ORIGINAL ARTICLE

Improving graph collaborative filtering with network motifs

Yuqi Zhang[1] · Jian Yu[1] · Zhizhong Liu[2] · Guiling Wang[3] · Minh Nguyen[1] · Quan Z. Sheng[4] · Nancy Wang[1]

Received: 10 June 2024 / Accepted: 7 February 2025

Abstract
Deep learning on graphs, specifically graph convolutional networks (GCNs), has exhibited exceptional efficacy in the domain of recommender systems. Most GCNs have a message-passing architecture that enables nodes to aggregate information from neighbours iteratively through multiple layers. This enables GCNs to learn from higher-order information, but the model does not allow for direct captions of the local structural patterns. Our rationale is to investigate the effectiveness of capturing such local patterns for graph-based collaborative filtering to enhance model's learning ability per layer. This technique combines lower-order and higher-order interactions during layer-wise propagation. In this paper, we propose MotifGCN to aggregate both lower-order and higher-order information in each graph convolution layer. Specifically, we develop dedicated algorithms of generating motif adjacency matrices. The matrices are then used for motif-enhanced neighbourhood aggregation in each layer. As this paper focuses on recommender systems, MotifGCN is built on the basis of bipartite graphs. Our experiments on four real-world datasets show that MotifGCN has a superior performance compared to various state-of-the-art methods.

Keywords Recommender systems · Collaborative filtering · Graph convolutional networks · Network motifs

1 Introduction

Graph neural networks (GNNs) have emerged as a promising solution for multiple industries such as e-Commerce [1–4] and social media [5–7] in favour of their superior performance in graph learning tasks including node classification, graph classification, and link prediction. A series of recent advancements in recommender systems [6, 8, 9] that adopt graph convolutional network (GCN) [10] show great potential of graph-based collaborative filtering (CF). Graph-based CF represents users and items as nodes and interactions as edges within a graph structure, which allows for the exploitation of higher-order connections. However, there are two key challenges for graph-based CF methods. One is the data sparsity issue in user–item interactions, which makes it more challenging to generate reliable recommendations. The other challenge is the lack of local pattern exploitation in the existing graph-based CF methods. Recent studies [11–14] have further explored network structures, underscoring the importance of local patterns and their impact on network-based tasks, which provides additional context and motivation for our work.

Although GNNs can learn from higher-order connections, it relies on multi-layer propagation. Local structural patterns cannot be captured directly by normal graph convolution. Therefore, our rationale is to investigate the effectiveness of capturing such local patterns for graph-based collaborative filtering to enhance the model's

https://doi.org/10.1007/s00521-025-11079-8

Published online: 24 February 2025

learning ability at each layer. This achieves similar effects to additionally preserving intermediate layer embeddings into the output layer as lower-order and higher-order interactions are both captured in the final embeddings [15].

As a simple and common higher-order patterns in graphs, network motifs have shown promising enhancement to graph representation learning on homogeneous graphs such as social networks [1, 16–18] and bioinformatics networks [19, 20], as well as heterogeneous networks [21–24]. Formally, we define network motifs as recurring patterns or subgraphs that appear in graph networks more frequently than in randomised graphs [25]. They can be considered the fundamental building blocks of graphs, capturing local structures that are often indicative of underlying node behaviours and characteristics. If multiple nodes can form a motif instance, they are considered as motif neighbours and are connected in the motif adjacency matrix, even if some of the nodes are not really connected in the graph. This means motif adjacency matrices can be used to preserve higher-order neighbours'information in a single layer's graph convolution operation.

By incorporating motifs in layer-wise propagation, nodes learn the local structure so more information can be captured than using the original adjacency matrix. In addition, network motifs can mitigate the limitation of user–item interaction data sparsity by utilising local connectivity patterns to infer missing links, which then enriches the data for recommendations. By identifying and applying motifs to a GNN framework, we can gain deeper insights into the structural properties of any given user–item interaction graphs, which can lead to more accurate and robust recommendation outcomes. Based on the rationales discussed above, we propose a new graph neural network that preserves higher-order information in each single-layer neighbourhood aggregation using network motifs.

In our previous work [26], we proposed a motif-based graph attention network for web service recommendation (MGSR). The model uses a motif attention mechanism that first calculates graph attentions on multiple motif adjacency matrices and then combine the outputs using a motif attention which measures the similarity between previous layer's hidden states and the outputs from the motif adjacency matrices. This means the motif attention mechanism with M motifs will have a significantly larger computational complexity than the original graph attention network [27], which is $O\left(M \cdot K \cdot (|V| \cdot d \cdot d' + |E| \cdot d')\right)$ where M is the number of motifs, K is the number of attention heads, d is the input feature dimension, and d' is the output feature dimension. Additionally, due to the nature of motif adjacency matrix, the matrix has a higher density than the original adjacency matrix so E is significantly larger than the actual number of edges.

While MGSR is suitable for relatively small datasets like a mashup-API dataset in service computing, it is challenging to scale to large datasets used in general recommender systems, such as the Amazon datasets. To overcome this limitation and explore the impact of motifs in general recommender systems, we propose a series of efficient motif adjacency matrix generation algorithms and a generic framework for recommender systems called MotifGCN.

MotifGCN integrates motifs in graph convolution layers to enhance the neighbourhood aggregation efficiency within each single layer. Compared to MGSR, MotifGCN does not use any form of attention mechanism and has fewer learnable parameters hence has a significantly improved computational complexity of $O\left(|V| \cdot d\right)$. Our experiments on four real-world recommendation datasets show that motifs can significantly improve GCNs for CF.

In this paper, we propose one main research question along with two sub-research questions.

Main RQ: How can the incorporation of network motifs enhance graph-based collaborative filtering methods?

Sub-RQ1: How does the proposed motif adjacency matrix generation algorithm compare to general approaches in terms of computational efficiency?

Sub-RQ2: How does the proposed MotifGCN perform on different datasets compared to basic graph CF methods and other higher-order-enhanced graph CF methods?

Our contributions are summarised as follows:

- We propose a novel CF framework called MotifGCN, which integrates motifs in graph convolution layers to enhance the neighbourhood aggregation efficiency within each single layer. This generic framework can be applied to most recommendation datasets.
- We propose dedicated algorithms of generating motif adjacency matrices for all motifs consisting of up to four nodes in bipartite graphs. Theoretical analyses and experiments show that the algorithms have improved time and space complexity over general approaches.
- We conduct experiments on four real-world datasets to demonstrate the effectiveness of our proposed framework.

The rest of paper is organised as follows: In Sect. 2, we briefly review and discuss some recent graph-based CF approaches for recommender systems and motif-based graph neural networks. Section 3 provides the background knowledge to understand our proposed method. Section 4 presents our proposed motif adjacency matrices generation algorithms and MotifGCN architecture. Section 5 shows our experimental results and performance analysis. Finally, Sect. 7 concludes this paper.

2 Related work

In this paper, our work is dedicated to motif-based GNN for CF. To the best of our knowledge, we are the first to explore the application of network motifs specifically on bipartite graphs within this context. For this section, we extend the scope slightly and review current literature in the fields of GNNs for CF, motif-based GNNs, and higher-order enhanced GNNs for CF in Sections 2.1, 2.2, and 2.3, respectively.

2.1 Graph neural networks for collaborative filtering

There has been an extensive collection of research works on GNNs for CF in recent couple of years. Following the earlier works of GraphSAGE [28] and GCN [10], neural graph collaborative filtering (NGCF) [8] is one of the earliest works to apply GCN to CF. The paper proposes a message-passing design for convolution layers with a short circuit mechanism that concatenates the hidden states after each layer as the final output. The datasets used are represented as bipartite graphs so edges only exist between user–item pairs. The message between a user-item pair is the weighted average of the item node embedding and the Hadamard product of the two nodes' embeddings. The convolution layers then aggregate the messages for all nodes to update their embeddings. After obtaining the final node embeddings, the prediction of the interaction between a user–item pair is calculated via the inner product of their embeddings.

Influenced by NGCF, He et al. [6] perform ablation studies on NGCF and show that feature transformation matrix and nonlinear activation function do not have additional effectiveness for GCNs on CF tasks. In the same work, an NGCF-inspired model called LightGCN is proposed to eliminate the feature transformation matrix and nonlinear activation function in the convolution layers. In a way, LightGCN can be considered a reduced and refined version of the NGCF model while further enhancing experimental performances.

