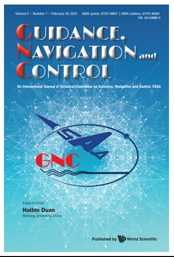
ISSN (print): 2737-4807 | ISSN (online): 2737-4920 | CN: 10-1988/V

Guidance, Navigation and Control

https://www.worldscientific.com/worldscinet/gnc



Investigating Hypernode Classification of Complex Systems Based on High-order Graph Neural Networks

Jiawen Chen [*], Yanyan He [†], Duxin Chen [*,‡] and Wenwu Yu [*]

[*] *School of Mathematics, Southeast University, Nanjing 210096, P. R. China*
[†] *School of Cyber Science and Engineering, Southeast University, Nanjing 210096, P. R. China*

Received 6 August 2024; Revised 17 September 2024; Accepted 23 September 2024; Published 27 December 2024

KEYWORDS
Graph neural network;
high-order network;
graph isomorphism network;
hypernode classification

Abstract Investigating latent interactions beyond direct connections is essential for analyzing complex networks. However, traditional graph structures often fail to capture complex relationships, especially in the high-order interactions among multiple individuals. To address this issue, we extend the graph isomorphism network (GIN) framework to hypergraphs, treat nodes as self-hyperedges, and propose the self-hypergraph isomorphism network (SHGIN). Meanwhile, the hypergraph Weisfeiler–Lehman (WL) test is also proposed to distinguish different isomorphisms of hypergraphs and improve the representation power of hypergraph neural networks. Extensive experiments on co-authorship and co-citation networks demonstrate the effectiveness of SHGIN. The results indicate that our model displays superior hypernode classification accuracy compared to traditional graph neural networks in semi-supervised learning (SSL). Furthermore, it surpasses existing hypergraph neural network models in co-authorship datasets, highlighting its effectiveness in capturing high-order relationships in complex networks.

1. Introduction

A network comprises numerous individuals that engage in communication, cooperation, coordination, scheduling, and control to manifest the functional and behavioral characteristics of the entire system. Each agent functions as an independent and autonomous entity within the system.[1] In practical scenarios, individuals with diverse characteristics in the network are segmented into various groups, exhibiting heterogeneity and extensive distribution. For instance, human social networks inherently form agent clusters, encompassing diverse individuals and organizations within their respective industry domains.[2] Within these groups, individuals engage in interactions and information exchange using diverse approaches, including connections, sharing information, and collaborative efforts. However, the complexity and diversity of complex networks often restrict observers from identifying solely the direct connections and interactions between the systems. Data correlations can be more intricate than simple pairwise relationships, posing challenges in modeling the potential high-order cooperative connections within traditional graph structures.[3] Investigating the

[‡] Corresponding author.
E-mail address: jiawenchen@seu.edu.cn (Jiawen Chen); heyy@seu.edu.cn (Yanyan He); chendx@seu.edu.cn (Duxin Chen); wwyu@seu.edu.cn (Wenwu Yu)
Cite this: Jiawen Chen, Yanyan He, Duxin Chen and Wenwu Yu, Investigating Hypernode Classification of Complex System Based on High-order Graph Neural Networks, *Guid. Navig. Control*, **5**(1), 57–69 (2025), doi: 10.1142/S2737480724500249

indirect influence and interaction of different groups enables a comprehensive understanding of the behavioral characteristics of individuals and facilitates the prediction of the evolution and trends of these groups.

