good performance for large λ values (>0.4). Fortunately, benefiting from automatic λ determination scheme, an outstanding performance has been achieved.

From performance comparison in Section 3.3.5 and the above sensitivity analysis, we see that MOSC-GL and MOSC-RW have different performance on graphs with different triangle densities. MOSC-RW tends to be better for graphs with high triangle densities while MOSC-GL tends to be better for graphs with low triangle densities. For MOSC-GL, $\mathbf{W}_T$ can dominate $\mathbf{W}_X$ in $\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}$, especially for dense graphs. Each entry

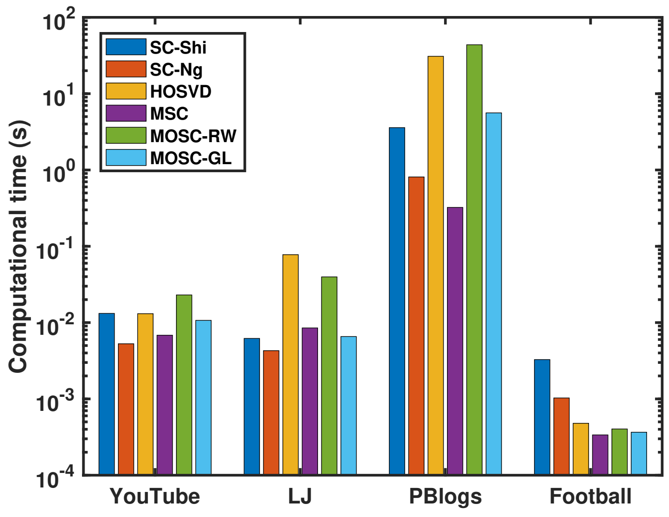


Fig. 3.7 Computational time (in log scale) on YouTube, LJ, PBlogs and Football.

of $\mathbf{W}_T$ denotes the number of triangles containing the corresponding edge while $\mathbf{W}$ is a binary matrix. Therefore, for most non-zero pairs (i, j), $\mathbf{W}_T(i, j)$ is much larger than $\mathbf{W}(i, j)$ especially for dense graphs. This can be the reason that MOSC-GL is less sensitive to tuning λ, or finding the appropriate λ is more difficult. That is, $\mathbf{W}_X$ tends to encode much less edge information. In contrast, MOSC-RW does not have such issue since $\mathbf{A}$ and $\mathbf{P}$ are normalised and thus they are in similar scales before linear combination. Therefore, MOSC-RW has a better performance than MOSC-GL in SNAP graphs and the dense full graph PBlogs. Furthermore, based on the above discussion, we can give an explanation that MOSC-RW is quite sensitive to λ on PBlogs. In PBlogs, triangle information is likely to dominate the mixed-order structure and thus tuning λ may not change its mixture proportion largely especially for the PBlogs with dense triangles.

3.3.8 Computational Time

Fig. 3.7 compares the computational time of different methods on YouTube, LJ, PBlogs and Football, using k-means to obtain the final clusters to avoid the effect of cut criteria. We have the following three observations: 1) Both HOSVD and MOSC-RW involve tensor construction and operations so they are both more time consuming, in particular on dense graphs such as LJ and PBlogs, where HOSVD is the slowest and MOSC-RW is the second slowest. The reason is that HOSVD uses a more complicated dimension reduction method than MOSC-RW. 2) MOSC-GL is more efficient than MOSC-RW in all cases and has similar efficiency as conventional SC methods on the whole. 3) SC-Shi and SC-Ng are slower than MOSC-RW and MOSC-GL on Football since they use more time on converging of k-means step. But for PBlogs that is dense and large, MOSC-RW spends lots of time on constructing the triangle tensor while MOSC-GL is scalable to construct the triangle matrix.

3.4 Summary

This chapter proposed two mixed-order spectral clustering (MOSC) methods, MOSC-GL and MOSC-RW, which model both second-order and third-order structures simultaneously. MOSC-GL combines edge and triangle adjacency matrices with theoretical performance guarantee. MOSC-RW combines first-order and second-order random walks with a probabilistic interpretation. Moreover, we designed mixed-order cut criteria to enable existing single-order SC to preserve mixed-order structures, and new mixed-order evaluation metrics for structure evaluation. Experiments on community detection and superpixel segmentation tasks show that MOSC algorithms outperform existing SC methods in most cases and the proposed mixed-order approach has produced superior clustering of graphs and superpixel segmentations of images.

Chapter 4

Unifying Homophily and Heterophily Graph Transformation via Motifs

4.1 Introduction

As pointed out in Sec.2.4.2, preserving HOP instead of only considering one-hop neighbourhood relationship (e.g., adjacency matrix) has been shown to be effective for the graph representation. Based on the proximity assumption, there are three categories of HOP: homophily, heterophily, and hybrid. In homophily [36], nodes that are highly interconnected and in the same community should be placed tightly in embedding space. For example, in Fig. 4.1a, proximity within the same department should be higher than different ones after embedding, which benefits community detection [110] and node classification [82]. For heterophily [57], nodes that are far away and in different groups, but due to their strong structural similarity, they should be close after embedding. For example, in Fig. 4.1b, department heads from different academic areas (yellow nodes) should have stronger relation than their immediate neighbours due to the same job role under heterophily, good for structural role classification [88].

The higher-order proximities in the aforementioned methods are defined to be either homophily or heterophily. Such "one-size-fit-all" proximity representation potentially limits the performance and interpretation on many graph-based tasks. To alleviate this problem,

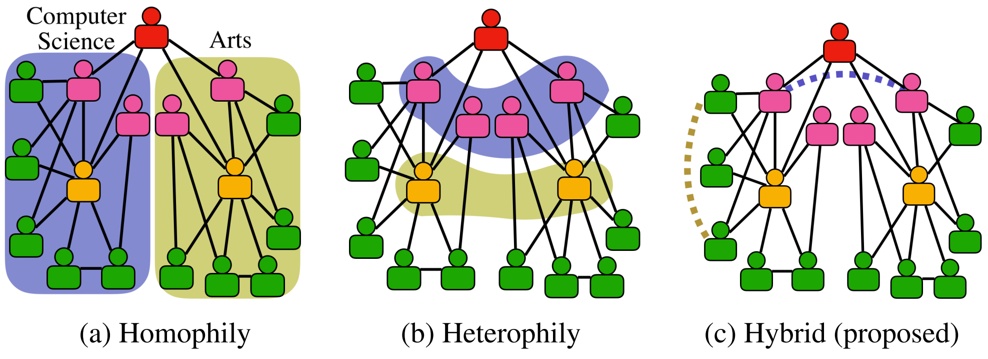


Fig. 4.1 Three categories of higher-order proximity assumptions for graph embedding: homophily, heterophily and hybrid (proposed). Homophily assumes proximity within the same department should be higher than different ones after embedding; department heads (yellow nodes) from different academic areas should have stronger relation than their immediate neighbours.

one representative *hybrid* solution is the node2vec method [46], which flexibly adopts both breadth-first and depth-first search strategies to conduct a biased random-walk process. However, it is designed specifically for random-walk-based methods and cannot take advantage of other more powerful graph embedding methods proposed recently, such as GCN and AROPE. Additionally, most existing proximity preserving methods still rely on simple pairwise relations without considering motifs (e.g., triangles, 4-vertex cliques) that can directly capture interactions between more than two nodes [10]. For example, triangular structures, with three reciprocated edges connecting three nodes, play important roles in social networks [59] that can be partitioned into dense triangle communities with motif spectral clustering (MotifSC) [10]. This thesis will focus on fundamental triangle motif structure, though our proposed method can be easily extended to other motifs.

To design a general framework with a hybrid HOP assumption, we propose a **h**omophily and **h**eterophliy preserving **g**raph **t**ransformation (H^{2}GT) with motif representations. Our H^{2}GT defines a new HOP by micro-level and macro-level walk paths as two complementary components to represent homophily and heterophily. The *micro-level* walk paths embody the homophily assumption, aiming to collect the similarity of close neighbours according to their homophily levels generated by motif information. The *macro-level* walk paths embody

Table 4.1 Comparison with state-of-the-art methods in term of fulfilled (✓) and missing (✗) properties.

Property	DeepWalk	LINE	node2vec	AROPE	GCN	struc2vec	MotifSC	**H^{2}GT**
Homophily	✓	✓	✓	✓	✓	✗	✓	✓
Heterophily	✗	✗	✓	✗	✗	✓	✗	✓
Motif	✗	✗	✗	✗	✗	✗	✓	✓

the heterophily assumption, aiming to encourage walk paths to explore global information according to structural similarity.

As a general framework, H^{2}GT is not limited to one specific algorithm but can be integrated to any graph embedding algorithm as a preprocessing step, without requiring changing their cores. Furthermore, the two walk path strategies rely on local motif structures and sparsifying graphs, which subsequently improves the computational efficiency when we integrate H^{2}GT with existing graph embedding algorithms. Table 4.1 compares three desirable properties to show the uniqueness of H^{2}GT compared with several state-of-the-art (SOTA) methods. To summarise, the contributions of this chapter are as follows:

1. We propose micro-level and macro-level walk paths to preserve homophily and heterophily in HOP by theoretically studying why most HOP preserving embedding methods only hold a homophily assumption.

2. We propose a simple and novel framework to unify homophily and heterophily representations according to micro-level and macro-level walk paths, and three instantiations.

3. We conduct experiments on three tasks, node classification, structural role classification, and motif prediction (a generalised link prediction problem) to show the superior performance of H^{2}GT over SOTA methods.

4.2 Methodology

4.2.1 Theoretical Framework and Motivations

In this section, we theoretically reveal why most graph embedding algorithms only hold a homophily assumption in HOP. This motivates us to propose micro-level and macro-level walk paths strategies to represent homophily and heterophily in HOP respectively.

We first introduce an existing fully-connected planted partition model (PPM) [28] as follows:

Definition 6. ***(Fully-Connected PPM)*** *Let $G_f \sim G_f(nk,k,p,q)$ be a graph sampled from the planted partition model on nk vertices, with k clusters $C = \{C_1, \cdots, C_i, \cdots, C_k\}$ each with exactly n vertices. The edge set is then generated as follows: two vertices $\{v_i, v_j\} \in C_i$ are connected with weight p otherwise with weight $q < p$ to ensure well-connected clusters.*

We prove the following lemma to show a relationship between homophily and HOP.

Lemma 4. *Let $G_f \sim G_f(nk,k,p,q)$ be a fully-connected PPM with k clusters $C = \{C_1, \cdots, C_i, \cdots, C_k\}$, nodes $\{ v_i, v_j\} \in C_s$ and $v_k \in C_t$, the value of l^{th}-order proximity between v_i and v_j is $a_{ij}^{(l)}$, then*

$$a_{ik}^{(l)} < a_{ij}^{(l)}. \tag{4.1}$$

Proof. Based on the following Chapman-Kolmogorov equations [103], we have

$$a_{ik}^{(l)} = \sum_{z=1}^{kn} a_{iz}^{(l-1)} \cdot a_{zk}, \;\; a_{ij}^{(l)} = \sum_{z=1}^{kn} a_{iz}^{(l-1)} \cdot a_{zj}. \tag{4.2}$$

The above equations interpret the proximity between node v_i and node v_j in l steps is obtained by summing the proximity of the mutually events of going from node i to some node v_k in the first $n-1$ walk steps and then going from node k to node j in the l^{th} walk step. Therefore, the difference is

$$\begin{aligned} a_{ij}^{(l)} - a_{ik}^{(l)} &= \sum_{z \in C_s} a_{iz}^{(l-1)} \cdot (a_{iz} - a_{zk}) + \sum_{z \in C_t} a_{iz}^{(l-1)} \cdot (a_{iz} - a_{zk}) \\ &+ \sum_{z \in \{C_h \mid 1 \le h \le k, h \notin \{s,t\}\}} a_{iz}^{(l-1)} \cdot (a_{iz} - a_{zk}) \\ &= \sum_{z \in C_s} a_{iz}^{(l-1)} \cdot (p-q) - \sum_{z \in C_t} a_{iz}^{(l-1)} \cdot (p-q). \end{aligned}$$

Then, according to Lemma 5 and Lemma 6,

Lemma 5. *Let $G_f \sim G_f(nk,k,p,q), \{v_i, v_d, v_e\} \in C_s$, then we have $a_{id}^{(h)} = a_{ie}^{(h)}$.*

Lemma 6. *Let $G_f \sim G_f(nk,k,p,q), v_i \in C_s, \{v_r, v_s\} \in C_t$, we have $a_{ir}^{(h)} = a_{is}^{(h)}$.*

we can have $\sum_{z \in C_t} a_{iz}^{(l-1)} = n a_{ij}^{(l-1)}$ and $\sum_{z \in C_s} a_{iz}^{(l-1)} = n a_{ik}^{(l-1)}$.

Therefore,

$$\begin{aligned} a_{ij}^{(l)} - a_{ik}^{(l)} &= (p-q)(n a_{ij}^{(l-1)} - n a_{ik}^{(l-1)}) \\ &= n^{l-1}(p-q)^{l-1}(a_{ij} - a_{ik}) = n^{l-1}(p-q)^{l} > 0. \end{aligned}$$

This completes the proof of the inequality Eq. (4.1). □

Observations. We have two main observations:

- From Lemma 4, most existing HOPs hold the assumption of homophily. In the embedding space, distance of two nodes residing in different communities is inherently larger than that of those in the same community.