As another key publication, Yu et al. [3] propose low-pass collaborative filter network (LCFN) that simplifies spectral graph convolution by calculating only a small number of graph Laplacian eigenvectors of small eigenvalues. The work shows that small eigenvalues correspond to low-frequency graph signals which can reflect nodes being connected, while large eigenvalues correspond to high-frequency graph signals which represent noises in data. Through removing high-frequency eigenvectors in graph Laplacian, LCFN avoids high computational cost of eigen-decomposition and eliminates noise in graphs, thereby improves model accuracy and efficiency.

LCFN has a limitation of constructing separate graphs for user nodes and item nodes thus ignores the direct interaction between a user and an item. To address this issue, low-pass graph convolutional network (LGCN) [9]

uses bipartite graphs to preserve the user–item interactions. Similar to LightGCN, LGCN also simplifies the graph convolution layer by removing feature transformation and nonlinear activation. The authors also show that low-pass filter enables LGCN to avoid over-smoothing issue.

Ouyang et al. [29] identify that traditional CF models tend to overfit due to noise in labelled training data and propose to mitigate this issue by integrating contrastive learning with a GNN to enhance the alignment and uniformity of user and item representations independently of labels. Alignment is improved by minimising the distance between representations of interacting user–item pairs. Uniformity is enhanced by distributing these representations more evenly across their respective hyperspheres. Their framework also leverages a novel 0-layer embedding perturbation mechanism for minimal data augmentation, allowing the model to learn these properties in a label-independent manner with faster convergence.

2.2 Motif-based graph neural networks

A series of recent works show that the topological structure of graphs has significant influence on local and global representation learning [30, 31]. As one way of exploiting the topological information in graphs, network motifs have been used in a range of research works for GNNs. In this section, a range of motif-based GNNs will be explained.

As one of the first works using motifs in GNNs, MotifNet [16] uses motifs for directed graphs. The work obtains motif adjacency matrices from directed motifs and then get the motif Laplacian. Although the adjacency matrices are still symmetric, different motifs induce different structures which enables the model to be anisotropic. For the core structure of MotifNet, multivariate polynomials of degree p are applied to the motif Laplacians. Due to the large number of coefficients required, the polynomials are simplified using two approaches. One is to only consider the two motifs of incoming and outgoing edges. The other is to define the polynomials with recursive products of each degree.

Rossi et al. [32] propose a framework that uses motifs to learn higher-order network embeddings which represent interactions between multi-hop neighbours. They adopt the same definition of motif adjacency matrices as in [33] except that they use undirected motifs. Additionally, the authors propose k-step motif adjacency matrix which is the power of the original motif adjacency matrix. For m motifs with k steps, there are $m \times k$ motif adjacency matrices in total. This embedding method is adopted in motif convolutional networks [34] which uses an attention mechanism and an epsilon greedy strategy to select only one neighbour for each node's neighbourhood aggregation.

With a focus on heterogeneous graphs, motif-CNN [35] also uses motifs to capture higher-order information. The authors differentiate nodes into different semantic roles according to their types and occurrences in the motifs. Instead of the adjacency matrix in GCN, motif-CNN uses a motif adjacency matrix weighted by semantic roles to perform convolution operation. Nodes with the same semantic role in all instances of a motif share the same weight. For the weights of different motifs, Motif-CNN adopts the attention mechanism to dynamically adjust the importance of different motifs for each node. Similarly, InfoMotif [36] also captures multi-hop interactions using structural roles of nodes in motifs. Nodes are considered as the same attributed role if they have co-varying attributes in similar motif instances. To identify structurally similar nodes, InfoMotif adopts mutual information maximisation to obtain attribute correlations in motifs.

Li et al. [37] propose to incorporate global information with motif-based subgraph convolutional neural networks. Instead of directly linked neighbours, motif neighbours are used for neighbourhood aggregation. An input graph is transformed to an m-ary tree by cascading each node's m-ary subtree following the order of node degrees. The m-ary tree is then used to generate an adjacency matrix for feature aggregation.

The recent works of motif-based multiview graph attention networks with gated fusion [1] and motif graph neural network [20] have rather similar structures. They both follow the commonly used message-passing paradigm of GNN framework. Specifically, they first generate motif adjacency matrices and then aggregate node features on each motif adjacency matrix. The key difference between the two pieces of work is how the features

aggregated from different motif adjacency matrices are combined. The former uses an attention layer to get a weighted average from the feature matrices before cascading a gated fusion layer to control the weights of residual connections. The latter minimises redundant information among all motifs. Here, the redundant information is measured by the similarity between node features aggregated from each motif adjacency matrix and those from all other motif adjacency matrices. An inner product and a non-linear activation function are used in the similarity calculation.

Motifs are also used in graph classification. Zhang et al. [38] present a novel framework that uses motifs in GNNs pre-training for molecular property prediction. The framework learns subgraph embeddings by maximising the likelihood of each subgraph belonging to a motif, and then optimise the embeddings by combining them to contrast with initial who-graph embeddings. Similarly, Zhou et al. [39] also use motifs to learn subgraph embeddings for graph classification. Differently, they rank subgraph importance by calculating the similarity between subgraph embeddings and a graph-level vector and combine important subgraphs into the overall graph embedding. In the subgraph-motif alignment phase, top candidate subgraphs are selected based on importance scores, and motifs are iteratively extracted using k-means clustering. These motifs guide the alignment of subgraph representations through a contrastive loss function, ensuring that subgraphs align with motifs of their class while remaining distinct from others.

Despite the rich collection of motif applications in GNNs, there is currently very limited research in motif-based CF.

2.3 Higher-order enhanced graph neural networks for collaborative filtering

Although there has been considerable amount of research on motif-based GNNs, applications of motif on CF task are still limited. Here we review some CF models that consider higher-order structure in their architecture designs, which is also related to our work as motif is a tool for capturing higher-order connectivity as well.

HyperRec [40] is a recommendation model that employs hypergraphs to represent higher-order relationships between users and items. The hypergraph is constructed such that each hyperedge represents a group of items that a user has interacted with within a specific time window, thereby encapsulating the sequential and co-occurrence information in the user's interaction history. By utilising sequential hypergraphs, the model captures complex dependencies between short-term user intent and dynamic item embeddings, offering a richer and more comprehensive user representation compared to traditional user-item graphs.

Ji et al. [41] propose a model called dual channel hypergraph collaborative filtering which also leverages hypergraphs to capture higher-order relationships among users and items, but form two hypergraphs for user–user and item–item, respectively. The hyperedge in user–user hypergraph is a group of users who share common interactions with one or more items. Similarly, the hyperedge in item–item hypergraph is a group of items that are connected with at least one same user. Through the two hypergraphs, the model can learn complex higher-order dependencies within users and items.

Zhang and McAuley [42] introduce a model that mixes multiple order's higher-order graph convolutions within a stacked structure to capture a broader range of user–item interactions. This hybrid approach is an adaptation of the shortcut technique [15, 43] as it essentially concatenates interim representations to generate the final embeddings. Similarly, Nguyen et al. [44] incorporate higher-order connections within autoencoder framework by augmenting the interaction graph with second-order and fourth-order interactions.

Yu et al. [45] utilise multi-channel hypergraphs with self-supervised learning strategy. The channels are user views generated from different aspects such as social relationships and common purchases. Auxiliary tasks are then generated from these channels for self-supervised learning. Hypergraph Contrastive Collaborative Filtering [46] also combine hypergraphs and self-supervised learning for CF. Specifically, hypergraph is constructed based on mutual connections for both users and items. Both the hypergraph and original interaction graph are used to generate contrastive views by edge dropout. The contrastive loss for self-supervised learning maximises the agreement between same nodes in different views and minimises the agreement between different nodes. This

objective helps the model to refine the embeddings, ensuring that similar nodes have closer representations, while dissimilar ones are pushed apart.

Xia et al. [47] introduce a graph signal processing approach that uses two types of filters to separately learn user unique behaviours and global behaviours. The user unique behaviours are learned among users that share similar preferences, preventing user-specific preferences from being influenced by the behaviours of dissimilar users. The global behaviours are general interaction patterns across all users, which are captured by a combination of higher-order and ideal low-pass filters. This balances the need to leverage broad user interaction data while avoiding the over-smoothing problem commonly associated with traditional filtering methods.

Gong et al. [48] design two higher-order neighbour-enhanced strategies to improve GNNs for CF. The embedding propagation (EP)-based strategy and the positive user–item pair (PP)-based strategy. The EP-based strategy employs a novel higher-order graph convolution method that mitigates the excessive decay of higher-order neighbour information by assigning different weight coefficients to various order neighbours, thereby capturing richer collaborative signals. The PP-based strategy enhances the model's training process by identifying potentially positive samples from a user's higher-order neighbours, based on item similarity and user similarity, to improve the quality of the positive user–item pairs used during model training.