Motivations: Research of complex interactions in swarm control on networks motivates effective decision-making. Hypergraphs, as high-order networks, effectively model many-to-many relationships by allowing each hyperedge to connect multiple nodes,[4] offering significant advantages in swarm formations and guidance systems. Traditional networks are limited to representing pairwise interactions, and hypergraphs provide the flexibility to model multi-agent collaboration and information sharing, particularly in swarms of drones, robots, or vehicles. Regular graphs typically handle local interactions in multi-agent systems, but hypergraphs extend this capability by directly modeling global relationships, imposing global formation constraints, and decomposing problems into subsystems to optimize global solutions.[5] In navigation and guidance systems, path planning involves complex environmental awareness and decision-making among agents, where their paths are closely interdependent (e.g., collision avoidance, coordinated navigation). High-order graph models can more effectively capture and optimize these inter-agent dynamics, enhancing multi-agent navigation and guidance in dynamic, complex environments.[6] In complex systems and swarm formations, only direct interactions between agents are typically observable, while high-order relationships[7] and identification system[8] are often latent and challenging to model. Node attributes and labels are frequently used to represent high-order interactions of agents, but many entities within swarm formations or networked clusters may lack such explicit attribute information. However, effectively modeling these high-order relationships requires the prediction of missing labels for supernodes, facing difficulties with incomplete or limited data, to better capture the dynamics and coordination within the system. Furthermore, the mathematical formulation of high-order interactions between multiple agents remains underdeveloped.

To address the problems of missing label information and formation grouping in the network, graph neural networks (GNN) have furnished powerful instruments for feature extraction and structure learning.[9–12] including node classification, node cluster, and link prediction. GNN facilitates the encapsulation of both local and global characteristics of networks, the discernment of relationships, the anticipation of individuals' behavior and cooperations. Graph variational autoencoder (VGAE)[11] learns to reconstruct potential connections between collaborators in team collaboration networks. Graph convolutional neural network (GCN)[13] encodes node features and graph structures in convolutional layers by coupling feature transformations and neighborhood aggregation, predicting missing node labels by node classification tack. Message passing neural network (MPNN)[14] updates node representations by passing information through the graph, while graph attention mechanism GAT[15] adaptively allocates node weights using masked self-attention layers, distinguishing the importance of different neighborhood nodes during information aggregation. GraphSAGE[16] achieves commendable results in relationship prediction and individual classification tasks in collaboration networks by sampling and integrating nodes' local neighborhoods considering different aggregators. However, the presence of diverse types of heterogeneous data leads to multimodal data representations, such as visual connections and text connections. Binary structures have limitations in expressing the correlation of multimodal data, restricting the application of traditional GNNs and making it challenging to explore high-order interactions between network individuals.

To tackle GNNs' inability to capture high-order interactions and influences, the combinations of hypergraph theory and neural networks have demonstrated universal learning capabilities and flexible model to investigate latent connections and model expressive ability. Hypergraph neural network (HGNN)[17] introduces the hypergraph Laplacian matrix into graph convolutional networks and proposes hypergraph convolution operations. By flexibly handling hyperedge scales, it integrates multimodal information into a unified structure and enables the classification of diverse groups or individual data. However, HGNN model introduces excessive noise during information fusion, resulting in poor performance in semi-supervised tasks. To filter potential noise and enhance the model's learning effectiveness, HyperGCN[18] employs the breaking ties randomly (BTR)[19] and mediator[20] algorithms for node sampling, selectively sampling binary edges after hypergraph expansion. DHNE[21] defines first-order and second-order similarities of hypergraphs, reconstructs the hypergraph Laplacian matrix containing structural information using an autoencoder, but is limited to handle fixed types and sizes of heterogeneous hyperedges. HyperSAGNN[22] utilizes self-attention mechanisms to aggregate hypergraph information, constructing pairwise attention coefficients between nodes as dynamic features, enabling flexible learning of relationships between nodes of different hyperedge scales. These methods typically assume that the hypergraph is unique and that each hyperedge contains at least two nodes. However, due to the limited representational capacity of hypergraphs, non-isomorphic hypergraphs are often embedded into identical node representations through domain-based neural network models, leading to a degradation in node classification accuracy. Moreover, existing approaches tend to neglect self-hyperedges, which represent critical self-relational information, resulting in the loss of intrinsic message-passing capabilities at the node level. Therefore, there is a pressing need for distinguishing non-isomorphic hypergraph structures, ensuring that HGNNs maintain both the uniqueness of structural mapping and improved representational power, particularly when predicting missing node labels.