- Eq. (4.2) reveals that HOP between any pair of nodes v_i and v_j essentially represents the total similarity of a sequence of nodes traversed by all possible walk paths from v_i to v_j, and all walk paths share the same contributions to represent $a_{ij}^{(l)}$ regardless of various downstream graph tasks. However, we argue that every walk path should have task-relevant contributions to $a_{ij}^{(l)}$. This inspires the following question: *are there some specific walk paths that characterise the task-relevant HOP w.r.t. homophily or heterophily?* Subsequently, to explicitly reveal the characteristics of HOP, we

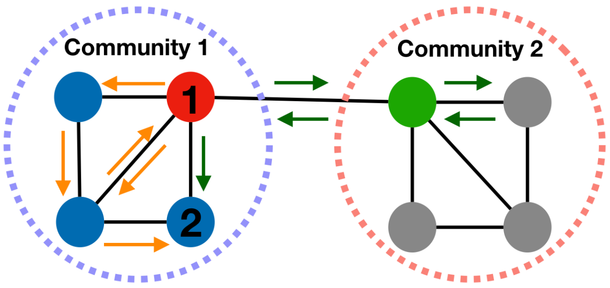


Fig. 4.2 Illustration: 1) Micro-level (orange arrows within the community 1) and macro-level (dark green arrows across communities 1 and 2) walk paths represent the 5th-order proximity between nodes 1 and 2; 2) The green node contributes more centrality to the red node than any blue node.

categorise all possible walk paths into micro-level and macro-level walk paths as two complementary components to represent HOP. Our proposed definitions are below.

Definition 7. *(**Micro-level walk path**) A micro-level walk path connecting v_i and v_j is a sequence of vertices $V' = (v_i, v_k, \cdots, v_j)$ traversed a sequence of edges $E' = (e_1, e_1, \cdots, e_n)$ that ensures $V' \subseteq C_i$ and $E' \subseteq C_i$.*

Definition 8. *(**Macro-level walk path**) A macro-level walk path connecting v_i and v_j is a sequence of vertices $V' = (v_i, v_k, \cdots, v_j)$ traversed a sequence of edges $E' = (e_1, e_1, \cdots, e_n)$ and $E' \cap C_j \neq \phi$ and $E' \cap C_i \neq \phi$ and $i \neq j$.*

For example, in Fig 4.2, we observe that orange (micro-level) path and dark-green (macro-level) path play very different roles to induce a five-step connectivity pattern from node 1 to node 2, although both can contribute to 5th-order proximity between node 1 and 2. Under the homophily assumption, the orange path has more expressive power than the dark-green path since it is likely to leverage tightly close neighbourhood similarity within a community, which may benefit community detection and node classification tasks. In contrast, under the heterophily assumption, the dark-green path has more appropriate expressive power than the orange path to heterophily since it tends to use weak-connectivity and distant neighbourhood similarity across communities, good for structural role classification. Moreover, only homophily or heterophily cannot be suitable for all graph-based tasks.

In general, micro/macro-level walk paths allow us to capture the structure of the traversed region and provide an attention mechanism to guide the walk. This allows us to focus on task-

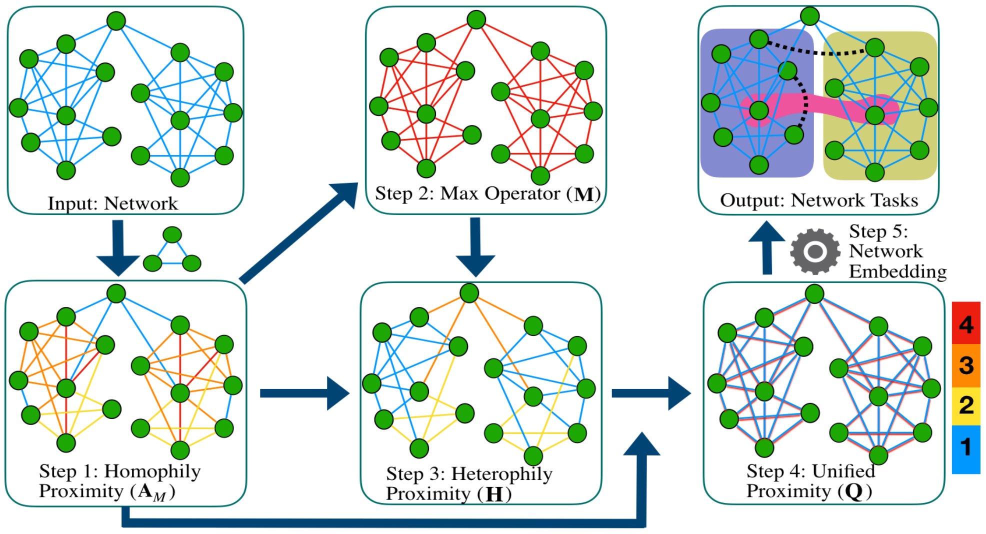

Input: Network
Step 2: Max Operator (**M**)
Output: Network Tasks
Step 5: Network Embedding
Step 1: Homophily Proximity ($\mathbf{A}_M$)
Step 3: Heterophily Proximity (**H**)
Step 4: Unified Proximity (**Q**)
4
3
2
1

Fig. 4.3 The H^{2}GT framework. H^{2}GT first constructs homophily proximity ($\mathbf{A}_M$) by triangle representations. This homophily proximity is proceeded by a maximum operation (**M**) and then can generate heterophily proximity (**H**). We finally linearly combine $\mathbf{A}_M$ and **H** to form unification proximity **Q** which can be integrated with any existing graph embedding methods. The colour bar indicates the weight scales of edges.

relevant parts of the graph while eliminating the noise in the rest of the graph which results in the graph embedding that provides better predictive performance. Note that different from the breadth-first and depth-first approximately search in node2vec, fixed length of micro-level and macro-level walk paths will be exhaustedly and accurately considered to represent HOP between two nodes. While they share a general idea that representation of a node is determined by its neighbourhoods that need to be flexibly defined.

Building on the above discussions, in the following we propose H^{2}GT that defines new HOP to flexibly preserve both homophily and heterophily by characterising walk paths with micro-level and macro-level walk paths. It is a generic model that can be used as a preprocessing step to provide input to any graph embedding methods. A whole process of H^{2}GT is illustrated in the Fig. 4.3, and it will be elaborately discussed in the following sections.

4.2.2 Homophily Proximity Representation

To represent homophily proximity, the adjacency matrix **A** or random walk matrix **P** is widely used but both are noisy and sparse. Another choice is to perform community detection so that homophily character of each edge can be clearly shown, such as by spectral clustering [79]). However, this will lead to a binary-valued output edge (i.e., within or across communities) and limits our ability in exploring unseen patterns of a graph.

To address the above challenges, we adopt a scalable and *smooth* community detection method with motif representations [10] to represent homophily proximity as $\mathbf{A}_M$,

$$\mathbf{A}_M(i,j) = \sum_{v_i,v_j \in \mathcal{V}} \mathbb{1}\left(v_i, v_j \text{ occur in } M\right), \tag{4.3}$$

where $i \neq j$, v_i and v_j belong to motif M , and $\mathbb{1}(s)$ is the truth-value indicator function, i.e., $\mathbb{1}(s)$ = 1 if the statement s is true and 0 otherwise. Note that the weight is added to $\mathbf{A}_M(i,j)$ only if node v_i and v_j occur in the given motif M. In this chapter, we only focus on undirected triangle motif, but it can be easily generated to any other type of motifs. The intuition of homophily proximity representation (Eq. (4.3)) is that motif representation smooths out the neighbourhood over the graph, acting as denoise filter due to removed edges that do not participate in any motif.

4.2.3 Heterophily Proximity Representation

Most of existing graph embedding algorithms (e.g., DeepWalk, AROPE) hold a homophily assumption in HOP but overlook heterophily. To discover heterophily proximity, we encourage macro-level walk paths to contribute more to proximity. The reason is that structurally similar nodes tend to play the same role and the defined macro-level walk path can identify such roles by a centrality measure. Centrality measure that can identify the importance of nodes is an effective tool to identify hub [135] and bridge [6] roles. Thus, we consider a contribution centrality [6] (Definition 9) into each step of the defined macro-level walk path such that *encoded* contribution centrality instructs walks to access distant nodes in different communities and to connect nodes acting as bridge roles.

Definition 9. ***(Contribution Centrality)*** *Contribution centrality of a node v_i is a measure of the importance of v_i as a central node in G, which is determined by the contribution of its immediate neighbours as bridge roles to access distant neighbours in different communities.*

From the above definition, a node bridging different communities contributes more centrality to a central node if a central node can only access nodes in different communities via the bridging node. For example, considering centrality of red node in Fig. 4.2, centrality contribution from green to red is larger than that from any immediate blue nodes. The reason is that the red one can access the grey ones only through the green, and the blue nodes are redundant for the red one because it can access directly each blue node without any intermediary. Therefore, the macro-level walk path strategy with encoded contribution centrality can identify structurally similar (i.e., bridge) but distant nodes.

However, estimation of such macro-level walk paths seems not straightforward to overcome because heterophily information is not explicitly provided in most common graphs. A signed network $G^{\pm} = (G^{+}, G^{-})$ has both positive and negative edge weights, where positive relations G^{+} encode friendship, and negative relations G^{-} encode enmity interactions [61]. Essentially, a signed network is represented by two networks with largely different structural properties.

Inspired by signed networks, we propose motif-aware signed networks $G^{\pm}_{M} = (G^{+}_{M}, G^{-}_{M})$. where G^{+}_{M} and G^{-}_{M} are homophily and heterophily graphs respectively. An intuition of designing $G^{\pm}_{M}$ is based on a random walk theory that homophily/heterophily edges increase the probability of staying in/escaping from a community. We represent homophily as $\mathbf{A}_M$ shown in the last section. We represent heterophily as $-\mathbf{A}_M$ by turning all values in $\mathbf{A}_M$ into their opposite numbers. We interpret that the larger non-zero value of $-\mathbf{A}_M(i,j)$, the higher chance to achieve macro-level walk paths that explore global structures. However, most graph embedding methods do not have capacity to handle negative weights. To develop a general and universal graph transformer model, we use the following simple but efficient approach to transform negatives into positives while preserving heterophily:

$$\mathbf{H} = -\mathbf{A}_M + \mathbf{M}, \tag{4.4}$$

where

$$\mathbf{M}(i,j) = \begin{cases} f_{max}(\mathbf{A}_M) & v_i, v_j \text{ are contained in a triangle,} \\ 0 & \text{otherwise,} \end{cases}$$

where $f_{max}(\mathbf{A}_M)$ indicates a maximum value in $\mathbf{A}_M$. Heterophily proximity ($\mathbf{H}$) can be achieved by Eq. (4.4).

We give more attention to the edge containing more triangles via a maximum operation. The heterophily proximity $\mathbf{H}$ is an effective instantiation of macro-level walk paths to represent heterophily information because it can encode the below two aspects of information: 1) for distant nodes, the value of entries in $\mathbf{H}$ reflects the ease of accessing distant neighbours residing in different communities; 2) for contribution centrality, the weight $\mathbf{H}(i,j)$ between the central node v_i and its neighbours v_j indicates the contribution of v_j as a bridge role to access indirect neighbours of v_i. Therefore, our heterophily proximity $\mathbf{H}$ can efficiently encode heterophily information in graphs. Also, we will validated its efficiency by conducting experiments on the role classification task.

4.2.4 Unification and Instantiations

We linearly unify the homophily and heterophily proximity as follows:

$$\mathbf{Q} = \mathbf{A}_M + \lambda \mathbf{H}, \tag{4.5}$$

However, micro-level walk paths are still able to represent proximity with increasing of orders due to the inherit graph structure. To prevent it, we need to enlarge the difference of entries in $\mathbf{H}$. Considering the scalability issue in the graph embedding field, we focus on a linear combination (λ), but similar ideas (e.g., non-linear operators) can be straightforwardly generalised. The first-order proximity $\mathbf{Q}$ flexibly shows homophily and heterophily characteristics by hyperparameter λ that controls over the importance of heterophily. The Eq. (4.4) models a heterophily representation, which aims to give more weights to edges acting as a bridge role connecting two communities. The Eq. (4.5) linearly combines and heterophily and homophily representations with a hyperparameter λ, which helps us to flexibly preserve both representations. By tuning λ, the unification proximity $\mathbf{Q}$ enables existing homophily-based methods (e.g., AROPE) to preserve heterophily proximity in node embedding. Therefore, combining our H^{2}GT with existing homophily-based methods overcomes the limitation of one specific proximity and integrates both. Fig 4.3 illustrates the proposed H^{2}GT framework.

The H^{2}GT framework has three key benefits:

1. **Unification.** The proposed H^{2}GT embodies both homophily and heterophily, with the trade-off hyperparameter (λ) determined by cross validation. This enables existing graph embedding methods to break the limitation of preserving either homophily or heterophily by simply combining with our framework.

2. **Simplification.** To generate a new graph as illustrated in Fig 4.3, it only involves some simple graph transformation, such as max operation, subtraction and summation of matrix, without any optimisation process. Motif counting [5, 97] has some mature solutions to be addressed in large-scale graphs. Although it is of simplification, it still can offer a solution to preserve homophily and heterophily together with partial mathematical guarantee and strong intuition. Moreover, benefiting from its simplification, it is easy to implement.

3. **Efficiency.** It is able to improve efficiency of a whole pipeline from an input original graph to a final graph embedding through using an extra enhancer H^{2}GT (show experiment results in Section 4.3.8). The reason is two-fold. First, our proposed H^{2}GT enjoys simplification, and thus does not give much pressure on whole learning pipeline. Second, the sparsity of output unified representation $\mathbf{Q}$ is commonly lower than that of the input original graph. When constructing the homophily proximity, some edges

Algorithm 4 H^{2}GT

```
Input: Adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
Output: Unification of homophily and heterophily proximity representation $\mathbf{Q} \in \mathbb{R}^{n \times n}$
1: Construct a homophily matrix $\mathbf{A}_M \in \mathbb{R}^{n \times n}$.
2: Construct a maximality $\mathbf{M}$ from $\mathbf{A}_M$ by converting all non-zero entries in $\mathbf{M}$ to a maximum value in $\mathbf{A}_M$.
3: Construct a heterophily matrix $\mathbf{H} = \mathbf{M} - \mathbf{A}_M$.
4: Construct an unification matrix $\mathbf{Q} = \mathbf{A}_M + \lambda \mathbf{H}$.
```

are removed due to not contained in any triangle. It will improve the efficiency of the latter combined method if its computation relies on the sparsity of the graph.