3 Background

In this section, we first introduce network motifs and explain the differences between motifs in homogeneous graphs and bipartite graphs. We then discuss the difference and relationship between motif adjacency matrices and powers of adjacency matrix.

3.1 Network motifs in bipartite graphs

Network motifs are recurring and significant patterns or subgraphs that appear in complex networks more frequently than in randomised networks [25]. As most motifs involve more than two nodes, they can provide useful local structure information and also contribute to global properties of the whole graph. Depending on the type of graphs, motifs can be either directed or undirected. Figure 1 shows some common motifs in directed and undirected graphs. Our proposed method uses bipartite graphs to represent user–item interactions. This is a natural choice for collaborative filtering as bipartite graphs can effectively represent the relationships between the two distinct user set and item set.

We provide the formal definition of bipartite graphs as follows.

Definition 1 (Bipartite graphs [49]) A bipartite graph $G = (U, V, E)$ is a graph where all nodes in G are partitioned into two disjoint classes U and V, i.e. $U \cap V = \emptyset$, such that every edge has its terminal node in different classes from its initial node: nodes in the same partition class must not be adjacent.

The interactions between users and items are usually not symmetric in recommender systems, e.g. in movie rating a rating is established by a user for a movie but a movie cannot establish an interaction with a user. This

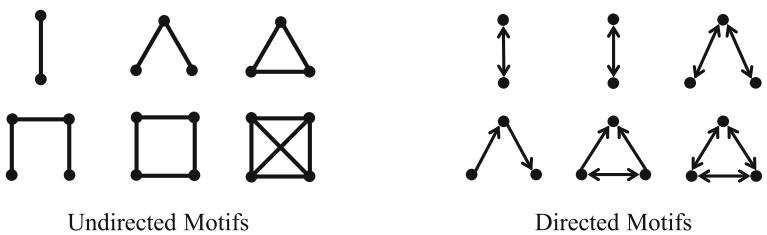

Fig. 1 Examples of undirected motifs and directed motifs.

means that there does not exist a symmetric pair of directed edges between any two nodes. To define graphs for recommender systems, we need to define oriented graphs first.

Definition 2 (Oriented graphs [50]) An oriented graph $G = (V, E)$ is a directed graph such that none of its node pairs is linked by more than one edge and none of its edges has the same initial and terminal node.

Now, we can define oriented bipartite graphs, which can be used to represent recommendation data.

Definition 3 (Oriented bipartite graphs [51]) An oriented bipartite graph $G = (U, V, E)$ is a directed bipartite graph such that for any $u \in U, v \in V, uv \in E$ implies $vu \notin E$.

We can use oriented bipartite graphs to represent recommender systems datasets but finding instances of directed motifs in oriented bipartite graphs is in fact equivalent to identifying undirected motifs in undirected bipartite graphs. The generated motif adjacency matrices are the same because they only record the coexistence of nodes in motif instances. An example of this equivalence is shown in Fig. 2. For the ease of understanding and implementation, we consider the graphs and motifs as undirected in this paper. If the datasets are represented as directed graphs, our methods are completely compatible and no further modification is needed.

Due to the absence of user–user edges and item–item edges, the complex types of motifs in homogeneous graphs such as four-clique do not exist in bipartite graphs. This makes bipartite graphs not as computationally expensive as homogeneous graphs for motif analysis. For the purpose of exploring different motifs within limited computational resources, we only consider motifs consisting of up to four nodes. Figure 3 shows all seven motifs used in this work.

3.2 Motif adjacency matrix

A common approach to incorporate network motifs into existing graph learning frameworks is to form motif adjacency matrices [33]. Given a graph and a certain motif m, we can form the motif adjacency matrix $\mathbf{A}_m$ whose elements (i, j) are the number of instances that nodes i and j co-exist in. We refer to nodes that co-exist in at least one motif instance as motif neighbours. A common approach to generate a motif adjacency matrix is to carry out a subgraph isomorphism test between the motif and every subgraph in the graph. An example is the VF2 [52] graph matching algorithm, which has a complexity of $O(N!N)$ where N is the number of nodes in the graph. To improve the computational efficiency, we introduce our dedicated algorithm for generating motif adjacency matrices in Sect. 3.1.

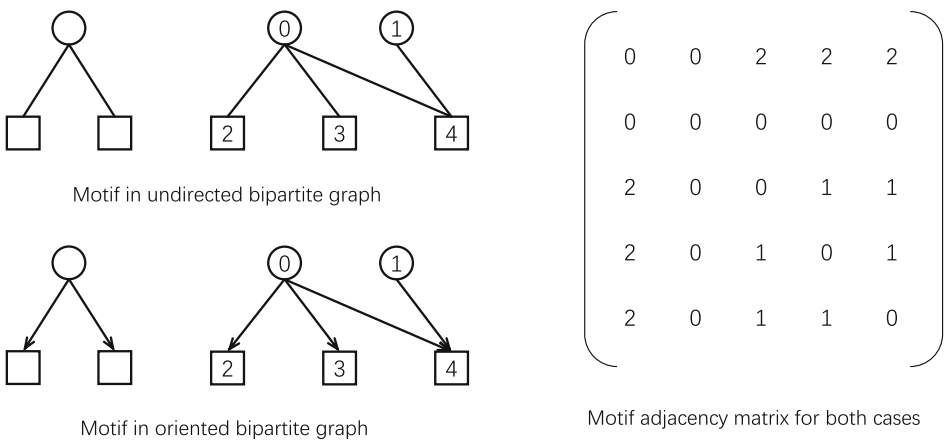

Fig. 2 Example motif in undirected bipartite graph and oriented bipartite graph. Circle nodes and square nodes are used to represent two types of nodes, e.g. users and items.

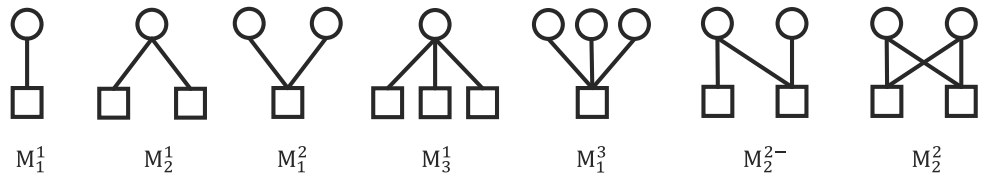

Fig. 3 All motifs consisting of two to four nodes in bipartite graphs. Superscript refers to the number of user nodes in the motif while subscript means the number of item nodes in the motif. The hyphen in M_2^{2-} indicates one less edge in the motif compared to M_2^2.

4 Method

In this section, we first present the algorithms for generating motif adjacency matrices for motifs in bipartite graphs and then introduce the architecture of our proposed motif graph convolutional network.

4.1 Generating motif adjacency matrices for bipartite graphs

Previously, Bouritsas et al. [31] have tried counting motifs using the generic subgraph isomorphism algorithm VF2 [52]. The method is usable for small graphs such as the Cora dataset but cannot cope with large graphs such as the Amazon-Book dataset due to a high complexity of $O(N!N)$. As most datasets for recommender systems have a very low density, we decide to design tailored algorithms to generate motif adjacency matrices for each motif. As shown in Fig. 3, motif M_1^1 is simply an edge between two nodes, so the motif adjacency matrix is the same as the original adjacency matrix and we follow the generation method used in GCN [10]. In our previous work [26], we briefly described the algorithms for M_2^1 to M_2^2 and demonstrate the details for motif M_2^1. Here, we omit the algorithm for M_2^1 and provide steps to alter the algorithm for M_1^2 to M_1^3: (1) For M_1^2, simply replace all appearances of U with V. (2) For M_3^1, replace $(j,k) \in \binom{|\mathcal{N}(i)|}{2}$ at line 6 with $(j,k,l) \in \binom{|\mathcal{N}(i)|}{3}$ and follow the same steps of line 7 for (j, l) and (k, l); (3) For M_1^3, refer to both (1) and (2).