Contributions: To address the limitations of classical GNN and HGNN methods, this paper focuses on enhancing the capabilities of HGNNs and bridging the gap in hypernode classification from the perspective of hypergraph isomorphism theory. The main contributions of this work are as follows. First, we propose Self-hypergraph Graph Isomorphism Network (SHGIN) to address the limitations of structural isomorphism and construct nodes themselves as self-hyperedges to achieve higher accuracy of hypernode classification than existing GNN and HGNN models. Second, we establish equivalent conditions between the hypergraph Weisfeiler–Lehman (WL) test and HGNNs. Specifically, we display that the hyperedge aggregation and graph readout functions yield a HGNN as powerful as the hypergraph WL test, capable of distinguishing non-isomorphic hypergraphs and reaching the expressive upper bound of HGNN. Third, we address the challenge that attributes between different individuals are often unobservable, complicating the detection of indirect effects and interactions within groups and among individuals. The absence of identifiable labels further impedes the assignment of individuals to distinct groups or formations. Finally, the state-of-the-art methods in this category are applied to demonstrate the effectiveness of classification of individual members in real-world networks.

2. High-order Graph Neural Networks

2.1. *High-order interaction discovery*

Definition 1 (Hypergraph). A hypergraph $H(V, E)$ is defined by a set of n hypernodes $V = \{v_1, v_2, \dots, v_n\}$ and a set of m hyperedges $E = \{e_1, e_2, \dots, e_m\}$. The adjacency matrix H is defined as

$$H_{ij} = \begin{cases} 1, & v_i \in e_j, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where $e_j = (v_1^{(j)}, \dots, v_k^{(j)})$ signifies an unordered set of nodes on hyperedge e_j, with $k = |e_j|$ denoting the number of nodes in the hyperedge.

Definition 2 (High-order Relationship Discovery). A network $G(V, E)$, capturing intricate high-order interactions among individuals within a group using a standard graph structure, can pose a significant challenge. For a subset of nodes $(v_1, v_2, \dots, v_k)$, a designed function f is defined as $f(v_1, v_2, \dots, v_k) \geq \tilde{p}$, $(v_1, v_2, \dots, v_k) \in E$, $f(v_1, v_2, \dots, v_k) < \tilde{p}$, $(v_1, v_2, \dots, v_k) \notin E$, where the function f utilizes the mean function of nodes' features, and $\tilde{p}$ is a threshold. If the score p obtained from the hyperedge's attention mechanism exceeds the threshold $\tilde{p}$, it indicates the presence of a high-order interaction among the designated node subset $(v_1, v_2, \dots, v_k)$. Conversely, if $p < \tilde{p}$, there is no high-order interaction existing between the subset.

Through the algorithm designed to uncover high-order relationships based on node vector features, the network's high-order interactions are revealed. For instances, this process entails constructing the hypergraph incidence matrix H using a node feature attention mechanism as shown in Figs. 1(a) and 1(b), where the node subset $(v_1, v_2, \dots, v_k)$ embodies potential hyperedges in the network. Traditional graph structures face limitations when representing relationships among individuals in group networks, emphasizing how the exploration of high-order interactions offers a superior and more versatile structural model for elucidating the indirect interactions between nodes.

2.2. *Related works*

To tackle the issue of missing label information, GCN[13] utilizes labeled data from individuals for semi-supervised learning (SSL) to classify missing label data. It can be defined as

$$H^{l+1} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^l W^l), \tag{2}$$

where the input is denoted as $H^l \in \mathbb{R}^{N\times D}$, $H^{(0)} = X$, X denotes the feature matrix and $\tilde{A} = A + I$ represents the adjacency matrix with self-connections, $\tilde{D}$ is the node degree matrix, $\tilde{d}_i = \sum_j \tilde{A}_{ij}$ denotes the degree matrix, W represents the parameters of the neural network, and σ is the activation function. However, traditional methods do not consider high-order interactions among individuals.