Our framework H^{2}GT can be integrated with any graph embedding methods as a pre-processing step to provide input to them. Here we select three representative algorithms, AROPE on matrix factorisation, DeepWalk on random walks and GCN on convolutional neural networks.

- **H^{2}GT-AROPE** (H^{2}GT-A). To preserve HOP, we use a linear combination of power of biased matrix $\mathbf{Q}$ as follows,

$$\mathbf{P}_M = w_1 \mathbf{Q} + w_2 \mathbf{Q}^2 + \ldots + w_l \mathbf{Q}^l. \tag{4.6}$$

  When $\lambda = 0$, we interpret $\mathbf{Q}^r(i,j)$ $(1 \leq r \leq l)$ as the total number of motifs traversed by all possible l-length walk paths connecting nodes v_i and v_j. Increasing λ results in more similarity from global neighbours to represent the HOP between v_i and v_j. Moreover, benefiting from AROPE, H^{2}GT-A can explore arbitrary walk length l between two nodes without increase computational complexity.

- **H^{2}GT-DeepWalk** (H^{2}GT-D). The objective function of DeepWalk can be written as:

$$\max_{\Phi} \log \Pr\left(\{v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}\} \,|\Phi(v_i)\right), \tag{4.7}$$

where w is the window size, $\Phi(v_i)$ is the representation of v_i. Under the proposed H^{2}GT framework, instead of uniformly random sampling neighbours of v_i, we adopt a biased sampling strategy. Different from the biased sampling strategy in node2vec that considers a second-order random walks by tuning two out-in parameters, H^{2}GT-D only needs to tune one unifying parameter λ. A small λ makes it more likely to sample local, close neighbours. By contrast, a large λ makes it more likely to sample global, far-away neighbours.

- **H^{2}GT-GCN** (H^{2}GT-G). The layer-wise propagation rule in GCN can be written as,

$$\mathbf{F}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{Q}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{F}^{(l)}\mathbf{W}^{(l)}\right), \tag{4.8}$$

where $\tilde{\mathbf{Q}} = \mathbf{Q} + \mathbf{I}_N$, $\mathbf{I}_N$ is an identity matrix, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{Q}}_{ij}$, $\mathbf{W}^{(l)}$ is a layer-specific trainable weight matrix, $\mathbf{F}^{(l)}$ is an activation in the l^{th} layer; $\mathbf{H}^{(0)}$ is the given feature matrix, and $\sigma(\cdot)$ is an activation function. An activation function, such as ReLU [43], is a function that is added into an artificial neural network to enable the network learns complex non-linear patterns in the data. If the feature of nodes is not given, we set $\mathbf{H}^{(0)}$ as one-hot encoding. From Eq. (4.8), H^{2}GT-G can discriminate the neighbourhoods. Specifically, it will give more attention to neighbourhoods that has homophily assumption if λ is small, and otherwise, it will give more attention to neighbourhoods that has heterophily assumption.

4.2.5 Complexity Analysis

The time complexity of homophily representation can be as large as $O(n^3)$ for a complete graph, where n is the number of nodes in the graph. Let t is the number of triangle of a input graph G. While most real graphs are far from complete so the actual complexity is much lower than $t < O(n^3)$. According to empirical study in [9], the value of t in real-world graphs is linear with $|E|$. For heterophily and unification step, each has the same complexity, which is the number of non-zero entries $j \leq |E|$ in $\mathbf{A}_M$. Only $j = |E|$ when the graph is complete. Thus the total complexity of H^{2}GT is $O(n^3)$+$O(j)$ after ignoring lower order terms. The running time of H^{2}GT is dominated by the time of finding all the triangles, which can be $O\left(n^3\right)$. In addition, the computational complexity of both heterophily and unification operations linearly depends on the number of edges j.

Table 4.2 Statistics of graphs with isolated nodes removed. #Test Triangles indicates the number of removed triangles as testing set for motif prediction.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	Edge Density	Labels	#Test Triangles
Amherst	2,021	81,492	40.3	15	10K
Hamilton	2,116	87,486	41.3	15	10K
Mich	2,924	54,903	18.7	13	10K
Rochester	4,140	14,5309	35.1	19	10K
Brazil	131	1,038	7.9	4	200
Europe	399	5,995	15.0	4	300
USA	1,190	13,599	11.4	4	500

4.3 Experiments

4.3.1 Datasets

We conduct extensive experiments on the following seven real networks covering social networks [1] and traffic graphs [2]. Statistics of graphs are shown in Table 4.2:

1. Amherst, Hamilton, Mich, Rochester [104]: They are the Facebook social networks at different universities in US. Nodes are students and links represents friendship links between students' pages. We use *class year* as the *node labels* in *node classification*.

2. Brazil, Europe, USA [87]: They are air-traffic networks. Nodes indicate airports and edges correspond to commercial airlines. We use the level of airport activity (e.g., passenger traffic) as the node labels in structural role classification.

4.3.2 Baselines

We extensively compare the proposed H^{2}GT with the following eight state-of-the-art methods covering graph embedding methods and a SOTA graph transformer method:

[1]https://escience.rpi.edu/data/DA/fb100/
[2]https://github.com/leoribeiro/struc2vec

1. Deepwalk[3]: This approach learns an embedding by sampling random walks from each node, applying SkipGram learning on those walks. We vary the window size{1, 2, 3, 4, 5, 6} and use default settings for other hyperparameters.

2. LINE[4]: It defines loss functions to preserve the first-order or second-order proximity separately. We study two versions of LINE that preserves the first-order proximity (LINE-1st) and second-order proximity (LINE-2nd). We use the default settings for other hyperparameters.

3. node2vec[5]: It defines a flexible random walk approach that extends DeepWalk by adding two parameters inward (i) and outward (o), so as to control DeepWalk's random walk sampling. The special case with parameters $i = 1$, $o = 1$ corresponds to DeepWalk. We finely tune the bias hyperparameters inward, outward from {0.25, 0.5, 1, 2, 4} and use the default settings for other hyperparameters.

4. AROPE[6]: It is a matrix factorisation based method that preserves the polynomial-based HOP (Eq. (2.20)). It is able to preserve arbitrary-order proximity and achieves high scalability as it only performs eigen-decomposition to the adjacency matrix by re-weighting eigen-decomposition. We tune the number of preserved higher-order proximity {1, 2, 3, 4, 5, 6} and $w_i = 0.1^i$.

5. struc2vec[7]:It first encodes the node structural role similarity into a multilayer graph. DeepWalk is then performed on this cobstructed multilayer graph to learn vertex representations, such that vertices close to each other in the multilayer graph (with high structural role similarity) are embedded closely in the new representation space. We study all four different optimisation strategies of struc2vec.

[3]https://github.com/phanein/deepwalk
[4]https://github.com/snowkylin/line
[5]https://github.com/aditya-grover/node2vec
[6]https://github.com/ZW-ZHANG/AROPE
[7]https://github.com/leoribeiro/struc2vec

6. Graph Neural Network[8]: It is derived from spectral graph convolutions [14, 25]. Neighbourhood aggregation is a key that step each node only aggregates feature vectors of its first-order neighbours. We use the default settings to conduct experiments.

7. MotifSC: It is an unsupervised motif-based spectral clustering method. It aims to minimise the conductance after producing partitions. we use the default settings to conduct experiments.

8. Graph diffusion convention (GDC) [9] [57]: This approach is a graph transformer model. A new graph is generated by firstly constructing a diffusion graph obtained by polynomial function, and then sparsitying this diffusion graph by setting a threshold. However, it does not incorporate heterophily assumption. We study two variants of GDC, namely personal pagerank and heat kernel. Furthermore, we combine GDC with arope (GDC-A), DeepWalk (GDC-D) and GCN (GDC-G) with recomended transport probability $\{0.05, 0.15, 0.3\}$, exponential in heat kernel $\{1, 5, 10\}$, and sparsity threshold $\{ 0.00001, 0.0001, 0.001, 0.01 \}$

For our method, we study three variations: H^2GT-A, H^2GT-D and H^2GT-G. H^2GT-A and H^2GT-D take the number of proximity as $\{1, 2, 3, 4, 5, 6\}$ and $\lambda = \{0.1, 0.3, 0.5, 0.7, 1.3, 1.5, 1.7\}$. H^2GT-G uses the same λ with other variant but only uses two layers.

The dimension of embedding vector is 128 for all social networks and 16 for all traffic networks considering the number of nodes in graphs. The best performance results for all methods will be reported, with termination of the computation if no complete result is returned within twelve hours. We use the open-source Python library GEM[10] [44] to study all methods under the same software framework. All experiments were performed on a Linux machine with 2.4GHz Intel Core and 16G memory.

[8]https://github.com/tkipf/pygcn
[9]https://github.com/klicperajo/gdc
[10]https://github.com/palash1992/GEM

4.3.3 Evaluation Metrics

For motif prediction, we use precision@N_p to evaluate the performance [110, 135]. It is defined as:

$$\text{precision } @N_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \delta_i, \tag{4.9}$$

where δ_i =1 means the i-th reconstructed motif is correct (i.e., the reconstructed motif exists in the graph), δ_i =0 otherwise and N_p is the number of evaluated motifs. For node and structural role classification, we use accuracy, i.e. the percentage of nodes whose labels are correctly classified, to evaluate the performance [128]:

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left(\hat{y}_i = y_i\right), \tag{4.10}$$

where $\hat{y}$ and y are predicted label and true label respectively, $\mathbb{I}$ is an indicator operator (1 if two labels are equal otherwise 0).

4.3.4 Performance for Motif Prediction

Besides links, motifs are small subgraphs fundamental in graphs. Thus, prediction of motif structures is important in real applications. Therefore, we design the motif prediction task as a generalised link prediction task in our evaluation. We focus on fundamental triangle prediction task, though it can be generalised to other motif structures. Triangle/Motif prediction is not as commonly evaluated as link prediction. A recent paper in [10] points out the importance of triangles/motifs in real-world applications and calls for more attention. Thus, this chapter addresses such a practical need that many other methods fail to address. GCN and H^{2}GT-G are not studied here since GCN is primarily designed for node classification tasks.

In our experiments, we first randomly remove some triangles to be used as testing set. The number of removed triangles are shown in Table 4.2 (the right most column). Then we train all models on the rest of the graph. Note that the summation of the number of triangles in testing and training are not equal with the total number of triangles in the original

Table 4.3 Motif prediction (i.e. generalised link prediction) results reported in precision @N_p. The best results are in **bold** and the second best ones are <u>underlined</u>. We set N_p to 500 for all traffic graphs and 10k for all Facebook social networks.

Methods	Amherst	Hamilton	Mich	Rochester	Brazil	USA	Europe
MotifSC	0.658	0.654	0.702	0.873	0.096	0.187	0.331
AROPE	<u>0.894</u>	<u>0.898</u>	<u>0.928</u>	0.955	<u>0.548</u>	<u>0.985</u>	<u>0.742</u>
DeepWalk	0.639	0.658	0.789	0.864	0.050	0.060	0.035
LINE-1st	0.082	0.084	0.085	0.088	0.066	0.088	0.082
LINE-2nd	0.310	0.310	0.307	0.319	0.075	0.062	0.060
node2vec	0.094	0.085	0.087	0.097	0.125	0.100	0.295
struc2vec	0.167	0.181	0.225	0.152	0.387	0.628	0.126
GDC-A	0.679	0.734	0.665	0.774	0.515	0.881	0.635
GDC-D	0.821	0.811	-	-	0.121	0.397	0.282
H^2GT-A	**0.928**	**0.927**	0.927	**0.978**	**0.578**	**0.986**	**0.747**
H^2GT-D	0.864	0.857	**0.938**	<u>0.972</u>	0.098	0.777	0.324

graph since triangle structures are correlated with each other in a graph. To evaluate the performance, we take the following five steps:

1. Positive sampling: we sample existing triangles (i.e., testing set) in the original graph.

2. Negative sampling: we sample three-node tuples and ensure every tuple cannot compose triangles in the original graph. Its quantity is ten times over positive samples.

3. After obtaining embedding of nodes, we calculate the mean of tuplewise similarity (e.g., dot product) in positive and negative sampling sets.

4. Mix and sort similarity of negative and positive together, and use precision@N_p to evaluate. Here, we set the maximal N_p as 500 for all traffic networks and 10,000 for all Facebook social networks (noting the total number of triangles is at exponential scale) and with the reasoning that for a good model, the similarity of positive sampling should be larger than that of negative sampling.

5. Calculate all precision@N_p from 1 to maximum N_p and average them. Finally, the average results of 5 runs are reported in Table 4.3.

We have following observations:

Table 4.4 Node classification results (accuracy) on three datasets. The best results are in **bold** and the second best ones are underlined. The results of LINE-1st and LINE-2nd have much lower accuracy so they are not shown.