The algorithms for motif M_2^{2-} and M_2^2 are shown as follows. Algorithm 1 details the method for constructing the motif adjacency matrix for motif M_2^{2-}. The process begins by grouping all edges according to their associated item nodes, and an empty matrix $\mathbf{A}_m$ is initialised. In each iteration of the outer loop, with j and k representing any two item nodes, the algorithm calculates both the intersection S and the symmetric difference P of the neighbours of j and k. The sizes of S and P are denoted by n_s and n_p , respectively. If S has at least one node and P has at least one node, the algorithm proceeds to update four types of counts: 1) The count between j and k is increased by$n_s \times n_p$, since a motif instance requires one mutual neighbour and one non-mutual neighbour of j and k. 2) For each mutual neighbour i, the counts between i, j and i, k are increased by n_p, reflecting the number of ways to choose one node from P. 3) For each $i \in S$ and $l \in P$, the count between i and l is incremented by one, as i, l, j, and k can form only one motif instance. 4) For each node $l \in P$, the counts between l, j and l, k are increased by n_s, corresponding to the number of ways to select one node from S. The algorithm concludes by adding the transpose of $\mathbf{A}_m$ to itself to produce a symmetric matrix.

Algorithm 2 outlines the procedure for generating the motif adjacency matrix for motif M_2^2. Initially, all edges are grouped based on common item nodes, and an empty matrix $\mathbf{A}_m$ is established. In each iteration of the outer loop, where j and k represent any two item nodes, the algorithm identifies the intersection S of the neighbours of j and k. If S contains at least two nodes, three types of counts are updated: 1) The count between j and k is increased by $\frac{n(n-1)}{2}$, since forming an instance of M_2^2 requires selecting any two of the n users in S, which has $\binom{n}{2}$ possibilities. 2) For each user node i, the counts between i, j and i, k are increased by $n-1$, as there are $n-1$ options for choosing the final user node to form an M_2^2 instance with i,j, and k. 3) The count between any two

user nodes in S is incremented by one, since any two user nodes can only appear together in one M_2^2 instance with j and k. Finally, to ensure symmetry, the transpose of matrix $\mathbf{A}_m$ is added to itself.

Algorithm 1 Motif adjacency matrix generation algorithm for M_2^{2-}.

Input: Edges E; number of nodes N; neighbourhood function $\mathcal{N}$; intersection function $INTERSECT$; symmetric difference function $SYMDIFF$
Output: Motif adjacency matrix $\mathbf{A}_m$

$\mathbf{A}_m \leftarrow N \times N$ matrix filled with 0;
for $\forall (j,k) \in V$ **do**
  $S \leftarrow INTERSECT(\mathcal{N}(j), \mathcal{N}(k))$;
  $P \leftarrow SYMDIFF(\mathcal{N}(j), \mathcal{N}(k))$;
  $n_s \leftarrow |S|$;
  $n_p \leftarrow |P|$;
  if $n_s > 0$ *and* $n_p > 0$ **then**
    Add $n_s \times n_p$ to element (j,k) in $\mathbf{A}_m$;
    for $\forall i \in S$ **do**
      Add n_p to elements (i,j) and (i,k) in $\mathbf{A}_m$;
      for $\forall l \in P$ **do**
        Add 1 to element (i,l);
      end
    end
    for $\forall l \in P$ **do**
      Add n_s to elements (l,j) and (l,k) in $\mathbf{A}_m$;
    end
  end
end
$\mathbf{A}_m \leftarrow \mathbf{A}_m + \mathbf{A}_m^T$

Algorithm 2 Motif adjacency matrix generation algorithm for M_2^2.

Input: Edges E; number of nodes N; neighbourhood function $\mathcal{N}$; intersection function $INTERSECT$
Output: Motif adjacency matrix $\mathbf{A}_m$

$\mathbf{A}_m \leftarrow N \times N$ matrix filled with 0;
for $\forall (j,k) \in \binom{|V|}{2}$ *pairs of items* **do**
  $S \leftarrow INTERSECT(\mathcal{N}(j), \mathcal{N}(k))$;
  $n_s \leftarrow |S|$;
  if $n_s > 1$ **then**
    Add $\frac{n_s(n_s-1)}{2}$ to element (j,k) in $\mathbf{A}_m$;
    for $\forall i \in S$ **do**
      Add $n_s - 1$ to elements (i,j) and (i,k) in $\mathbf{A}_m$;
    end
    for $\forall (i,l) \in \binom{n_s}{2}$ *pairs of users* **do**
      Add one to element (i,l) in $\mathbf{A}_m$;
    end
  end
end
$\mathbf{A}_m \leftarrow \mathbf{A}_m + \mathbf{A}_m^T$

To help better understand the advantage of our proposed algorithms, we present the average time complexity for all algorithms and a comparison against the VF2 algorithm.

The average time complexity of algorithm for motif M_2^1 is:

$$\begin{aligned} & O(1) + O\left(|U| \cdot \binom{\frac{|E|}{|U|}}{2}\right) \\ =& O\left(|U| \cdot (\frac{|E|}{|U|})^2\right) \\ =& O(\frac{|E|^2}{|U|}) \end{aligned} \tag{1}$$

For sparse graphs, $|E| = c|N|$ so the final complexity for M_2^1 is $O(N)$ which is faster than VF2's best case $O(N^2)$. As the topological structure of M_2^1 and M_1^2 is identical, they have the same average time complexity.

The average time complexity of algorithm for motif M_3^1 is:

$$\begin{aligned} & O(1) + O\left(|U| \cdot \binom{\frac{|E|}{|U|}}{3}\right) \\ =& O\left(|U| \cdot (\frac{|E|}{|U|})^3\right) \\ =& O(\frac{|E|^3}{|U|^2}) \end{aligned} \tag{2}$$

For sparse graphs, $|E| = c|N|$ so the final complexity for M_2^1 is $O(N)$ which is faster than VF2's best case $O(N^2)$. As the topological structure of M_3^1 and M_1^3 is identical, they have the same average time complexity.

The average time complexity of algorithm for motif M_2^{2-} is:

$$\begin{aligned} & O\left(\binom{|V|}{2} \cdot (|S| + |S| \cdot |P| + |P|)\right) \\ =& O\left(|V|^2 \cdot |S| \cdot |P|\right) \\ \leq& O\left(|V|^2 \cdot (\frac{|S| + |P|}{2})^2\right) \\ \leq& O\left(|V|^2 \cdot (\frac{|E|}{|V|})^2\right) \\ =& O(|E|^2) \end{aligned} \tag{3}$$

For sparse graphs, $|E| = c|N|$ so the final complexity is $O(N^2)$ which is the same as VF2's best case.

The average time complexity of algorithm for motif M_2^2 is:

$$
\begin{aligned}
& O\left(\binom{|V|}{2} \cdot (1 + |S| + \binom{|S|}{2})\right) \\
=& O\left(|V|^2 \cdot |S|^2\right) \\
\leq& O\left(|V|^2 \cdot (\frac{|E|}{|V|})^2\right) \\
=& O(|E|^2)
\end{aligned} \tag{4}
$$

For sparse graphs, $|E| = c|N|$ so the final complexity is $O(N^2)$ which is the same as VF2's best case.

4.2 MotifGCN

In this section, we present the architecture of our framework MotifGCN, as illustrated in Fig. 4. We first generate motif adjacency matrices from the interaction graph and then combine them with a combination weight. The combined matrix will then be sent to graph convolution layers to generate the final embeddings. After that, the user and item embeddings are split and used to calculate the dot product for predicting user's preference. The motif adjacency matrices generation algorithm only needs to run once for each motif on each dataset and the matrices can be reused in later experiments. The node-level propagation of MotifGCN is defined by the following equations.

$$
\mathbf{h}_u^{l+1} = \sum_{m \in M} \sum_{i \in \mathcal{N}_m(u)} \frac{1}{c_{ui}} \alpha_m \mathbf{h}_i^l \tag{5}
$$

$$
\mathbf{h}_i^{l+1} = \sum_{m \in M} \sum_{u \in \mathcal{N}_m(i)} \frac{1}{c_{ui}} \alpha_m \mathbf{h}_u^l \tag{6}
$$

In equations 5 and 6, $\mathbf{h}_u^l, \mathbf{h}_i^l \in \mathbb{R}^d$ denote the embedding vectors of user node u and item node i at layer l. M is the set of all selected motifs including m_1. $\mathcal{N}_m(u)$ and $\mathcal{N}_m(i)$ denote the motif neighbours of user node u and item