Subsequently, HGNN[17] was first introduced for learning hidden layer representations of high-order data structures. It designs hyperedge convolution operations to handle data correlations in the representation learning. Similarly, considering an input hypergraph signal $X \in R^{n\times C_1}$ containing n nodes with feature dimensions of C_2,

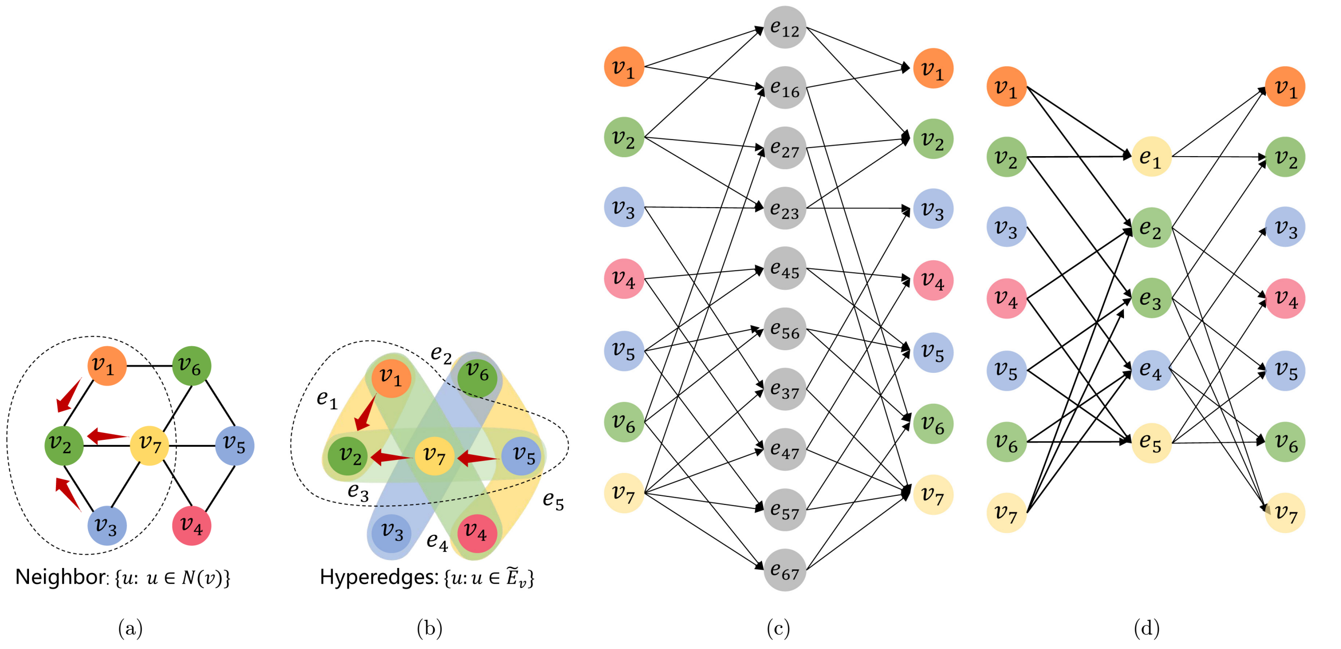

Neighbor: $\{u: u \in N(v)\}$
Hyperedges: $\{u: u \in \tilde{E}_v\}$
(a)
(b)
(c)
(d)

Fig. 1. (Color online) Illustrations of graph and hypergraph structures along with their corresponding message passing mechanisms. Different colors represent node labels. (a) Graph $G = (\mathcal{V}, \mathcal{E})$. (b) Hypergraph $H = (\mathcal{V}, \mathcal{E})$. (c) Message passing mechanism in a GNN: $h_{G,v} = \phi_1(h_{G,v}, \phi_2(\sum_{u\in N(v)} h_{G,u}))$, where $h_{G,v}$ is the feature vector of node v, $N(v)$ is the set of neighbors of v, and ϕ_1 and ϕ_2 are functions for processing messages and updating node features. (d) Message passing mechanism in a HGNN: $h_{H,v} = \phi_1(h_{H,v}, \phi_2(\sum_{u\in \tilde{E}_v} h_{H,u}))$, where $h_{H,v}$ is the feature vector of node v, $\tilde{E}_v$ is the hyperedges' set that connects node v, and ϕ_1 and ϕ_2 are the same as GNN.