	Hamilton					Rochester					Mich				
%Labels	2%	4%	6%	8%	10%	2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
MotifSC	0.209	0.218	0.255	0.249	0.327	0.210	0.221	0.244	0.279	0.303	0.208	0.233	0.231	0.239	0.241
AROPE	0.770	0.840	0.862	0.874	0.876	0.717	0.770	0.790	0.799	**0.811**	0.465	0.506	0.524	0.537	0.542
DeepWalk	0.721	0.804	0.842	0.861	0.864	0.711	0.768	0.787	0.798	0.807	0.464	0.506	0.523	0.536	0.547
node2vec	0.277	0.317	0.331	0.348	0.354	0.252	0.292	0.319	0.334	0.340	0.226	0.247	0.258	0.257	0.257
GCN	0.649	0.679	0.704	0.742	0.747	0.605	0.660	0.650	0.680	0.675	0.422	0.494	**0.531**	**0.549**	**0.550**
struc2vec	0.218	0.238	0.251	0.260	0.271	0.201	0.207	0.211	0.214	0.215	0.196	0.208	0.220	0.224	0.225
GDC-D	0.758	0.829	0.848	0.859	0.862	0.690	0.738	0.756	0.765	0.773	0.475	0.505	0.513	0.524	0.532
GDC-A	0.224	0.284	0.317	0.313	0.423	0.212	0.238	0.274	0.332	0.389	0.211	0.237	0.247	0.242	0.251
GDC-G	0.481	0.592	0.660	0.730	0.719	0.187	0.421	0.269	0.431	0.413	0.260	0.329	0.380	0.477	0.409
H^2GT-D	**0.789**	**0.843**	**0.866**	**0.877**	**0.879**	**0.745**	**0.780**	**0.796**	**0.802**	**0.811**	**0.485**	**0.507**	0.525	0.535	0.541
H^2GT-A	0.729	0.795	0.815	0.824	0.829	0.680	0.723	0.744	0.744	0.754	0.438	0.450	0.460	0.470	0.465
H^2GT-G	0.642	0.688	0.730	0.738	0.745	0.605	0.646	0.662	0.674	0.669	0.412	0.484	0.511	0.529	0.533

1. H^2GT-A achieves the overall best performance over all datasets, and AROPE achieves the second best.

2. H^2GT-A improves the AROPE by 2.23% on average. Additionally, H^2GT-D can improve DeepWalk by 23.9% for all social networks on average, and it even can improve more than ten times for two sparse USA and Europe traffic networks. It shows the effectiveness of H^2GT for preserving triangle structures in embedding space.

4.3.5 Performance for Node Classification

We evaluate the node classification performance. Specifically, we randomly select a portion of nodes as training set and leave the rest as test set. Then, we train a one-vs-all logistic regression with L2 regularisation. We repeat the process for 10 times and report the average accuracy in Table 4.4. We have following observations:

1. H^2GT-D achieves the overall best performance.

2. H^2GT-D and H^2GT-A have poorer results than DeepWalk and AROPE, i.e., there is degradation rather than improvement. The reason could be matrix factorisation and convolutional neural network are less sensitive to heterophily.

Table 4.5 Structural role classification results on Brazil, Europe and USA with 90% training data. The results of LINE-1st and LINE-2nd have much lower accuracy so they are not shown. The best results are in **bold** and the second best ones are underlined.

Datasets	MotifSC	AROPE	DeepWalk	node2vec	GCN	struc2vec	GDC-D	GDC-A	GDC-G	H^2GT-D	H^2GT-A	H^2GT-G
Brazil	0.564	<u>0.686</u>	0.429	0.450	0.379	**0.736**	0.607	0.436	0.428	0.514	0.664	0.500
Europe	0.365	0.535	0.365	0.422	0.362	<u>0.568</u>	0.530	0.452	0.350	0.430	**0.577**	0.450
USA	0.379	0.589	0.493	0.479	0.549	<u>0.608</u>	0.588	0.519	0.403	**0.629**	0.600	0.565

4.3.6 Performance for Structural Role Classification

Earlier, we heuristically show that the H^2GT can help preserve node centrality as an application of the structural role classification. To validate the effectiveness, we conduct this task on Brazil, USA and Europe and show result in Table 4.5 with 90% training ratio. We observe that:

1. H^2GT-A and H^2GT-D achieve the best performance on Europe and USA respectively. Struc2vec is specifically designed to this task so it achieves better performance on Brazil than H^2GT-based methods. It could be caused by sparsity problem of H^2GT-based methods due to motif representation, and especially for Brazil, the most sparse graph among all datasets.

2. H^2GT-based method can improve the overall performance of original methods. For example, in Europe, H^2GT-D, H^2GT-A and H^2GT-G improve 17.8%, 7.9% and 24.3% over original DeepWalk, AROPE, and GCN respectively.

4.3.7 Sensitivity Analysis

We conduct a sensitivity study for two hyperparamters: the order of HOP P and unifying weight λ, as shown in Fig. 4.4. The left of Fig. 4.4 shows the performance variation of H^2GT-A for motif prediction task on Hamilton and Rochester. We see that H^2GT-A is less sensitive to the unifying weight than the number of HOP. The right of Fig. 4.4 shows the performance variation of H^2GT-D for node classification task on Hamilton and Rochester. We see that H^2GT-D is more sensitive to the unifying weight than the number of HOP.

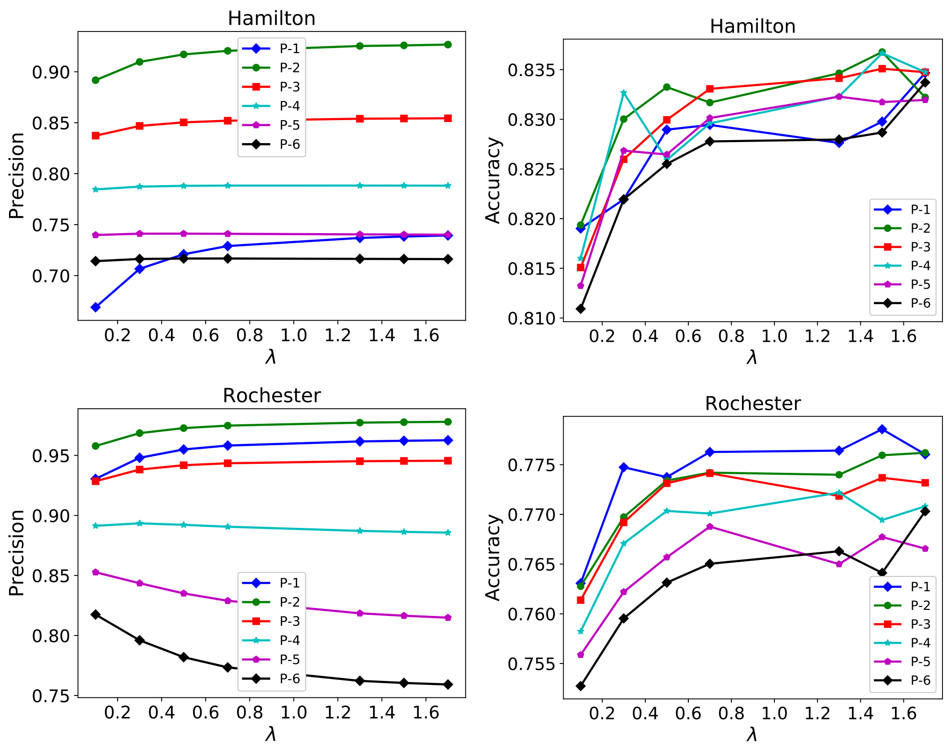


Fig. 4.4 Sensitivity analysis on two hyperparameters proximity order P and unifying weight λ. Left: H^{2}GT-A for motif prediction on Hamilton and Rochester. Right: H^{2}GT-D for node classification on Hamilton and Rochester.

4.3.8 Computational Time

Table 4.6 compares the computational time of the original AROPE, DeepWalk and GCN with H^{2}GT-A, H^{2}GT-D and H^{2}GT-G. For our H^{2}GT, the computational time includes the whole pipeline from input original graph to output graph embedding. We have two key observations:

1. Our H^{2}GT-A, H^{2}GT-D and H^{2}GT-G can improve efficiency of AROPE, DeepWalk, and GCN by 35.1%, 45.7% and 10.1% respectively. This is because our H^{2}GT only uses local motif structures and sparsifies the original graph, which accelerates the optimisation process of combined methods and improves efficiency.

2. We further study the overhead of the motif representation calculation. Our studies show that the motif calculation is not the most important part in computational cost, e.g. it accounts for only 5% and 8% of the total time of H^{2}GT-A on Amherst and Rochester,

Table 4.6 Computational time (in seconds). The last row shows the average with the most efficient result in **bold** and the second <u>underlined</u>. We can see that our H^2GT-A, H^2GT-D and H^2GT-G can improve efficiency of AROPE, DeepWalk, and GCN by 35.1%, 45.7% and 10.1% respectively.

Datasets	AROPE	H^2GT-A	DeepWalk	H^2GT-D	GCN	H^2GT-G
Amherst	3.99	2.79	216.97	99.17	26.46	28.34
Hamilton	4.06	2.84	218.87	112.23	32.57	28.10
Mich	5.35	2.63	306.91	121.21	25.46	19.89
Rochester	5.40	3.96	443.54	311.06	72.17	64.42
Average	<u>4.70</u>	**3.05**	296.57	160.92	39.16	35.19

respectively. Thus, the efficiency gain due to increased sparsity has exceeded this small overhead, leading to an overall improvement of computational efficiency.

4.4 Summary

In this chapter, we proposed an H^2GT framework that makes use of motif representations to transform a graph into a new graph preserving both homophily and heterophily via flexible and complementary micro-level and macro-level walk paths. H^2GT can be integrated with any existing graph embedding methods without requiring changing their cores such that it can take advantage of powerful graph embedding methods proposed recently. We conducted experiments on node classification, structural role classification and newly designed motif prediction to show the superior prediction performance and computational efficiency improvement of H^2GT over SOTA methods.

Chapter 5

Trustworthiness-aware Knowledge Graph for Recommendation via Motifs

5.1 Introduction

Knowledge-aware recommendation has shown great potential to improve accuracy and explainability. However, when incorporating KGs into RS, most existing methods, including the introduced methods in Sec. 2.5.2, do not consider noise in KGs. In real-world KGs, some noise is inevitably introduced in the process of automatically constructing large-scale KGs due to limited labour supervision [119, 54]. We argue that such noise in KGs as auxiliary data can degrade the performance of RS, which will be verified by our experiments. As illustrated in Fig. 5.1, the red dashed arrow indicates an interaction to be predicted between a user and a movie *Death Becomes Her*. Assuming that this user has interacted with three similar movies *Back to The Future I & II* and *Forrest Gump* due to the same director *Robert Zemeckis* by using the KG. The correctness of the director of *Death Becomes Her* can determine whether to recommend it to this user. In this case, we fail to recommend *Death Becomes Her* if a noisy triple (*Death Becomes Her, IsDirectedBy, Christopher Nolan*) exists. Therefore, it is essential to tolerate such noisy triples in the KG incorporated with RS since KGR, as an important way to integrate with RS, heavily relies on triples.

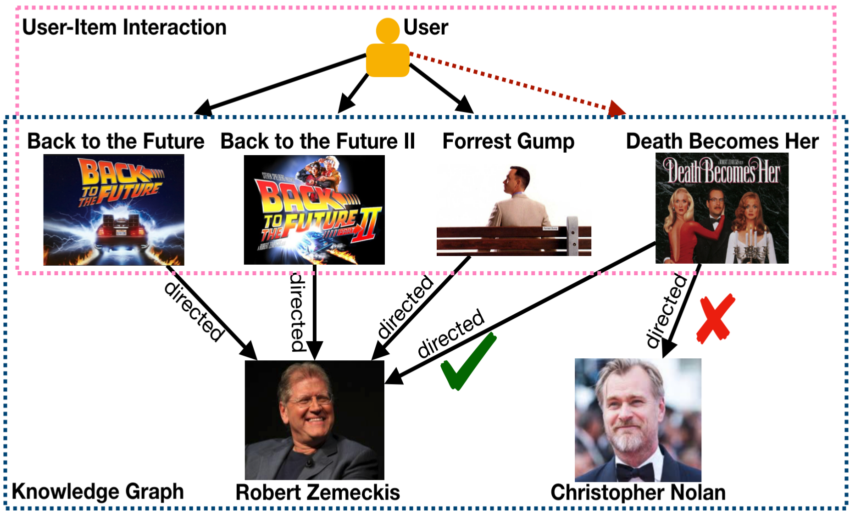


Fig. 5.1 An example to show that a noisy triple (*Death Becomes Her, IsDirectedBy, Christopher Nolan*) can degrade the recommendation performance (i.e., recommend interesting *Death Becomes Her* to the user). It motivates our model to tolerate such noisy triples.

In this chapter, we aim to estimate noise in KGs, while constructing noise-tolerant KGR to incorporate with RS. However, there remains two challenges: 1) Noise estimation in **any** KG. Some works [32, 64] strongly rely on external information (e.g., web content, text) but do not have good generalisation to estimate noise in KGs. Therefore, it is a challenge that how to estimate noise without relying on external information to enhance generalisation; 2) Noise estimation integration. Some existing works [129, 18] study an integration between two modules (KGR and RS) through, for example, linearly combining the entity and the corresponding item embeddings. However, building on this two-module integration, introducing another noise estimation module is still unclear.

To address the above challenges, we propose a novel method **trust**worthiness-aware KGR for **rec**ommendations (TrustRec). TrustRec incorporates noise-tolerant translation-based KGR into a CF-based method through a trustworthiness estimator, which gives the degree of certainty of triples. Specifically, to construct this trustworthiness estimator, we firstly leverage internal structural information in KGs from microscopic to macroscopic levels: the motif (co-occurrence in the same type of local connectivity pattern), communities (co-occurrence in the same high association group) and global information (correlation strength on all paths). Then we use a neural network architecture to fuse the structural information, and finally yield a trustworthiness value for every triple. By this way, we can estimate triple trustworthiness in any KG by leveraging internal information to enhance generalisation capacity, which tackles the first challenge. To address the second challenge, building on our estimator, we integrate triple trustworthiness into a proposed neural/weighted pairwise ranking loss functions for noise-tolerant KGR. Meanwhile, we integrate entity trustworthiness as a linear combination ratio of an entity embedding to learn a noise-tolerant item representation for RS. We summarise our contributions of this chapter as follows:

1. We propose a trustworthiness estimator to take noise in KGs into account.
2. We propose trustworthiness integration to learn noise-tolerant KGR and item representations for RS.
3. We conduct extensive experiments to show the superior performance of TrustRec over SOTA methods.