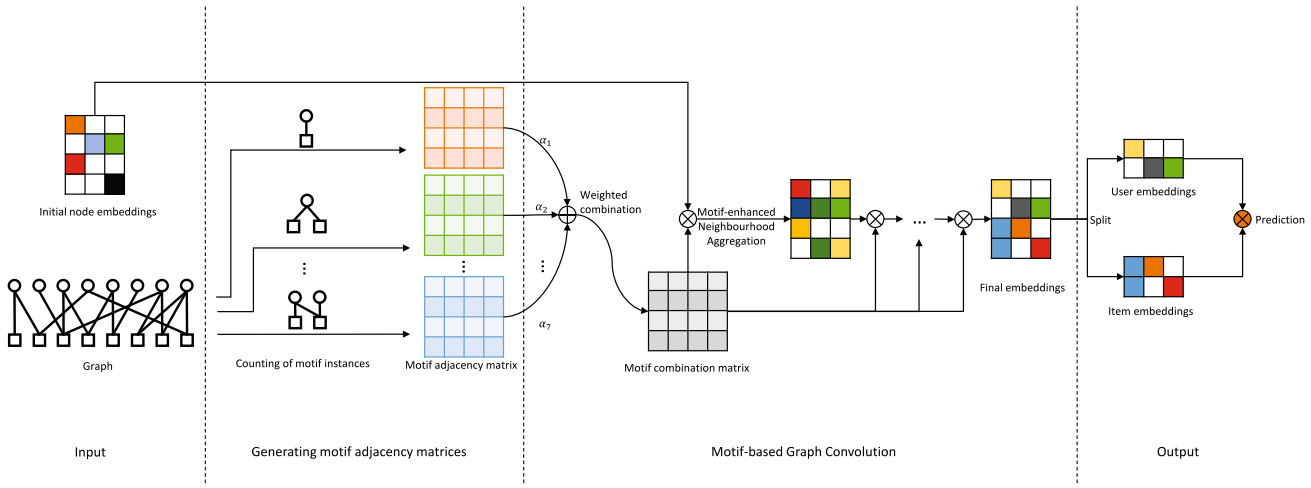

Fig. 4 An illustration of MotifGCN model architecture.

node i for motif m respectively. The normalisation constant c_{ui} is computed by $\sqrt{|\mathcal{N}_m(u)||\mathcal{N}_m(i)|}$ and α_m is the motif combination coefficient of motif m.

To explore whether network motifs can improve GCNs, we hope to emphasise the influence of motif adjacency matrices to the performance as much as possible. Following LightGCN [6], we ignore the linear transformation matrix, non-linear activation function, and self-loops in the original GCN layer [10] to restrict the quantity of parameters. To explicitly test the performance of individual motifs, we set $M = \{m_1, m\}$ for each motif m other than m_1 and set $\alpha_1 = \alpha_x = 0.5$. The motif combination coefficient can also be learned automatically such as using the attention mechanism [35], which will be explored in future work.

After obtaining the final node embeddings, the prediction scores between any user–item pairs can be calculated as the dot product of their embedding vectors:

$$\hat{y}_{ui} = \mathbf{h}_u^T \cdot \mathbf{h}_i \tag{7}$$

The prediction scores will be ranked to generate the top-p list of nodes as the recommendation results.

The matrix form of the layer propagation is provided as follows:

$$\mathbf{H}^{l+1} = \widetilde{\mathbf{A}}_{mc}\mathbf{H}^l, \tag{8}$$

$$\widetilde{\mathbf{A}}_{mc} = \sum_m \alpha_m \widetilde{\mathbf{A}}_m, \tag{9}$$

$$\widetilde{\mathbf{A}}_m = \mathbf{D}_m^{-1/2}\mathbf{A}_m\mathbf{D}_m^{-1/2}, \tag{10}$$

where $\widetilde{\mathbf{A}}_{mc}$ is the motif combination matrix, $\widetilde{\mathbf{A}}_m$ is the symmetrically normalised motif adjacency matrix, and $\mathbf{D}_m$ is the degree matrix of motif m's motif adjacency matrix $\mathbf{A}_m$. The motif adjacency matrices including the original adjacency matrix are generated from the edge set E and $\mathbf{A}_m \in \mathbb{R}^{N\times N}$. The embedding matrix at layer l is denoted by $\mathbf{H}^l \in \mathbb{R}^{N\times d}$.

4.3 Optimisation settings

Following the common choices of optimisation strategies for collaborative filtering, we use Bayesian personalised ranking [53] as loss function and adaptive moment estimation [54] as optimiser. For the generalisation strategy, we adopt L2 regularisation without dropout since our model only has the initial node embeddings as the learnable parameters.

5 Experiments

In this section, we present experimental evaluations of our work in two subsections which correspond to sub-RQ1 and sub-RQ2. To help answer the first sub-research question, we evaluate the efficiency of our dedicated motif adjacency matrix generation algorithms. With respect to the second sub-research question, we assess the recommendation performance of our framework MotifGCN.

Table 1 Comparison of motif adjacency matrix generation algorithms

Motif	Algorithm	ProgrammableWeb		Yelp	
		Running time	Peak memory usage	Running time	Peak memory usage
M_2^1	VF2	0.43s	0.11 GB	/	OOM
	Ours	0.05s	0.06 GB	121.94s	7.38 GB
M_1^2	VF2	53.11s	3.24 GB	/	OOM
	Ours	8.58s	0.36 GB	345.54s	8.76 GB
M_3^1	VF2	2.33s	0.24 GB	/	OOM
	Ours	0.08s	0.06 GB	281.49s	6.96 GB
M_1^3	VF2	/	OOM	/	OOM
	Ours	8.61s	0.36 GB	346.86s	8.76 GB
M_2^{2-}	VF2	228.14s	8.36 GB	/	OOM
	Ours	33.73s	0.15 GB	10865.24s	51.90 GB
M_2^2	VF2	7.64s	0.46 GB	/	OOM
	Ours	1.9s	0.08 GB	657.56s	2.03 GB

5.1 Evaluating efficiency of motif adjacency matrix generation algorithms

To evaluate the efficiency of our proposed motif adjacency matrix generation algorithms, we conduct experiments using two datasets: ProgrammableWeb and Yelp. We include this small dataset while ignoring the other three datasets used for model performance comparison in next section because VF2 is unable to generate any motif adjacency matrix for relatively small-sized yelp dataset due to out-of-memory (OOM). For the baseline method, we select VF2 algorithm as mentioned in Sect. 4.1.

Datasets. ProgrammableWeb: A mashup-API network consisting of 1,611 mashup nodes, 6,336 API nodes, and 13,219 interactions. Yelp: A business review dataset containing 29,601 users, 24,734 items, and 1,517,326 interactions.

Baseline algorithm. VF2 algorithm: A well-known algorithm for subgraph isomorphism detection that is applicable to most type of graph including directed, undirected, bipartite, and homogeneous graphs.

Hardware configuration. All experiments are conducted on a computer with an AMD Ryzen 9 7900 CPU and 128GB RAM. The efficiency of evaluated methods is measured in running time and peak memory usage.

The results are shown in Table 1. From the comparison, we can draw a conclusion that our proposed dedicated algorithms for generating motif adjacency matrices of bipartite graphs are significantly faster than the VF2 algorithm and consume less memory usage as well.

5.2 Evaluating performance of MotifGCN

5.2.1 Datasets and evaluation metrics

The statistics of the four datasets used for performance evaluation are shown in Table 2. The records in each dataset are randomly split into training, validation, and test sets with a ratio of 70%, 20% , and 10% , respectively. All models are evaluated by top-20 and top-40 recommendations with the evaluation metrics of recall and normalised discounted cumulative gain (NDCG).

Table 2 Statistics of the datasets

Dataset	# of Users	# of Items	# of Interactions	Density
Yelp	29,601	24,734	1,517,326	0.00051
Gowalla	50,821	57,440	1,172,425	0.00010
Amazon	78,578	77,801	2,240,156	0.00009
Tmall	47,939	41,390	2,357,450	0.00030

5.2.2 Baseline methods

We select seven baseline methods from the perspectives of basic graph CF, higher-order-enhanced graph CF, and self-supervised graph CF methods. Although self-supervised learning is out of scope in this paper, we have included this type of methods as they are commonly regarded as state-of-the-arts for CF [55].