the hypergraph convolutional layer is defined as

$$X^{(l+1)} = \sigma(D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} X \Theta), \tag{3}$$

where W represents the parameters of the neural network layer, Θ denotes learned parameters during training. The convolutional kernel Θ is applied to extract features from all nodes in the hypergraph. $X^{(l)} \in R^{n\times C}$ denotes the signal of the hypergraph at the lth layer, with $X^{(0)} = X$, and σ denotes a non-linear activation function. HGNN effectively extracts high-order correlations on the hypergraph through the node-edge-node transformation. However, it faces challenges when simplifying the hypergraph to a conventional graph in the clique expansion process, as it cannot handle substructures, such as hyperedges recursively containing other hyperedges, where internal hyperedges cannot be identified through clique expansion.

Dual-channel Hypergraph Collaborative Filtering (DHCF) algorithm[23] enhances the representation learning process for user and item structural information. By incorporating the results from the previous layer, DHCF accelerates model convergence. This dual-channel structure achieves collaborative filtering goals while maintaining user and item distinctions. However, the model's inter-object relationships rely solely on weight sharing, making connections highly dependent on data quality, posing challenges with noisy data. Based on the BTR algorithm[19] and the mediator algorithm,[20] HyperGCN[18] samples nodes and filters binary edges after unfolding the hypergraph, aiming to improve model learning effectiveness by filtering out potential noise. Although HyperGCN enhances the hypergraph Laplacian matrix by considering additional edge weights, it still struggles with preserving complex hypergraph information.

Hypergraph Networks with Hyperedge Neurons (HNHN)[24] proposes a hypergraph convolutional network that applies nonlinear activation functions to hypernodes and hyperedges, allowing flexibility to adjust the importance of high-cardinality hyperedges and high-order vertices. Initially, the model extracts neighborhood information from the hypergraph structure and computes normalization factors. Subsequently, in each layer of HNHN's hypergraph convolution, signals propagate from hypernodes to hyperedges. Similarly, signals are then transmitted back from hyperedges to hypernodes, utilizing node embeddings obtained through a multi-layer neural network. The model calculates loss via cross-entropy, predicting the difference between predicted labels and target labels. This model enables flexible information propagation and learning on the hypergraph, capturing complex relationships within the hypergraph through nonlinear activation functions and node-specific normalization.

Hypergraph Semi-supervised Aggregation (HyperSAGE)[25] avoids information loss by directly processing hypergraphs without converting them into ordinary graphs. Its two-stage information propagation mechanism is defined as

$$\begin{cases} h_{i,e} = M_1(\{x_j\}_{j\in N(i,e;\alpha)}), \\ \tilde{x_i} = W(x_i + M_2(\{h_{i,e}\}_{e\in E})), \end{cases} \tag{4}$$

where W represents a linear transformation, M_1 and M_2 are power mean functions, $N(i,e)$ denotes the neighbors of node i within hyperedge e, and $N(i,e,a)$ samples a nodes from $N(i,e)$. HyperSAGE model excels in hypergraph learning tasks. However, the original algorithm suffers from computational inefficiency in calculating hyperedge representations as it depends on different (i,e) pairs, leading to redundant nested loops. Additionally, both stages employ power mean functions which are not injective, resulting in it unable to distinguish isomorphic problems within the hypergraph.