5.2 Methodology

In this section, we propose to estimate trustworthiness of triples through internal structural information: motifs, communities and global information. We then integrate triple trustworthiness into a weighted/neural loss function of KGR to learn noise-tolerant KGR. Meanwhile we integrate entity trustworthiness into RS to learn noise-tolerant item representations for RS.

5.2.1 Motif-aware Trustworthiness

Triangular motifs (shown in left bottom of Fig. 5.3) demonstrate very important local structures underlying various complex networks, such as social networks.

We use the strength of a tie between head and tail linked by a relation to measure the trustworthiness of triple (e_h, r, e_t). If head e_h and tail e_t have a strong tie, the relation r between head e_h and tail e_t is expected to be strong. Motif modelling is an effective approach to measure the strength of a tie between two entities [10, 105]. For example, in

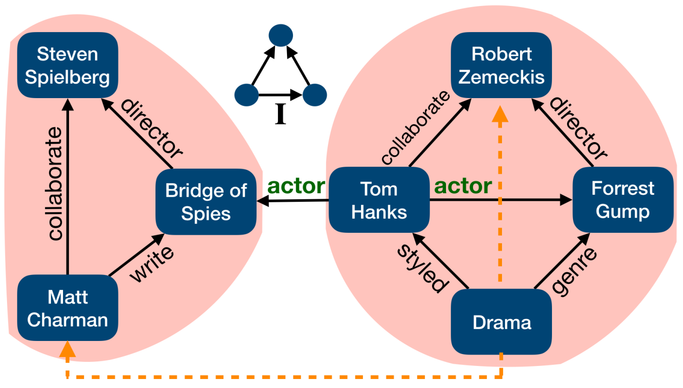


Fig. 5.2 An example to show 1) a strong tie occurs in the triple *(Tom Hanks, actor, Forrest Gump)* through the motif I; 2) high association in the triple *(Tom Hanks, actor, Forrest Gump)* within a community (shadow); 3) high correlation (an orange dashed arrow) occurs in an entity pair (*Drama*, *Robert Zemecks*) through all paths.

a social network, two people who have a common friend are likely to be friends, so this common friend and two people constitute a triangular motif connectivity pattern. Intuitively, if two people have more common friends, the stronger strength of a tie between them can occur. Additionally, considering motifs can capture the rich context of relations to diversify strengthen of ties while direct edges relation cannot. For example, in Fig. 5.2, if only considering the simple edge relation, triples *(Tom Hanks, actor, Bridge of Spies)* and *(Tom Hanks, actor, Forrest Gump)* has the same strength of a tie. However, when considering a motif type I in Fig. 5.2, the triple *(Tom Hanks, actor, Forrest Gump)* has a rich context (e.g., with *Robert Zemeckis*) to enhance its strength of a tie. This chapter will focus on all triangular motifs as shown in Fig. 5.3, though our proposed method can be easily extended to other motifs.

Based on the above analysis, we take the input (e_h, r, e_t) from G, and quantify the strength of a tie for it by counting the number of the motif type $\mathcal{M}_i$ containing this triple. Different type of triangular motifs reflect different connectivity patterns. Thus, we construct a feature vector $\mathbf{m}(e_h, r, e_t)$ to consider all, and the i^{th} entry in $\mathbf{m}(e_h, r, e_t)$ are decided by:

$$\mathbf{m}_i(e_h, r, e_t) = \sum_{e_h, e_t \in \mathcal{E}, \mathrm{r} \in \mathcal{R}} \mathbb{1}(e_h, r, e_t \text{ occur in } \mathcal{M}_i)\,, \tag{5.1}$$

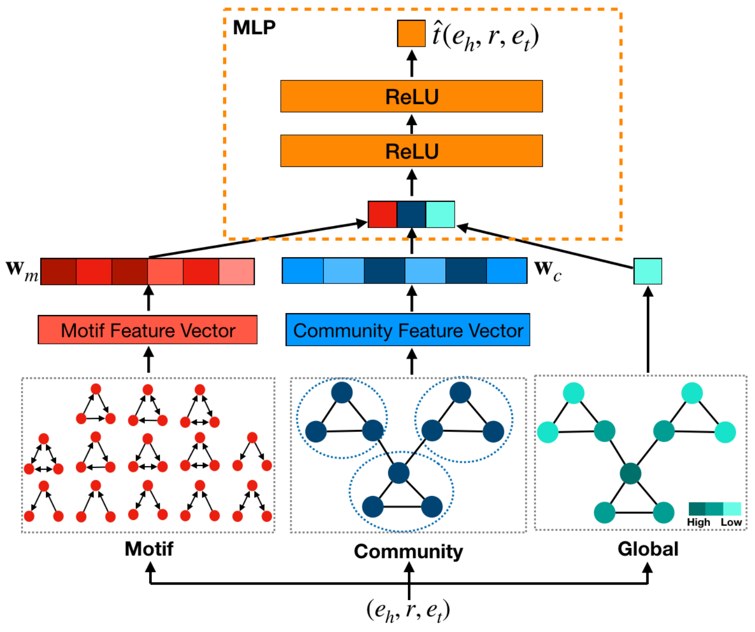


Fig. 5.3 The framework of the proposed trustworthiness estimator by leveraging internal structure information of KGs: motifs, communities and global information.

where $\mathbb{1}(s)$ is the truth-value indicator function, i.e., $\mathbb{1}(s) = 1$ if the statement s is true and 0 otherwise. It shares the similar idea with motif-based clustering in the Chapter 3. We form a feature vector $\mathbf{m}(e_h, r, e_t)$ where the i-th element indicates the number of motif type $\mathcal{M}_i$ containing (e_h, r, e_t). We then compress the motif feature vector $\mathbf{m}(e_h, r, e_t)$ into a value $m(e_h, r, e_t)$ by a trainable weight $\mathbf{w}_m$ as $m(e_h, r, e_t) = \mathbf{m}(e_h, r, e_t) \cdot \delta(\mathbf{w}_m^T)$, We interpret the value $\mathbf{w}_m(i)$ as the importance of motif type $\mathcal{M}_i$. Note that all triples in the KG share the same trainable weight $\mathbf{w}_m$ to largely avoid the increase of the model complexity with the increase of KG size.

5.2.2 Community-aware Trustworthiness

The motif-aware trustworthiness estimator based on the local neighbours is straightforward but cannot take fully advantage of rich structural information of KGs. To capture a more complete picture of triples, we consider a community structure that consists of a group of entities. Community structure refers to the occurrence of groups of nodes in a graph that are more densely connected than with the rest of the graph. Some existing works [114, 20] show that entities within a community have relatively higher association than entities in different communities. If head e_h and tail e_t have a higher association, head e_h and tail e_t are more likely to have a trusted relation. For example, in Fig. 5.2, the same relation *actor* connecting an intra-community entity pair *(Tom Hanks, Forrest Gump)* is more trustful than it in an inter-community entity pair *(Tom Hanks, Bridge of Spies)*.

Inspired by the above, we thus perform community detection task on G. Since our focus is the association of triples, we first convert all directed edges in G to undirected ones and form a graph G_u. For the graph G_u, we then use a spectral clustering (SC) [108] method to cluster G_u into k communities $\mathbb{S} = \{S_1, \ldots, S_k\}$. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an adjacency matrix of weighted graph G_u where the entry $\mathbf{A}(i,j)$ is the number of relations between e_i and e_j. The degree matrix $\mathbf{D}$ is a diagonal matrix with diagonal entries $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{A}(i,j)$, which is the degree of the entity e_i. We then construct a Laplacian matrix $\mathbf{L}$ as follows:

$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}, \tag{5.2}$$

where $\mathbf{I}_n$ is an identity matrix. SC aims to learn a spectral embedding $\mathbf{Z} \in \mathbb{R}^{n\times k}$ by optimizing a function as follows:

$$\min_{\mathbf{Z}} \operatorname{tr}\left(\mathbf{Z}^T\mathbf{L}\mathbf{Y}\right), \text{ s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}, \tag{5.3}$$

where $\operatorname{tr}(\cdot)$ is the trace function. The above function can be solved by eigenvalue decomposition of $\mathbf{L}$, i.e., $\mathbf{Z} = [\mathbf{z_1}, \mathbf{z_2}, \cdots, \mathbf{z_k}]$ are the eigenvectors corresponding to the smallest k eigenvalues of $\mathbf{L}$. To find clusters, SC then uses $\mathbf{Z}$ as an input to perform k-means.

The changing number of communities can determine the state that whether a triple (e_h,r, e_t) are in the same community. Our model thus contains multiple states to represent community-aware trustworthiness by constructing a community indicator vector $\mathbf{c}$. The i^{th} entry in $\mathbf{c}(e_h, r, e_t)$ is determined by:

$$\mathbf{c}_i^j(e_h, r, e_t) = \begin{cases} 1 & (e_h, r, e_t) \in S_n \text{ with } j \text{ partitions,} \\ 0 & \text{otherwise,} \end{cases} \tag{5.4}$$

where $1 \leqslant j \leqslant k$. Consistent with compression operation in motif-aware trustworthiness, we have a community-aware trustworthiness $c(e_h, r, e_t) = \mathbf{c}(e_h, r, e_t) \cdot \delta(\mathbf{w}_c^T)$, where $\mathbf{w}_c$ is a shared trainable weight to indicate the importance of the number of communities.

5.2.3 Global-structure-aware Trustworthiness

Motif-aware and community-aware estimators mainly focus on microscopic and mesoscopic structural information. The global structure, one important macroscopic description of the graph structure, is a complementary component to represent the trustworthiness of triples in KGs. Therefore, to consider the global structure, we introduce the concept of correlation strength that captures how difficult to reach a tail entity e_t from a head entity e_h through a sequence of relations in a ***whole*** graph. For example, in Fig. 5.2, there are dense paths from *Drama* to *Robert Zemecks* (e.g., *Drama* $\rightarrow$ *Tom Hanks* $\rightarrow$ *Robert Zemecks*), that is, there is a high correlation between them. By contrast, it is impossible to reach from *Drama* to *Matt Charman* following all paths in the graph.

To instantiate the above idea, we adopt source allocation theory in PageRank [81] to characterise the correlation strength for triples. We assume that the trustworthiness between entity pairs (e_h, e_t) will be higher, and more resource is passed from the head e_h through all paths to the tail e_t in a whole graph G. The amount of resource aggregated into e_t indicates the trustworthiness between e_h and e_t. Specifically, starting from e_h each node in the graph should be reached. In the initial state, the resource amount of e_h is 1, and all others are 0. In the process of resource allocation, the sum of all resources of nodes is always 1. We simulate resource flowing until distribution steady. The value of the resource on the tail entity is $p(e_t|e_h)$, it is calculated as follows:

$$p(e_t|e_h) = (1-\alpha) \sum_{e_i \in \mathcal{D}} \frac{p(e_i \mid e_h) \cdot w_{e_i t}}{d(e_i)} + \frac{\alpha}{n}, \tag{5.5}$$

where $\mathcal{D}$ is a set of entities that have outgoing links to the entity e_t, $w_{e_i t}$ is the weight from the e_i to e_t, $d(e_i)$ is the out-degree of the entity e_i. Thus, for each entity e_i in $\mathcal{D}$, the resource flows from e_i to e_t should be $\frac{p(e_i|e_h) \cdot w_{e_i t}}{d(e_i)}$. The entities without outgoing links can cause the absorption of the resource. To prevent it, resource flow from each entity may directly jump to a random entity with the same probability α. This part of the resource that flows to e_t randomly is $\frac{1}{n}$.

5.2.4 Fusion of Estimators

We use a neural network structure multi-layer perceptron (MLP) to extract a final trustworthiness from three estimators. Note that the way of extraction is not limited to MLP, and we can use a more elaborate design of the neural network. For the triple (e_h, r, e_t), we first concatenate the above three-level trustworthiness $\mathbf{x}(e_h, r, e_t) = [m(e_h, r, e_t), c(e_h, r, e_t), p(e_t|e_h)]$. The vector $\mathbf{x}(e_h, r, e_t)$ will be input into the MLP and transformed passing L hidden layers as follows:

$$\begin{aligned} \hat{t}(e_h, r, e_t) &= \mathcal{M}(\mathcal{M}(\cdots \mathcal{M}(\mathbf{x}(e_h, r, e_t)))) \\ &= \mathcal{M}^L(\mathbf{x}(e_h, r, e_t)), \end{aligned} \tag{5.6}$$

where $\mathcal{M}(\mathbf{x}) = \sigma(\mathbf{W}_m \mathbf{x} + \mathbf{b}_m)$ is a fully-connected neural network layer with weight $\mathbf{W}_m$, bias $\mathbf{b}_m$, and nonlinear ReLU activation function $\sigma(\cdot)$. In the output layer of $\mathcal{M}^L(\cdot)$, we use a sigmoid function $\delta(\cdot)$ to ensure the returned $\hat{t}(e_h, r, e_t)$ in the range 0 to 1. The whole framework of the trustworthiness estimator is shown in Fig. 5.3.