- **Basic Graph CF:**
  - **NCF** [56] is an CF model leveraging MLPs to capture non-linearity in the data.
  - **LightGCN** [6] is a simplified GCN model for recommender systems. It keeps the linear propagation of node embeddings while removing all feature transformation and non-linear activation functions in the model architecture.
- **Higher-order-enhanced Graph CF:**
  - **HyperRec** [40] uses a series of hypergraphs from different time periods to generate dynamic node embeddings that are aware of multi-hop relationships.
  - **MHCN** [45] considers directed triangle motifs to incorporate multi-hop interactions and social relationships into recommendation. It also adapts attention mechanism and self-supervised learning in the training strategy.
  - **HCCF** [46] leverages hypergraphs to model higher-order connections and use the resulting embeddings as a view to contrast with the embeddings generated from normal graph convolution.
- **Self-supervised Graph CF:**
  - **SGL** [57] is a self-supervised CF model that constructs contrasting views by randomly dropout edges or nodes from the graph.
  - **SimGCL** [58] is a self-supervised CF model that constructs contrasting views by simply adding noise to the node embeddings.

5.2.3 Hyperparameter settings

We compare our results with those reported in [59] which also used the four datasets for performance evaluation. The hyperparameters of all baseline methods are tuned according to the original papers, except that all models have the following same settings: Embeddings size is 32; the batch size is 256; the number of MLPs or graph convolution layers is two.

For MotifGCN, we search the learning rate from $\{1e^{-4}, 1e^{-3}, 1e^{-2}\}$. The L_2 regularisation term is searched from $\{1e^{-5}, 1e^{-4}, 1e^{-3}\}$. The motif type is searched from $\{M_2^1, M_1^2, M_3^1, M_1^3, M_2^{2-}, M_2^2\}$.

Table 3 Performance comparison with baseline methods

Dataset	Metric	NCF [56]	LightGCN [6]	HyperRec [40]	MHCN [45]	HCCF [46]	SGL [57]	SimGCL [58]	**MotifGCN**
Yelp	Recall@20	0.0252	0.0482	0.0472	0.0503	0.0626	0.0526	0.0718	**0.0777**
	NDCG@20	0.0202	0.0409	0.0395	0.0424	0.0527	0.0444	0.0615	**0.0663**
	Recall@40	0.0487	0.0803	0.0791	0.0826	0.1040	0.0869	0.1166	**0.1274**
	NDCG@40	0.0289	0.0527	0.0522	0.0544	0.0681	0.0571	0.0778	**0.0846**
Gowalla	Recall@20	0.0171	0.0985	0.0901	0.0955	0.1070	0.1030	0.1357	**0.1672**
	NDCG@20	0.0106	0.0593	0.0498	0.0574	0.0644	0.0623	0.0818	**0.0994**
	Recall@40	0.0216	0.1431	0.1356	0.1393	0.1535	0.1500	0.1956	**0.2391**
	NDCG@40	0.0118	0.0710	0.0660	0.0689	0.0767	0.0746	0.0975	**0.1183**
Amazon	Recall@20	0.0142	0.0319	0.0302	0.0296	0.0322	0.0327	0.0474	**0.0684**
	NDCG@20	0.0085	0.0236	0.0225	0.0219	0.0247	0.0249	0.0360	**0.0523**
	Recall@40	0.0223	0.0499	0.0432	0.0489	0.0525	0.0531	0.0750	**0.1071**
	NDCG@40	0.0133	0.0290	0.0246	0.0284	0.0314	0.0312	0.0451	**0.0651**
Tmall	Recall@20	0.0082	0.0225	0.0233	0.0203	0.0314	0.0268	0.0473	**0.0577**
	NDCG@20	0.0059	0.0154	0.0160	0.0139	0.0213	0.0183	0.0328	**0.0401**
	Recall@40	0.0140	0.0378	0.0350	0.0340	0.0519	0.0446	0.0766	**0.0927**
	NDCG@40	0.0079	0.0208	0.0199	0.0188	0.0284	0.0246	0.0429	**0.0522**

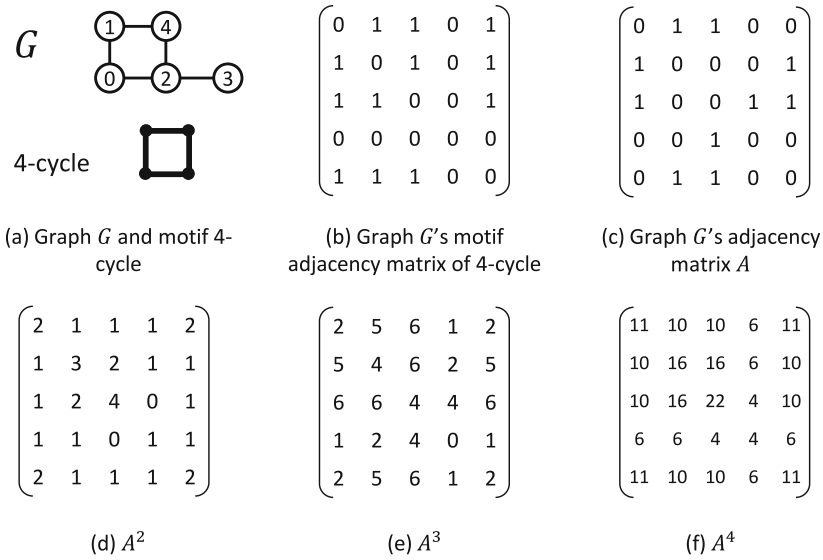

Fig. 5 An example showing the difference between powers of adjacency matrix and motif adjacency matrix.

5.3 Performance comparison

We present the performance comparison in Table 3. From the results, we can observe that MotifGCN shows superior performance against all baseline methods including the state-of-the-art self-supervised learning methods.

The improvement of MotifGCN over its backbone model LightGCN is significant: around 60%–70% on Yelp and Gowalla and more than 100% on Amazon and Tmall for all evaluation metrics. This performance improvement could be attributed to network motif's high efficiency of capturing higher-order information within single layer.

Comparing to the three higher-order-enhanced graph CF baseline methods, our framework does not rely on any attention mechanism or self-supervised learning strategy yet outperforms them by a large margin. The superiority

of MotifGCN indicates that our adopted undirected network motifs can better enhance graph CF models than general hypergraph approaches [40, 46].

6 Discussion

In this section, we discuss how motif adjacency matrices are related to powers of the original adjacency matrix and how motif adjacency matrices change the behaviour of graph convolution layer.

A motif adjacency matrix is not a trivial partition or transformation of the original adjacency matrix. Due to the definition of motif adjacency matrix, any two nodes in the motif become connected in the motif adjacency matrix regardless of the actual connectivity in the graph. This means that motif adjacency matrices can enhance the message-passing process, as a node can aggregate information from distant nodes that are only reachable after multiple layers with the original adjacency matrix. In addition, the elements in the matrix are counts of motif instances where node pairs appear together, so most of the elements are different while in the original adjacency matrix elements are either one or zero. This is effectively a redistribution of the weights in neighbourhood aggregation where nodes that appear in more motif instances together with current node have larger weights.

A series of previous work [6, 60, 61] simplify the traditional message-passing architecture by eliminating linear transformation, non-linear activation, and self-loops from the graph convolution layers and have powers of the adjacency matrix for the neighbourhood aggregation. According to algebraic graph theory, an element of powers of adjacency matrix, $(\mathbf{A}^n)_{ij}$, is the number of walks from i to j in the graph with length n [62]. For node i, a larger number of walks to j than to other nodes is an indication of j being structurally more important than other nodes for i. The walks include all possible pathways within the walk length so some of them can construct certain motifs. In other words, the powers of adjacency matrix contain information of various motifs. However, a large proportion of these walks contain repeated nodes and edges, which fails to reflect the distinctive impacts of different local structures to graph convolution. In addition, the number of walks is highly dependent on the parity of the walk length, causing imbalanced neighbourhood aggregation for different layers.

In comparison, the elements in a motif adjacency matrix only depend on the structure itself. Figure 5 illustrates an example of the difference between powers of adjacency matrix and motif adjacency matrix. Figure 5 a is simple graph where sequentially connected nodes 0, 1, 4, and 2 form an instance of a motif called four-cycle. In Fig. 5 b, nodes 0, 1, 2, and 4 have equal element values between each other in the motif adjacency matrix as they appear in a same motif instance.

In Fig. 5c–f, we can see that all nodes eventually are able to aggregate information from each other and some element values grow faster than others as the exponent increases, especially node 2 who has the largest degree in the graph. This observation is consistent with the analysis of the over-squashing issue by Xu et al. [15]. By contrast, a motif adjacency matrix captures higher-order interactions for selective nodes depending on the motif structure, without fast increasing edge weights as in powers of adjacency matrix. This observation implies motif adjacency matrices are able to alleviate the issue of over-squashing by limiting the number of neighbours and the weights of neighbourhood information. As a result, MotifGCN is able to achieves significantly improved performance.