In response to the issues in HyperSAGE, Unified Graph Neural Network (UniGNN)[26] modifies the two-stage message passing mechanism, where node embeddings are updated by aggregating neighbor embeddings, defined as

$$\begin{cases} h_{i,e} = \phi_1(\{x_j\}_{j\in e}), \\ \tilde{x_i} = \phi_2(x_i + \{h_e\}_{e\in E}), \end{cases} \tag{5}$$

where ϕ_1 and ϕ_2 are permutation-invariant functions. In the initial phase, ϕ_1 consolidates node attributes within every hyperedge e, and h_e signifies the attribute of an individual node in a typical graph. In the second stage, ϕ_2 updates individual nodes by aggregating the representations of hyperedges associated with them, which represents the set of directly connected neighbors in the communication graph. Meanwhile, UniGNN extends classic GNN models to hypergraphs by incorporating information weighting for nodes across different hyperedges in graph convolutional networks, leading to UniGCN and UniGAT, considering information embeddings of nodes and hyperedges. Furthermore, in the GraphSAGE model, UniGNN utilizes hyperedge weighting for connection aggregation based on different tasks. In deep convolutional networks, to overcome the issue of excessive smoothing during training, UniGCNII introduces initial residual connections and identity mapping, extending this concept to hypergraphs and the process can be defined as

$$\begin{cases} \hat{x}_i = \dfrac{1}{d_i}\displaystyle\sum_{e\in\tilde{E}_i}\dfrac{1}{d_e}h_e, \\ \tilde{x_i} = ((1-\beta)I + \beta W)((1-\alpha)\hat{x}_i + \alpha x_i^0), \end{cases} \tag{6}$$

where α and β denote hyperparameters, I represents for the identity matrix, W signifies the trainable parameters of the neural network, and x_i^0 is the initial feature vector of node i. In each layer, UniGCNII employs a two-stage aggregation similar to UniGCN, and then injects knowledge from previous layers to the current layer through skip connections.

3. Self-Hypergraph Graph Isomorphism Network

GNNs capture the structure of graphs through node embeddings, updating each node's feature vector by recursively aggregating features from adjacent nodes. This mechanism leverages the WL test to aggregate neighborhood information, characterizing different node labels and distinguishing non-isomorphic graphs, as shown in Fig. 1(c), where node v_2 aggregates label information from its neighbors $N(v_2) = \{v_1, v_3, v_7\}$. However, in HGNNs, the message-passing mechanism breaks the local neighborhood concept and instead updates via hyperedges that connect nodes, as illustrated in Fig. 1(d). Here, hyperedges e_1 and e_2 aggregate information from all nodes on the edge, and then node v_2 updates its information by aggregating the information from hyperedges e_1 and e_2. To effectively address the challenge of distinguishing non-isomorphic structures within the context of information propagation across nodes and hyperedges in hypergraphs,[27,28] HGNNs map different structural configurations to similar or identical embeddings. To mitigate this issue, we incorporate concepts from multiset theory to develop a theoretical framework for hypergraph WL test. Establishing the equivalence conditions between HGNN and the hypergraph WL test enables these methods to predict missing node labels accurately and achieve precise classification in hypergraphs that contain unlabeled nodes through the representation learning.

3.1. *Preliminaries*

Graph WL test[29,26] updates node attributes based on the degrees and properties of neighboring nodes in each iteration. Specifically, the label of node v is concatenated with the sorted labels of its neighbors, and then hashed to a new label. This process captures the local structure around each node and allows for the detection of structural differences between graphs.

Lemma 1. *Let $\mathcal{A} : \mathcal{G} \to \mathcal{R}^d$ be a GNN with k layers. If $\mathcal{A}$ maps two non-isomorphic graphs G_1 and G_2 such that $\mathcal{A}(G_1) \neq \mathcal{A}(G_2)$, then G_1 and G_2 are non-isomorphic according to the WL test.*

Theorem 1. *GNN $\mathcal{A} : \mathcal{G} \to R^d$, with k layers, $\mathcal{A}$ maps two graphs G_1 and G_2 that are WL tested to be non-isomorphic to different embeddings, if satisfied $\mathcal{A}$*

iteratively aggregates and updates node features

$$h_{G,v}^{(k)} = \phi_1(h_{G,v}^{(k-1)}, \phi_2(\{h_{G,u}^{(k-1)} : u \in N(v)\})), \tag{7}$$

where ϕ_1, ϕ_2 are injective functions. The graph-level readout function for $\mathcal{A}$ operates injective on the multiset of node embedding features.