5.2.5 Trustworthiness Integration

After obtaining trustworthiness of triples, TrustRec follows the conventional translation-based KGR to incorporate with CF. To inject auxiliary information from KG to RS, some existing works study the integration between two modules KGR and RS (e.g., linearly combine the entity and the corresponding item embeddings). When considering an additional trustworthiness estimator module, we propose trustworthiness integration with both KGR and RS. Specifically, we propose triple trustworthiness integration to learn noise-tolerant KGR, and entity trustworthiness integration to learn noise-tolerant item representations of RS.

For the triple trustworthiness integration, we propose a weighted and a neural margin-based ranking loss (MRL) of KGR.

Weighted MRL. The idea is that a triple with higher trustworthiness should be more important when training KGR. Based on it, we construct a weighted MRL as below

$$\mathcal{L}_k^{(w)} = \sum_{\substack{(e_h,r,e_t)\in\mathcal{G} \\ (e'_h,r,e'_t)\in\mathcal{G}^-}} \hat{t}(e_h,r,e_t)\cdot[\gamma+g_R(e_h,r,e_t)-g_R(e'_h,r,e'_t)]_+, \tag{5.7}$$

where $[\cdot]_+ \triangleq \max(0,\cdot)$, $\mathcal{G}^-$ contains incorrect triplets constructed by replacing head entity or tail entity in a valid triple randomly, and γ controls the margin between positive and negative triples, and $g_R(\cdot)$ is the energy function of TransR. We choose TransR because TrustRec is equivalent with CKE if our trustworthiness estimator is neglected, which can gain insights about the effect of our estimator. To learn noise-tolerant KGR, trustworthiness $\hat{t}(e_h,r,e_t)$ instructs our model to pay more attention on those more trustful triples.

Neural MRL. The idea is that if $\hat{t}(e_h,r,e_t)$ is involved in a parameterised way to determine the score of the energy function, TrustRec itself will learn to integrate trustworthiness for noise-tolerant KGR. For example, if $\hat{t}(e_h,r,e_t)$ negligibly contributes to the energy score of (e_h,r,e_t), TrustRec can assign very low trustworthiness to it. Thus, we first perform a concatenation operation $\mathbf{n}(e_h,r,e_t) = [\hat{t}(e_h,r,e_t), g_D(e_h,r,e_t)]$. We then construct a neural MRL as below:

$$\mathcal{L}_k^{(n)} = \sum_{\substack{(e_h,r,e_t)\in\mathcal{G} \\ (e'_h,r,e'_t)\in\mathcal{G}^-}} \left[\gamma+\mathcal{N}^L(\mathbf{n}(e_h,r,e_t))-g_D(e'_h,r,e'_t)\right]_+, \tag{5.8}$$

where $\mathcal{N}(\mathbf{x}) = \sigma(\mathbf{W}_n\mathbf{x}+\mathbf{b}_n)$. Here, we use the energy function of TransD because of a consideration of different types of entities in KGs and a study of the diverse KGR methods on TrustRec.

Integration with RS. Some existing works linearly combine the entity and corresponding item embedding as the final item embedding as

$$\mathbf{i}' = \mathbf{e} + \mathbf{q}_i. \tag{5.9}$$

However, the final item embedding $\mathbf{i}'$ contains noise from knowledge. Therefore, to learn noise-tolerant item representations of RS, we assume that if an entity is likely to be involved in triples with high trustworthiness, this entity has high combination ratio to form $\mathbf{i}'$. We propose entity trustworthiness that is an averaged summation of the triple trustworthiness it involves. It is formulated as below:

$$\hat{t}(e) = \frac{\sum_{e'_t \in \mathcal{E}, \mathrm{r}' \in \mathcal{R}} \hat{t}(e, r', e'_t)}{n_h} + \frac{\sum_{e'_h \in \mathcal{E}, \mathrm{r}' \in \mathcal{R}} \hat{t}(e'_h, r', e)}{n_t}, \tag{5.10}$$

where n_h and n_t are the number of triples that the entity e acts as heads and tails. TrustRec treats $\hat{t}(e)$ as an integration ratio of entity e, and formulates

$$\mathbf{q}'_i = \hat{t}(e) \cdot \mathbf{e} + \mathbf{q}_i, \tag{5.11}$$

where $\mathbf{q}_i$ is a learned latent vector of item i by MF. We then develop two variants of TrustRec depending on the overall loss. TrustRec(W) uses the overall loss

$$\mathcal{L}^{(w)} = \mathcal{L}_k^{(w)} + \mathcal{L}_r. \tag{5.12}$$

The TrustRec(N) uses the overall loss:

$$\mathcal{L}^{(n)} = \mathcal{L}_k^{(n)} + \mathcal{L}_r. \tag{5.13}$$

Table 5.1 Statistics of DBbook2014 and MovieLen-1M

		DBbook2014	MovieLens-1M
Rec	# Users	5,576	6,040
	# Item	2,680	3,240
	# Ratings	65,961	998,539
	# Avg. ratings	12	165
	# Completeness	0.4%	5.1%
KG	# Entity	13,882	14,708
	# Relation	13	20
	# Triple	334,511	434,189

5.3 Experiments

5.3.1 Datasets

We use two public datasets in the book and movie domains: DBbook2014 [1], MovieLens-1M[2].

1. DBbook2014: This dataset provides ratings for book recommendations, which consists of 5,578 users and 2,680 books.

2. MovieLens-1M: It is a popular benchmark dataset in movie recommendations, which consists of approximately 1 million explicit ratings on the MovieLens website.

Items in these two domains are mapped into DBPedia entities if there is a mapping available. DBpedia extracts triples from Wikipedia pages, and contains broad scope of entities covering different areas of knowledge. The extracted DBPedia is released by the paper [18]. Table 5.1 shows the statistics of datasets.

Following most item recommendation works that models implicit feedback, we treat existing ratings as positive interactions, and generate negative ones by randomly corrupting items. To study the effect of noisy triples, we generate noisy triples to be 10%, 20%, 30% and 100% of existing triples by the following protocol: for an existing triple (e_h, r, e_t) in a training set, we generate a corresponding noisy one by randomly replacing its head (e'_h, r, e_t) or tail (e_h, r, e'_t) while ensuring that 1) it cannot be found in the existing KG; 2) it

[1]http://2014.eswc-conferences.org/important-dates.html
[2]https://grouplens.org/datasets/movielens/1m/

contains at least one item; 3) for noise injection, triple (e_h, r, e_t) is replaced with (e'_h, r, e_t) (or (e_h, r, e'_t)) to ensure the total number of triples is unchanged. To validate the effect of KG-aware methods, we consider an item cold-start and sparsity scenarios. For both datasets, 25% of items in valid and test sets cannot be found in the train set. We randomly sparsify 70% interactions of MovieLens-1M since from Table 5.1 its completeness is more than ten times than the completeness of DBBook2014.

5.3.2 Baselines

We compare TrustRec with the following six SOTA RS methods:

1. Collaborative Filtering with Knowledge Graph (CFKG)[3] [133]: This method constructs an user-item KG and the relation is decided by user behaviours (i.e., review, brand, category, bought-together). This KG will be combined with the item-side KG by shared common items. It then uses translational recommendation to minimise the loss.

2. Collaborative Knowledge Embedding (CKE)[3] [129]: This approach applies matrix-factorisation-based CF to knowledge-base embedding for recommendation, which uses TransR to learn entity and relation embedding.

3. Knowledge Co-Knowledge factorization model (CoFM)[3] [84]: It studies the effect of knowledge transfer between item recommendations and KG completion via a co-factorisation model which can be seen as a transfer learning model.

4. Knowledge Translation-based User Preference model (KTUP) [3] [18]: KTUP models various implicit relations between users and items and transfer knowledge learned from TransH, which reveals the preferences of users on consuming items. Additionally, it provides explainability via aligned relations and preferences.

5. Knowledge Graph Convolutional Networks (KGCN)[4] [112]: It extends the GCN to the KG by aggregating neighbourhood information selectively and biasedly, which

[3]https://github.com/TaoMiner/joint-kg-recommender
[4]https://github.com/hwwang55/KGCN

Table 5.2 The comparison results about recommendation on the KG dataset without noise injection. The best results are in **bold** and the second best ones are <u>underlined</u>.

	DBBook2014 (@5, %)			MovieLens-1M (@10, %)		
	F1	Precision	Recall	F1	Precision	Recall
CoFM	2.56	2.04	4.39	2.71	2.72	4.01
CKE	6.08	4.86	10.07	3.47	3.46	<u>5.38</u>
CFKG	2.70	2.14	4.55	2.51	2.67	3.54
KTUP	4.72	3.82	7.78	3.54	3.64	5.16
KGCN	2.10	1.54	3.31	2.02	1.89	2.23
KGNN-LS	2.15	1.65	3.11	1.88	1.85	1.94
TrustRec (N)	**6.33**	**5.04**	**10.64**	**3.80**	**3.84**	**5.57**
TrustRec (W)	<u>6.25</u>	<u>5.02</u>	<u>10.17</u>	<u>3.66</u>	<u>3.80</u>	5.27

simultaneously learns both structural information and semantic information from the KG as well as users' preferences and potential interests.

6. Knowledge-aware Graph Neural Networks (GNN) with Label Smoothness regularisation (KGNN-LS)[5] [111]:This approach incorporates GNN architecture into KGs after converting KGs to weighted homogeneous graphs. This conversion uses a user-specific relation scoring functions and then aggregates neighbourhood information with different weights. In addition, KGNN-LS proposes label smoothness constraint to provide strong regularisation for learning the edge weights in KGs.

We construct the training set, validation set and testing set by randomly splitting the dataset with the ratio of $7:1:2$. Each experiment is repeated five times, and the average performance is reported. For hyperparameters, the learning rate of all methods is searched in $\{0.0005, 0.001, 0.005, 0.01\}$, the embedding size in $\{16, 32, 64\}$, We use an open-source PyTorch library to study all methods under the same software framework released by [18]. All trainable parameters are optimised by Adam algorithm. The co-efficient of L_2 regularisation is 10^{-5}. The batch size is 512. We perform early stopping strategy on validation sets. All other hyperparameters use default settings. At the beginning of training, we assume all triples are correct, and initialise the triple trustworthiness as 1. All experiments were performed on a Linux machine with 2.4GHz Intel Core and 8G memory.

[5]https://github.com/hwwang55/KGNN-LS

Table 5.3 The comparison results about recommendation on the effect of noisy triples. The best results are in **bold** and the second best ones are <u>underlined</u>.

	DBBook2014 (P@5, %)				MovieLens-1M (P@10, %)			
Noise Ratio	0.1	0.2	0.3	1.0	0.1	0.2	0.3	1.0
CoFM	2.06	2.00	1.98	1.90	2.89	2.89	2.83	2.56
CKE	4.88	4.72	4.70	<u>4.70</u>	<u>3.62</u>	<u>3.59</u>	3.47	3.33
CFKG	1.86	1.76	1.40	0.84	2.28	2.38	2.10	1.27
KTUP	3.46	3.72	3.98	3.24	3.51	3.45	3.43	3.50
KGCN	1.48	1.68	1.50	1.74	2.11	1.70	1.84	1.53
KGNN-LS	1.61	1.60	1.79	1.76	1.90	1.84	1.92	1.76
TrustRec (N)	**4.98**	**4.98**	**4.92**	**4.80**	**3.66**	**3.61**	**3.71**	**3.48**
TrustRec (W)	<u>4.96</u>	<u>4.86</u>	<u>4.76</u>	4.56	3.61	3.51	<u>3.53</u>	<u>3.46</u>

5.3.3 Evaluation Metrics

In recommendation, we use the trained model to select K items with highest predicted click probability for each user in the test set, and choose $F1@K$, $Precision@K(P@K)$ and $Recall@K$. For KG completion, we use $Hit\ ratio@K$.

1. Hit ratio@K: It is 1 if a correct items are recommended within the top K items, otherwise 0. We compute the mean of all users as the final hit ratio score.
2. F1-score@K: It is the combination mean of precision at rank K and recall at rank K.
3. Precision@K: It is the fraction of the items recommended that are relevant to the user. We compute the mean of all users as the final precision.
4. Recall@K: It is the proportion of the items relevant to the user that have been successfully recommended. We compute the mean of all users as the final recall.

5.3.4 Performance for Recommendations

For recommendations, we evaluate methods in three scenarios w.r.t KG datasets without noise injection, effect of noisy triples and top-K recommendations.

In Table 5.2, we show the performance comparison on DBook2014 and MovieLens-1M with DBPedia without noise injection. We observe that:

Table 5.4 The comparison results on top-K recommendation. The best results are in **bold** and the second best ones are <u>underlined</u>.

	DBBook2014 (P, %)					MovieLens-1M (P, %)				
Top K	@3	@5	@10	@15	@20	@3	@5	@10	@15	@20
CoFM	2.62	2.00	1.47	1.35	1.23	3.34	3.05	2.78	2.69	2.57
CKE	5.89	4.77	3.43	2.83	<u>2.46</u>	<u>4.41</u>	<u>4.05</u>	3.49	3.29	3.10
CFKG	1.83	1.60	1.33	1.15	1.04	2.53	2.32	2.14	2.05	2.02
KTUP	4.21	3.64	2.73	2.32	2.02	3.98	3.69	3.51	3.24	3.03
KGCN	2.11	1.59	1.06	0.83	0.68	2.52	2.16	1.81	1.56	1.45
KGNN-LS	2.16	1.68	1.13	0.88	0.72	2.29	2.15	1.85	1.64	1.46
TrustRec (N)	**6.07**	**4.94**	**3.59**	**2.92**	**2.53**	**4.48**	**4.09**	**3.66**	**3.39**	**3.19**
TrustRec (W)	<u>5.96</u>	<u>4.83</u>	<u>3.48</u>	<u>2.84</u>	2.44	4.25	4.01	<u>3.58</u>	<u>3.33</u>	<u>3.12</u>

1. our proposed TrustRec(N) consistently achieves the best performance and TrustRec(W) achieves the second best 5 out of 6 settings.