7 Conclusion and future work

In this work, we proposed to use network motifs to reinforce the architecture of GCNs motivated by an analysis on the effect of shortcuts in GCNs. We studied the network motifs on bipartite graphs and proposed dedicated algorithms for generating motif adjacency matrices which have a significantly lower time complexity than existing approaches. We then presented MotifGCN, a novel framework for recommender systems that incorporates motif adjacency matrices in graph convolution design. We conducted experiments on four real-world

datasets and showed that motifs can significantly improve a simple GCN's performance on recommendation tasks.

While our method is designed primarily for bipartite graphs, the underlying principles can be easily extended to homogeneous graphs as well. Applications on homogeneous graphs and extra information such as user–user social connections are beyond the scope of this work. For future work, we will explore different techniques that can be combined with motif analysis such as dimension reduction algorithms and spectral clustering methods.

Acknowledgements Not applicable.

Author Contributions Conceptualisation and methodology were contributed by Yuqi Zhang and Jian Yu; Software and investigation were involved by Yuqi Zhang; Validation was performed by Zhizhong Liu and Guiling Wang; Formal analysis was done by Jian Yu and Quan Z. Sheng; Writing–original draft preparation did by Yuqi Zhang and Nancy Wang; Writing–review and editing and supervision were carried out by Jian Yu and Minh Nguyen.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. We declare that no funds were received during the preparation of this manuscript.

Data Availability The data that support the findings of this study are openly available in Github at https://github.com/LatentAdventurer/datasets.

Declarations

Conflict of interest We declare that there have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Piao J, Zhang G, Xu F, Chen Z, Li Y (2021). Predicting Customer Value with Social Relationships via Motif-based Graph Attention Networks. In: Proceedings of the Web Conference 2021. WWW '21, pp. 3146–3157. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3442381.3449849 . Accessed 2021-08-03
2. Wang Z, Wei W, Cong G, Li X.-L, Mao X.-L, Qiu M (2020). Global Context Enhanced Graph Neural Networks for Session-based Recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '20, pp. 169–178. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3397271.3401142 . Accessed 2020-08-20
3. Yu W, Qin Z (2020). Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters. In: International Conference on Machine Learning . http://arxiv.org/abs/2006.15516 Accessed 2022-03-31
4. Zhou X, Zhuang F, Xu C, Zhao P, Liu Y, Xu J, Sheng V.S, Fang J (2019). Graph Contextualized Self-Attention Network for Session-based Recommendation. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, pp. 3940–3946 . https://www.ijcai.org/Proceedings/2019/547 Accessed 2020-08-10
5. Anwaar M.U, Han Z, Arumugaswamy S, Khan R.A, Weber T, Qiu T, Shen H, Liu Y, Kleinsteuber M (2021). Metapath and Entity-aware Graph Neural Network for Recommendation. arXiv:2010.11793 [cs] . Accessed 2021-11-09
6. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (2020). LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '20, pp. 639–648. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3397271.3401063 . Accessed 2020-08-20
7. Wu Q, Zhang H, Gao X, He P, Weng P, Gao H, Chen G (2019). Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In: The World Wide Web Conference. WWW

'19, pp. 2091–2102. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3308558.3313442 . Accessed 2022-03-30
8. Wang X, He X, Wang M, Feng F, Chua T.-S (2019). Neural Graph Collaborative Filtering. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 165–174 https://doi.org/10.1145/3331184.3331267 . arXiv: 1905.08108. Accessed 2021-11-25
9. Yu W, Zhang Z, Qin Z (2022). Low-Pass Graph Convolutional Network for Recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence 36(8):8954–8961
10. Kipf T.N, Welling M (2017). Semi-Supervised Classification with Graph Convolutional Networks. In: International Conference on Learning Representations . http://arxiv.org/abs/1609.02907 Accessed 2020-05-27
11. Tan F, Zhou L, Lu J, Zhang H (2024) Fixed-time synchronization in multilayer networks with delay cohen-grossberg neural subnets via adaptive quantitative control. Asian J Control 26(1):446–455. https://doi.org/10.1002/asjc.3217
12. Tan F, Zhou L, Lu J, Quan H, Liu K (2023) Adaptive quantitative control for finite time synchronization among multiplex switched nonlinear coupling complex networks. Eur J Control 70:100764. https://doi.org/10.1016/j.ejcon.2022.100764
13. Zhou L, Lin H, Tan F (2023) Fixed/predefined-time synchronization of coupled memristor-based neural networks with stochastic disturbance. Chaos, Solitons Fractals 173:113643. https://doi.org/10.1016/j.chaos.2023.113643
14. Zhou L, Lin H, Tan F (2024) Unified quantified adaptive control for multiple-time stochastic synchronization of coupled memristive neural networks. Neurocomputing 577:127384. https://doi.org/10.1016/j.neucom.2024.127384
15. Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi K.-i, Jegelka S (2018). Representation Learning on Graphs with Jumping Knowledge Networks. arXiv:1806.03536 [cs, stat] . Accessed 2020-07-30
16. Monti F, Otness K, Bronstein M.M (2018). Motifnet: a Motif-based graph convolutional network for directed graphs. In: 2018 IEEE Data Science Workshop (DSW), pp. 225–228 . https://doi.org/10.1109/DSW.2018.8439897
17. Zhao H, Xu X, Song Y, Lee DL, Chen Z, Gao H (2021) Ranking Users in Social Networks with Motif-Based PageRank. IEEE Trans Knowl Data Eng 33(05):2179–2192. https://doi.org/10.1109/TKDE.2019.2953264. (**Accessed 2022-03-30**)
18. Li X, Wei W, Feng X, Liu X, Zheng Z (2021) Representation learning of graphs using graph convolutional multilayer networks based on motifs. Neurocomputing 464:218–226. https://doi.org/10.1016/j.neucom.2021.08.028. Accessed 2022-03-31
19. Peng H, Li J, Gong Q, Ning Y, Wang S, He L (2020). Motif-Matching Based Subgraph-Level Attentional Convolutional Network for Graph Classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5387–5394 . https://doi.org/10.1609/aaai.v34i04.5987 . https://ojs.aaai.org/index.php/AAAI/article/view/5987 Accessed 2021-08-03
20. Chen X, Cai R, Fang Y, Wu M, Li Z, Hao Z (2022). Motif Graph Neural Network. arXiv:2112.14900 [cs] . Accessed 2022-03-14
21. Dareddy M.R, Das M, Yang H (2019). motif2vec: Motif Aware Node Representation Learning for Heterogeneous Networks. arXiv:1908.08227 [cs] . Accessed 2021-07-14
22. Liu X, Song Y (2022). Graph Convolutional Networks with Dual Message Passing for Subgraph Isomorphism Counting and Matching. In: 36th AAAI Conference On Artificial Intelligence 36(7):7594–7602
23. Yu Z, Gao H (2022). Molecular Graph Representation Learning via Heterogeneous Motif Graph Construction. arXiv:2202.00529 [cs] . Accessed 2022-03-14.
24. Hu Q, Lin F, Wang B, Li C (2022) MBRep: Motif-based representation learning in heterogeneous networks. Expert Syst Appl 190:116031. https://doi.org/10.1016/j.eswa.2021.116031. Accessed 2022-03-13.
25. Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002) Network Motifs: Simple Building Blocks of Complex Networks. Science 298(5594):824–827. https://doi.org/10.1126/science.298.5594.824. (**Accessed 2022-03-15**)
26. Wang G, Yu J, Nguyen M, Zhang Y, Yongchareon S, Han Y (2023) Motif-based graph attentional neural network for web service recommendation. Knowl-Based Syst 269:110512. https://doi.org/10.1016/j.knosys.2023.110512. Accessed 2023-05-08.
27. Velič ković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018). Graph Attention Networks. In: International Conference on Learning Representations . https://openreview.net/forum?id=rJXMpikCZ Accessed 2022-03-30
28. Hamilton W, Ying Z, Leskovec J (2017). Inductive Representation Learning on Large Graphs. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates Inc, https://papers.nips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html Accessed 2022-03-31
29. Ouyang Z, Zhang C, Hou S, Zhang C, Ye Y. How to improve representation alignment and uniformity in graph-based collaborative filtering? In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 18, pp. 1148–1159. https://doi.org/10.1609/icwsm.v18i1.31379 . https://ojs.aaai.org/index.php/ICWSM/article/view/31379 Accessed 2024-08-29
30. Xu K, Hu W, Leskovec J, Jegelka S (2018). How Powerful are Graph Neural Networks? In: International Conference on Learning Representations . https://openreview.net/forum?id=ryGs6iA5Km Accessed 2022-03-30
31. Bouritsas G, Frasca F, Zafeiriou SP, Bronstein M (2022) Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. IEEE Trans Pattern Anal Mach Intell 01:1–1. https://doi.org/10.1109/TPAMI.2022.3154319. (**Accessed 2022-03-31**)