2. TrustRec(N) outperforms TrustRec(W) because it can flexibly learn a proper way to incorporate triple trustworthiness into the energy function.

3. Two GNN-based methods do not show superior performance because their node features are randomly generated and thus such features are not related with information of KGs.

4. TrustRec(W) is superior over CKE, which indicates the efficacy of our trustworthiness estimator.

In Table 5.3, we show performance comparison over the effect of noisy triples. We observe the following:

1. With the increase of noisy triples our TrustRec(N) consistently outperforms compared methods. Also our TrustRec(W) can achieve the second best 5 out of 8 settings. It indicates our both methods are noise-tolerant.

2. The overall performance of all methods is degraded with the increase of noisy triples.

For top-K recommendation, in Table 5.4, we report the performance of KG-aware methods over precision at $K = \{3, 5, 10, 15, 20\}$. For each K, we report the averaged performance

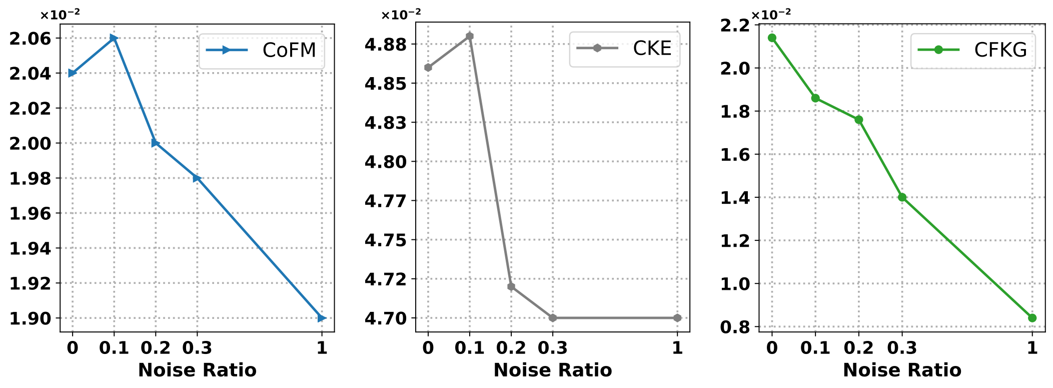


Fig. 5.4 Effect of noisy triples in DBBook2014 (The y-axis indicates the metric Precision@5).

over a range of noise ratio $\{0, 0.1, 0.2, 0.3, 1.0\}$ because noise in KGs can significantly affect performance. We can see that our TrustRec(N) is consistently superior over all baselines.

5.3.5 Performance for KG Completion

We evaluate on a KG completion task that predicts the missing entity e_h or e_t. For each missing entity, we take all entities as candidates and rank them according to the scores computed based on entity and relation embeddings. Fig. 5.5 shows the overall performance with the increase of ratio of noisy triples. We do not show the performance of TrustRec(N) since feeding all unseen triples (more than 100 billion in DBBook2014) to our neural energy function is unfeasible. From Fig. 5.5, we observe that TrustRec(W) has superior performance over SOTA KG-aware RS.

5.3.6 Sensitivity Analysis

Firstly, we study the effect of noisy triples on recommendations. In Fig. 5.4, we show the performance of three existing KG-aware RS methods CoFM, CKE and CFKG on DBBook2014 w.r.t Precision@5. We observe that:

1. with the increase of noise ratio the overall performance of all three methods are degraded. It indicates that noisy triples negatively affect the performance of KG-aware methods.

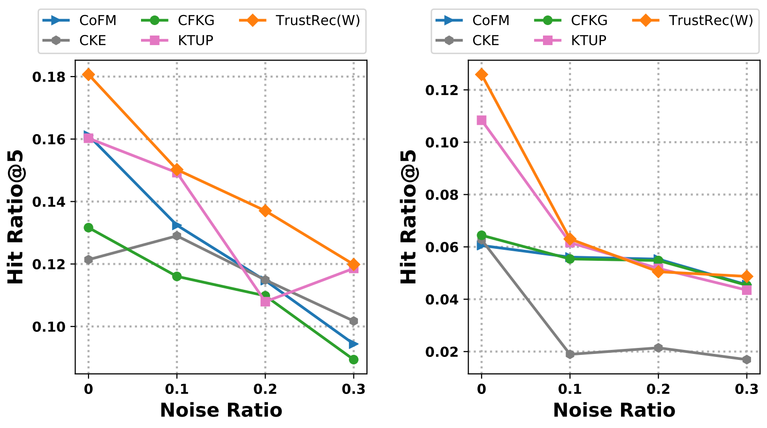


Fig. 5.5 The comparison results on the KG completion task for DBPedia in DBBook2014 (left) and MovieLens-1M (right). The higher *Hit Ratio*@5 indicates the better performance.

Table 5.5 Computational Time (in seconds)

	CoFM	CKE	CFKG	KTUP	KGCN	KGNN-LS	TrustRec(N)	TrustRec(W)
DBBook2014	596	856	654	912	3281	1221	1536	1167
MovieLens-1M	2105	2955	1986	2230	7556	5393	7219	5368

2. The effect of noisy triples is different for different methods. For example, noisy triples have more effect on CFKG than CoFM. This is because the embedding of entity with noise in CoFM is reweighted to determine ratings while CFKG are not.

5.3.7 Computational Time

We show computational time of all compared methods in Table 5.5. We observe the following:

1. KGCN and KGNN-LS are not efficient due to personalised relation score function.
2. Our TrustRec(N) and TrustRec(W) are not efficient because both need to train a trustworthiness estimator while all baselines do not need to train it.
3. Our TrustRec(W) is more efficient than TrustRec(N) because TrustRec(W) uses a direct weighted loss function to avoid training an additional neural network.

5.4 Summary

In this chapter, we proposed TrustRec that can estimate trustworthiness of triples in KGs through an estimator that uses motifs, communities and global information. Based on this, we proposed triple trustworthiness integration to learn noise-tolerant KGR, and entity trustworthiness to learn noise-tolerant item representations of RS. We conducted experiments to show that TrustRec outperforms SOTA methods.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

This thesis has investigated GRL methods with motif structures. We summarise this thesis via the tackled research challenges as shown in Chapter 1 as below:

1. **Structure preservation of different orders in clusters.** Both edges and motif structures are important in the real-world graphs. However, existing SC methods only explicitly encode either edges or motif structures, which are limited to considering a single order. To address this issue, in Chapter 3, we proposed a novel mixed-order spectral clustering framework to model both second-order and third-order structures simultaneously. To model mixed-order structures, we proposed two new methods based on graph Laplacian and random walks. MOSC-GL combines edge and triangle adjacency matrices, with theoretical performance guarantee. MOSC-RW combines first-order and second-order random walks for a probabilistic interpretation. Moreover, we designed mixed-order cut criteria to enable existing SC methods to preserve mixed-order structures, and developed new mixed-order evaluation metrics for structure-level evaluation.

2. **Inflexibility of GRL assumptions.** In real-world graphs, both homophily and heterophily assumptions can co-exist, and thus considering only one could limit the

prediction performance and interpretability. However, there is no general and universal solution that takes both into consideration. In Chapter 4, we proposed a simple yet powerful framework H^2GT to capture HOP that flexibly unifies homophily and heterophily. Specifically, H^2GT utilises motif representations to transform a graph into a new graph with a hybrid assumption via micro-level and macro-level walk paths. H^2GT can be used as an enhancer to be integrated with any existing graph embedding methods without requiring any changes to latter methods. Because H^2GT can sparsify graphs with motif structures, it can also improve the computational efficiency of existing graph embedding methods when integrated.

3. **Noise-tolerant KGR for RS.** For large-scale KGs, due to limited labour supervision, noises are inevitably introduced during automatic construction, which can hurt the performance of knowledge-aware applications. In Chapter 5, we integrated KGs into RS. However, most existing KG-aware RS do not consider such noises in KGs, which can degrade the performance of KG-aware RS. To address this issue, we proposed a novel method TrustRec. TrustRec introduced a trustworthiness estimator into noise-tolerant KGR methods for collaborative filtering. Specifically, to assign trustworthiness, we leveraged internal structures of KGs from microscopic to macroscopic levels: motifs, communities and global information. Building on this estimator, we then proposed trustworthiness integration to learn noise-tolerant KGR and item representations for RS.

6.2 Future Directions

While some fundamental problems in GRL have been addressed in this thesis, there are still many open problems to be considered. This section outlines three research topics that worth further investigation.

1. **Higher-order structures enrichment for GRL.** In this thesis, we only investigate triangular motifs for GRL. For the other more complex motifs, such as M_{bifan} and M_{loop}

in Fig. 1.2, this thesis does not extend to it. The type of motifs grows significantly with the increase of orders, so it is impractical to enumerate all types motifs in large-scale graphs. To ease the enumeration, we can incorporate some prior knowledge that cannot be automatically identified. For example, triangle structures play an important role in social networks. A few works such as [136] study the four-node loop and the five-node star but it is limited to these two types of motifs and undirected graphs.

2. **Dynamic GRL.** Most existing works on GRL has focused on static graphs. The real-world graphs (e.g., financial transaction graphs) are not always static and can evolve over time. Therefore, dynamic graphs are becoming an increasingly important topic of study. In such graphs, new nodes and/or new edges can appear while old ones can disappear, which changes graph structures. Besides graph structures, the node/edge context information can be also changed with varying time information. Dynamic graphs have unique challenges to be tackled in terms of graph structures and context information. Although a few attempts such as [134] can be applied to dynamic graphs, they are still limited to fixed node size and do not incorporate important motif structures into embedding space. Therefore, extending graph embedding techniques to consider the dynamic character will open a variety of attractive application domains.

3. **Substructure representations learning**. Node embedding is the most common output for GRL. Also, we study the relation (i.e., edge) embedding in KGs. In contrast to node/edge embedding, the substructure (e.g., clique, community) embedding and whole-graph embedding aims to represent the small-world graph and whole graph as a low-dimensional vector, which can benefit to visualisation and graph classification. Some existing works find some relations between node embedding and substructure embedding [121, 20]. For example, learning node embedding can benefit to learn community embedding [20]. However, these works do not study the effect of motifs on substructure embedding, although we show the importance of motif in graphs. Therefore, how to incorporate motifs into substructure embedding is an open problem.

4. **Motif-aware graph analysis in finance.** Graph analysis plays an important role in finance. In financial graphs, nodes are banks. If two banks share the same borrowers, there will be a link between them. In the constructed financial graphs, some existing works measure the systemic risk regarding vulnerability and reliability [41, 29, 2]. Vulnerability is an ability to maintain network functions under attacks or failures [96]. Reliability measures how long the network system performs effectively under attacks [96]. Some recent works [30] have revealed the importance of motif structures for the measurement of vulnerability and reliability in graphs (e.g., power-grid networks). In financial graphs, banking lending behaviours are co-related with each other, which is the syndicated bank lending [48]. For example, a group of banks share the same borrowers. For this case, motifs can model intrinsically interdependent and interactive banking behaviours, rather than edge structures that only capture independent and individual banking behaviour. Therefore, how to assess systemic risks through modelling motifs is an open problem.

References

[1] Abboud, A. and Williams, V. V. (2014). Popular conjectures imply strong lower bounds for dynamic problems. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 434–443.

[2] Acemoglu, D., Ozdaglar, A., and Tahbaz-Salehi, A. (2015). Systemic risk and stability in financial networks. *American Economic Review*, 105(2):564–608.

[3] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

[4] Adamic, L. A. and Glance, N. (2005). The political blogosphere and the 2004 us election: divided they blog. In *ACM 3rd International Workshop on Link discovery*, pages 36–43.

[5] Alon, N., Dao, P., Hajirasouliha, I., Hormozdiari, F., and Sahinalp, S. C. (2008). Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249.

[6] Alvarez-Socorro, A., Herrera-Almarza, G., and González-Díaz, L. (2015). Eigencentrality based on dissimilarity measures reveals central nodes in complex networks. *Scientific Reports*, 5:17095.

[7] Ana, L. and Jain, A. K. (2003). Robust data clustering. In *IEEE Computer Vision and Pattern Recognition*, pages 128–136.

[8] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *the Semantic Web*, pages 722–735. Springer.

[9] Benson, A. R., Gleich, D. F., and Leskovec, J. (2015). Tensor spectral clustering for partitioning higher-order network structures. In *SIAM International Conference on Data Mining*, pages 118–126.

[10] Benson, A. R., Gleich, D. F., and Leskovec, J. (2016). Higher-order organization of complex networks. *Science*, 353(6295):163–166.

[11] Bhagat, S., Cormode, G., and Muthukrishnan, S. (2011). Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. Springer.

[12] Bollacker, K., Cook, R., and Tufts, P. (2007). Freebase: A shared database of structured general human knowledge. In *Association for the Advancement of Artificial Intelligence*, volume 7, pages 1962–1963.

[13] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*, pages 2787–2795.

[14] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*.

[15] Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.

[16] Cai, L. and Ji, S. (2020). A multi-scale approach for graph link prediction. In *Association for the Advancement of Artificial Intelligence*, volume 34, pages 3308–3315.

[17] Cao, S., Lu, W., and Xu, Q. (2015). Grarep: Learning graph representations with global structural information. In *ACM International Conference on Information and Knowledge Management*, pages 891–900.

[18] Cao, Y., Wang, X., He, X., Hu, Z., and Chua, T.-S. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *International World Wide Web Conference*, pages 151–161.

[19] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Association for the Advancement of Artificial Intelligence*, pages 1306–1313.