32. Rossi R.A, Ahmed N.K, Koh E (2018). Higher-order Network Representation Learning. In: Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18, pp. 3–4. ACM Press, Lyon, France . https://doi.org/10.1145/3184558.3186900 . http://dl.acm.org/citation.cfm?doid=3184558.3186900 Accessed 2022-03-31
33. Benson AR, Gleich DF, Leskovec J (2016) Higher-order organization of complex networks. Science 353(6295):163–166. https://doi.org/10.1126/science.aad9029. (**Accessed 2020-09-29**)
34. Lee J.B, Rossi R.A, Kong X, Kim S, Koh E, Rao A (2019). Graph Convolutional Networks with Motif-based Attention. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. CIKM '19, pp. 499–508. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3357384.3357880 . Accessed 2021-01-25
35. Sankar A, Zhang X, Chang K.C.-C (2019). Motif-based Convolutional Neural Network on Graphs. arXiv:1711.05697 [cs] . Accessed 2021-02-04
36. Sankar A, Wang J, Krishnan A, Sundaram H (2020). Beyond Localized Graph Neural Networks: An Attributed Motif Regularization Framework. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 472–481 . https://doi.org/10.1109/ICDM50108.2020.00056
37. Li H, Wang B, Cui L, Bai L, Hancock E (2020). LGL-GNN: Learning Global and Local Information for Graph Neural Networks. In: S+SSPR . https://doi.org/10.1007/978-3-030-73973-7_13
38. Zhang S, Hu Z, Subramonian A, Sun Y (2024) Motif-driven contrastive learning of graph representations. IEEE Trans Knowl Data Eng 36(8):4063–4075. https://doi.org/10.1109/TKDE.2024.3364059. Accessed 2024-08-28.
39. Zhou Z, Zhou S, Mao B, Chen J, Sun Q, Feng Y, Chen C, Wang C. Motif-driven Subgraph Structure Learning for Graph Classification. arXiv. 1048550/arXiv.2406.08897 .Accessed 2024-08-28
40. Wang J, Ding K, Hong L, Liu H, Caverlee J. Next-item recommendation with sequential hypergraphs. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '20, pp. 1101–1110. Association for Computing Machinery. https://doi.org/10.1145/3397271.3401133 . https://dl.acm.org/doi/10.1145/3397271.3401133 Accessed 2024-05-12
41. Ji S, Feng Y, Ji R, Zhao X, Tang W, Gao Y. Dual channel hypergraph collaborative filtering. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '20, pp. 2020–2029. Association for Computing Machinery. https://doi.org/10.1145/3394486.3403253 . https://dl.acm.org/doi/10.1145/3394486.3403253 Accessed 2024-05-30
42. Zhang H, McAuley J (2020). Stacked mixed-order graph convolutional networks for collaborative filtering. In: Proceedings of the 2020 SIAM International Conference on Data Mining (SDM). Proceedings, pp. 73–81. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611976236.9 . https://epubs.siam.org/doi/abs/10.1137/1.9781611976236.9 Accessed 2024-05-30
43. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition 1512.03385. Accessed 2020-08-24
44. Nguyen M, Yu J, Nguyen T, Yongchareon S (2022) High-order autoencoder with data augmentation for collaborative filtering. Knowl-Based Syst 240:107773. https://doi.org/10.1016/j.knosys.2021.107773. Accessed 2022-10-12.
45. Yu J, Yin H, Li J, Wang Q, Hung N.Q.V, Zhang X (2021). Self-supervised multi-channel hypergraph convolutional network for social recommendation. In: Proceedings of the Web Conference 2021. WWW '21, pp. 413–424. Association for Computing Machinery. https://doi.org/10.1145/3442381.3449844. Accessed 2023-08-10.
46. Xia L, Huang C, Xu Y, Zhao J, Yin D, Huang J. Hypergraph contrastive collaborative filtering. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '22, pp. 70–79. Association for Computing Machinery. https://doi.org/10.1145/3477495.3532058. Accessed 2023-08-10
47. Xia J, Li D, Gu H, Lu T, Zhang P, Shang L, Gu N. Hierarchical graph signal processing for collaborative filtering. In: Proceedings of the ACM Web Conference 2024. WWW '24, pp. 3229–3240. Association for Computing Machinery. https://doi.org/10.1145/3589334.3645368. Accessed 2024-08-27
48. Gong K, Song X, Li W, Wang S (2024) HN-GCCF: High-order neighbor-enhanced graph convolutional collaborative filtering. Knowl-Based Syst 283:111–122. https://doi.org/10.1016/j.knosys.2023.111122. Accessed 2024-05-30.
49. Diestel R (2017). Graph Theory, Fifth edition edn. Graduate Texts in Mathematics, vol. 173. Springer, New York
50. West D.B, et al (2001). Introduction to Graph Theory (vol. 2). Upper Saddle River: Prentice hall
51. Das S, Ghosh P, Ghosh S, Sen S (2021) Oriented bipartite graphs and the Goldbach graph. Discrete Math 344(9):112497. https://doi.org/10.1016/j.disc.2021.112497. (**Accessed 2023-01-30**)
52. Cordella LP, Foggia P, Sansone C, Vento M (2004) A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans Pattern Anal Mach Intell 26(10):1367–1372. https://doi.org/10.1109/TPAMI.2004.75
53. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009). BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. UAI '09, pp. 452–461. AUAI Press, Arlington, Virginia, USA
54. Kingma D.P, Ba J (2015). Adam: A Method for Stochastic Optimization. In: International Conference for Learning Representations . 10.48550/arXiv.1412.6980 . arXiv:1412.6980 [cs]. http://arxiv.org/abs/1412.6980 Accessed 2023-01-29
55. Yu J, Yin H, Xia X, Chen T, Li J, Huang Z (2023) Self-supervised learning for recommender systems: A survey. IEEE Trans Knowl Data Eng 36(1):335–355. https://doi.org/10.1109/TKDE.2023.3282907

56. He X, Liao L, Zhang H, Nie L, Hu X, Chua T.-S (2017). Neural Collaborative Filtering. arXiv:1708.05031 [cs] . Accessed 2020-06-03
57. Wu J, Wang X, Feng F, He X, Chen L, Lian J, Xie X. Self-supervised graph learning for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '21, pp. 726–735. Association for Computing Machinery. https://doi.org/10.1145/3404835.3462862 . Accessed 2023-04-07
58. Yu J, Yin H, Xia X, Chen T, Cui L, Nguyen Q.V.H. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '22, pp. 1294–1303. Association for Computing Machinery. https://doi.org/10.1145/3477495.3531937. Accessed 2023-04-07
59. Cai X, Huang C, Xia L, Ren X. LightGCL: Simple yet effective graph contrastive learning for recommendation. https://openreview.net/forum?id=FKXVK9dyMM Accessed 2023-10-15
60. Wu F, Zhang T, Souza Jr. A.H.d, Fifty C, Yu T, Weinberger K.Q (2019). Simplifying Graph Convolutional Networks. arXiv:1902.07153 [cs, stat] . Accessed 2021-04-26
61. Zhu H, Koniusz P (2020). Simple Spectral Graph Convolution. https://openreview.net/forum?id=CYO5T-YjWZV Accessed 2021-04-13
62. Godsil C, and Royle G.F (2001). Algebraic Graph Theory (vol. 207). Springer Science & Business Media

Authors and Affiliations

Yuqi Zhang[1] · Jian Yu[1] · Zhizhong Liu[2] · Guiling Wang[3] · Minh Nguyen[1] · Quan Z. Sheng[4] · Nancy Wang[1]

✉ Jian Yu
jian.yu@aut.ac.nz

[1] Department of Computer and Information Sciences, Auckland University of Technology, Auckland Central 1010, New Zealand

[2] School of Computer and Control Engineering, Yantai University, Yantai, Shandong 264005, China

[3] School of Information Science and Technology, North China University of Technology, Beijing 100144, China

[4] School of Computing, Macquarie University, Sydney, NSW 2109, Australia