[20] Cavallari, S., Zheng, V. W., Cai, H., Chang, K. C.-C., and Cambria, E. (2017). Learning community embedding with community detection and node embedding on graphs. In *Conference on Information and Knowledge Management*, pages 377–386.

[21] Chakraborty, T., Dalmia, A., Mukherjee, A., and Ganguly, N. (2017). Metrics for community analysis: A survey. *ACM Computing Surveys*, pages 1–37.

[22] Chang, H., Feng, Z., and Ren, Z. (2017). Community detection using dual-representation chemical reaction optimization. *IEEE Transactions on Cybernetics*, 47(12):4328–4341.

[23] Chen, J., Li, Z., and Huang, B. (2017). Linear spectral clustering superpixel. *IEEE Transactions on Image Processing*, 26(7):3317–3330.

[24] Chen, X., Chen, M., Shi, W., Sun, Y., and Zaniolo, C. (2019). Embedding uncertain knowledge graphs. In *Association for the Advancement of Artificial Intelligence*, volume 33, pages 3363–3370.

[25] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

[26] Chung, F. R. (1997). *Spectral graph theory*. Number 92. American Mathematical Soc.

[27] Clough, J. R., Gollings, J., Loach, T. V., and Evans, T. S. (2015). Transitive reduction of citation networks. *Journal of Complex Networks*, 3(2):189–203.

[28] Condon, A. and Karp, R. M. (2001). Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2):116–140.

[29] Cont, R. and Schaanning, E. (2019). Monitoring indirect contagion. *Journal of Banking & Finance*, 104:85–102.

[30] Dey, A. K., Gel, Y. R., and Poor, H. V. (2019). What network motifs tell us about resilience and reliability of complex networks. *Proceedings of the National Academy of Sciences*, 116(39):19368–19373.

[31] Dong, L., Wei, F., Zhou, M., and Xu, K. (2015). Question answering over freebase with multi-column convolutional neural networks. In *Association for Computational Linguistics*, pages 260–269.

[32] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

[33] Feng, R., Yang, Y., Hu, W., Wu, F., and Zhang, Y. (2018). Representation learning for scale-free networks. In *Association for the Advancement of Artificial Intelligence*, volume 32.

[34] Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305.

[35] Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408.

[36] Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75–174.

[37] Ge, Y., Ma, J., Zhang, L., and Lu, H. (2020). Unifying homophily and heterophily network transformation via motifs. *arXiv preprint arXiv:2012.11400*.

[38] Ge, Y., Peng, P., and Lu, H. (2021). Mixed-order spectral clustering for complex networks. *Pattern Recognition*, 117:107964.

[39] Ghoshdastidar, D. and Dukkipati, A. (2014). Consistency of spectral partitioning of uniform hypergraphs under planted partition model. In *Neural Information Processing Systems*, pages 397–405.

[40] Ghoshdastidar, D. and Dukkipati, A. (2017). Uniform hypergraph partitioning: Provable tensor methods and sampling techniques. *Journal of Machine Learning Research*, 18(50):1–41.

[41] Girardi, G. and Ergün, A. T. (2013). Systemic risk measurement: Multivariate garch estimation of covar. *Journal of Banking & Finance*, 37(8):3169–3180.

[42] Gleich, D. F., Lim, L.-H., and Yu, Y. (2015). Multilinear pagerank. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1507–1541.

[43] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[44] Goyal, P. and Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94.

[45] Granovetter, M. S. (1977). The strength of weak ties. In *Social Networks*, pages 347–367. Elsevier.

[46] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 855–864.

[47] Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

[48] Hasan, I., Politsidis, P. N., and Sharma, Z. (2021). Global syndicated lending during the covid-19 pandemic. *Journal of Banking & Finance*, pages 106–121.

[49] Heindorf, S., Potthast, M., Stein, B., and Engels, G. (2016). Vandalism detection in wikidata. In *International Conference on Information and Knowledge Management*, pages 327–336.

[50] Jamali, M. and Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142.

[51] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

[52] Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Association for Computational Linguistics*, pages 687–696.

[53] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

[54] Jia, S., Xiang, Y., Chen, X., and Wang, K. (2019). Triple trustworthiness measurement for knowledge graph. In *International World Wide Web Conference*, pages 2865–2871.

[55] Kim, Y., Do, H., and Kim, S. B. (2020). Outer-points shaver: Robust graph-based clustering via node cutting. *Pattern Recognition*, 97:107001.

[56] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[57] Klicpera, J., Weißenberger, S., and Günnemann, S. (2019). Diffusion improves graph learning. In *Neural Information Processing Systems*.

[58] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[59] Kossinets, G. and Watts, D. J. (2006). Empirical analysis of an evolving social network. *science*, 311(5757):88–90.

[60] Lee, J. R., Gharan, S. O., and Trevisan, L. (2014). Multiway spectral partitioning and higher-order cheeger inequalities. *Annual Symposium on Theory of Computing*, 61(6):1–30.

[61] Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010a). Signed networks in social media. In *Special Interest Group on Computer–Human Interaction*, pages 1361–1370.

[62] Leskovec, J., Lang, K. J., and Mahoney, M. (2010b). Empirical comparison of algorithms for network community detection. In *ACM International World Wide Web Conference*, pages 631–640.

[63] Li, P. and Milenkovic, O. (2017). Inhomogoenous hypergraph clustering with applications. In *Neural Information Processing Systems*, pages 2305–2315.

[64] Li, X., Taheri, A., Tu, L., and Gimpel, K. (2016). Commonsense knowledge base completion. In *Association for Computational Linguistics*, pages 1445–1455.

[65] Liang, J., Xiao, Y., Zhang, Y., Hwang, S.-w., and Wang, H. (2017). Graph-based wrong isa relation detection in a large-scale lexical taxonomy. In *Association for the Advancement ofArtificial Intelligence*, volume 31.

[66] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Association for the Advancement of Artificial Intelligence*, volume 29.

[67] Liu, M.-Y., Tuzel, O., Ramalingam, S., and Chellappa, R. (2011). Entropy rate superpixel segmentation. In *IEEE Computer Vision and Pattern Recognition*, pages 2097–2104.

[68] Lukovnikov, D., Fischer, A., Lehmann, J., and Auer, S. (2017). Neural network-based question answering over knowledge graphs on word and character level. In *International Conference on World Wide Web*, pages 1211–1220.

[69] Lusseau, D. (2003). The emergent properties of a dolphin social network. *Proceedings of the Royal Society of London B: Biological Sciences*, 270(Suppl 2):S186–S188.

[70] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297.

[71] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444.

[72] Meila, M. and Shi, J. (2001). Learning segmentation by random walks. In *Neural Information Processing Systems*, pages 873–879.

[73] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *Workshop in International Conference on Learning Representations*.

[74] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*, pages 3111–3119.

[75] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., and Alon, U. (2004). Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542.

[76] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827.

[77] Muñoz, J. V., Gonçalves, M. A., Dias, Z., and Torres, R. d. S. (2019). Hierarchical clustering-based graphs for large scale approximate nearest neighbor search. *Pattern Recognition*, 96:106970.

[78] Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

[79] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems*, pages 849–856.

[80] Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *International Conference on Knowledge Discovery and Data Mining*, pages 1105–1114.

[81] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[82] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *International Conference on Knowledge Discovery and Data Mining*, pages 701–710.

[83] Petersen, S. E. and Sporns, O. (2015). Brain networks and cognitive architectures. *Neuron*, 88(1):207–219.

[84] Piao, G. and Breslin, J. G. (2018). Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model. In *Extended Semantic Web Conference*, pages 496–511. Springer.

[85] Reihanian, A., Feizi-Derakhshi, M.-R., and Aghdasi, H. S. (2018). Overlapping community detection in rating-based social networks through analyzing topics, ratings and links. *Pattern Recognition*, 81:370–387.

[86] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. In *Conference on Uncertainty in Artificial Intelligence*. Citeseer.

[87] Ribeiro, L. F., Saverese, P. H., and Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 385–394.

[88] Rossi, R. A., Jin, D., Kim, S., Ahmed, N. K., Koutra, D., and Lee, J. B. (2019). From community to role-based graph embeddings. *arXiv preprint arXiv:1908.08572*.

[89] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[90] Rozemberczki, B., Allen, C., and Sarkar, R. (2021). Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014.

[91] Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1):27–64.

[92] Serrour, B., Arenas, A., and Gómez, S. (2011). Detecting communities of triangles in complex networks using spectral optimization. *Computer Communications*, 34(5):629–634.

[93] Shen, J., Du, Y., Wang, W., and Li, X. (2014). Lazy random walks for superpixel segmentation. *IEEE Transactions on Image Processing*, 23(4):1451–1462.

[94] Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. (2002). Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31(1):64–68.

[95] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

[96] Silva, T. C., Souza, S. R. S., and Tabak, B. M. (2017). Monitoring vulnerability and impact diffusion in financial networks. *Journal of Economic Dynamics and Control*, 76:109–135.

[97] Slota, G. M. and Madduri, K. (2014). Complex network analysis using parallel approximate motif counting. In *IEEE International Parallel and Distributed Processing Symposium*, pages 405–414.

[98] Sporns, O. and Kötter, R. (2004). Motifs in brain networks. *PLoS Biology*, 2(11):e369.

[99] Stutz, D., Hermans, A., and Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27.

[100] Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. (2018). Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

[101] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *International World Wide Web Conference*, pages 1067–1077.

[102] Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

[103] Tijms, H. (2012). *Understanding probability*. Cambridge University Press.

[104] Traud, A. L., Mucha, P. J., and Porter, M. A. (2012). Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180.

[105] Tsourakakis, C. E., Pachocki, J., and Mitzenmacher, M. (2017). Scalable motif-aware graph clustering. In *International World Wide Web Conference*, pages 1451–1460. ACM.

[106] Tung, F., Wong, A., and Clausi, D. A. (2010). Enabling scalable spectral clustering for image segmentation. *Pattern Recognition*, 43(12):4069–4076.

[107] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations*.

[108] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

[109] Wagner, D. and Wagner, F. (1993). Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer.

[110] Wang, D., Cui, P., and Zhu, W. (2016). Structural deep network embedding. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234.

[111] Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., and Wang, Z. (2019a). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *International Conference on Knowledge Discovery and Data Mining*, pages 968–977.

[112] Wang, H., Zhao, M., Xie, X., Li, W., and Guo, M. (2019b). Knowledge graph convolutional networks for recommender systems. In *International World Wide Web Conference*, pages 3307–3313.

[113] Wang, J., Wang, Z., Zhang, D., and Yan, J. (2017a). Combining knowledge with deep convolutional neural networks for short text classification. In *International Joint Conference on Artificial Intelligence*, pages 2915–2921.

[114] Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. (2017b). Community preserving network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

[115] Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Association for the Advancement of Artificial Intelligence*, volume 14, pages 1112–1119.

[116] Wasserman, S., Faust, K., et al. (1994). Social network analysis: Methods and applications.

[117] Wu, G. and Kang, W. (2017). Exploiting superpixel and hybrid hash for kernel-based visual tracking. *Pattern Recognition*, 68:175–190.

[118] Wuchty, S., Oltvai, Z. N., and Barabási, A.-L. (2003). Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nature Genetics*, 35(2):176–179.

[119] Xie, R., Liu, Z., Lin, F., and Lin, L. (2018). Does william shakespeare really write hamlet? knowledge representation learning with confidence. In *Association for the Advancement of Artificial Intelligence*, volume 32.

[120] Xu, J., Wickramarathne, T. L., and Chawla, N. V. (2016). Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028.

[121] Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *International conference on Knowledge Discovery and Data Mining*, pages 1365–1374.

[122] Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. (2015). Network representation learning with rich text information. In *International Joint Conference on Artificial Intelligence*, volume 2015, pages 2111–2117.

[123] Yang, C., Sun, M., Liu, Z., and Tu, C. (2017). Fast network embedding enhancement via high order proximity approximation. In *International Joint Conference on Artificial Intelligence*, pages 3894–3900.

[124] Yang, J. and Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, pages 181–213.

[125] Yaveroğlu, Ö. N., Malod-Dognin, N., Davis, D., Levnajic, Z., Janjic, V., Karapandza, R., Stojmirovic, A., and Pržulj, N. (2014). Revealing the hidden language of complex networks. *Scientific Reports*, 4(1):1–9.

[126] Yin, H., Benson, A. R., Leskovec, J., and Gleich, D. F. (2017). Local higher-order graph clustering. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 555–564.

[127] Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473.

[128] Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2018a). Network representation learning: A survey. *IEEE Transactions on Big Data*, 6(1):3–28.

[129] Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In *International Conference on Knowledge Discovery and Data Mining*, pages 353–362.

[130] Zhang, L., Ge, Y., and Lu, H. (2020). Hop-hop relation-aware graph neural networks. *arXiv preprint arXiv:2012.11147*.

[131] Zhang, L. and Lu, H. (2020). A feature-importance-aware and robust aggregator for gcn. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

[132] Zhang, L., Song, H., and Lu, H. (2018b). Graph node-feature convolution for representation learning. *arXiv preprint arXiv:1812.00086*.

[133] Zhang, Y., Ai, Q., Chen, X., and Wang, P. (2018c). Learning over knowledge-base embeddings for recommendation. *ArXiv*, abs/1803.06540.

[134] Zhang, Z., Cui, P., Li, H., Wang, X., and Zhu, W. (2018d). Billion-scale network embedding with iterative random projection. In *IEEE International Conference on Data Mining*, pages 787–796.

[135] Zhang, Z., Cui, P., Wang, X., Pei, J., Yao, X., and Zhu, W. (2018e). Arbitrary-order proximity preserved network embedding. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 2778–2786.

[136] Zhou, D., Zhang, S., Yildirim, M. Y., Alcorn, S., Tong, H., Davulcu, H., and He, J. (2017). A local algorithm for structure-preserving graph cut. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 655–664.