3.2. *Theoretical framework*

In HGNNs, hyperedge aggregation replaces neighborhood aggregation, and the information transfer mechanism extended to hypergraphs can be defined as follows:

$$\begin{cases} h_{v,e} = \phi_2(\{x_u\}_{u\in e}), \\ \tilde{x}_v = \phi_1(x_v + \{h_{v,e}\}_{e\in E_v}), \end{cases} \tag{8}$$

where ϕ_1 and ϕ_2 are permutation invariant functions. In the initial phase, ϕ_2 merges the attributes of the nodes in each hyperedge e, and $h_{v,e}$ represents the attributes of a single node v in a typical graph. In the second stage, ϕ_1 updates individual nodes by aggregating hyperedge representations associated with them in Fig. 1(d). That is, this procedure aggregates the node features on all hyperedges e connecting node v considering high-order interactions. To enhance the learning capabilities of HGNNs, we extend the WL test[27,28] in graph isomorphism networks to the realm of hypergraphs.

Definition 3. Hypergraph $H(\mathcal{V}, \mathcal{E})$, $\forall v \in \mathcal{V}$, the label of node v is initialized to $l_{H,v}^{(0)} = 0$ at iteration 0 according to the WL test. The node labels are updated at iteration k as follows:

$$\begin{cases} l_{H,e}^{(k)} = \{\{l_{H,u}^{(k)}\}\}_{u\in e}, & \forall e \in \mathcal{E}, \\ l_{H,v}^{(k+1)} = \{\{l_{H,v}^{(k)}, l_{H,e}^{(k)}\}\}_{e\in E_v}, & \forall v \in \mathcal{V}, \end{cases} \tag{9}$$

where $\{\{\cdot\}\}$ denotes a multiset, and $l_{H,e}^{(k)}$ is the label of hyperedge $e \in \mathcal{E}$.

By contrast, hypergraph WL test updates node attributes based on the concatenation of node properties across all connected hyperedges as shown in Fig. 2. This approach considers a more complex pattern of connectivity, as it takes into account the information from all nodes participating in the hyperedges connected to node v. The labels are then hashed to new labels, allowing for the distinction of hypergraphs based on their structural characteristics and labels.

Lemma 2. *Let $E_{H,v}$ denote the set of hyperedges containing node v, and let $l_{H,v}^{(j)}$ and $h_{H,v}^{(j)}$ represent the label and feature vector of node v in hypergraph H at iteration j, respectively. If the conditions hold, for iterations* $0, 1, \ldots, k$,

$$\begin{cases} l_{H_1}^{(j)} = l_{H_2}^{(j)}, & \forall j \le k, \\ h_{H_1}^{(j)} = h_{H_2}^{(j)}, & \forall j \le k-1, \end{cases} \tag{10}$$

then, for nodes $v_1 \in H_1$ and $v_2 \in H_2$, if $l_{H_1,v_1}^{(k)} = l_{H_2,v_2}^{(k)}$, it follows that $h_{H_1,v_1}^{(k)} = h_{H_2,v_2}^{(k)}$ at iterations k.

Proof. At iteration $k = 0$, the condition holds. By induction, it follows that for all $j \le k-1$, $l_{H_1,v_1}^{(j)} = l_{H_2,v_2}^{(j)}$, which implies $h_{H_1,v_1}^{(j)} = h_{H_2,v_2}^{(j)}$. At iteration k, given $l_{H_1,v_1}^{(k)} = l_{H_2,v_2}^{(k)}$, we have

$$\begin{aligned} &\{\{(l_{H_1,v_1}^{(k-1)}, \{\{l_{H_1,u_1}^{(k-1)}\}\}_{u_1\in E_{H_1,v_1}})\}\} \\ &\quad = \{\{(l_{H_2,v_2}^{(k-1)}, \{\{l_{H_2,u_2}^{(k-1)}\}\}_{u_2\in E_{H_2,v_2}})\}\}. \end{aligned} \tag{11}$$

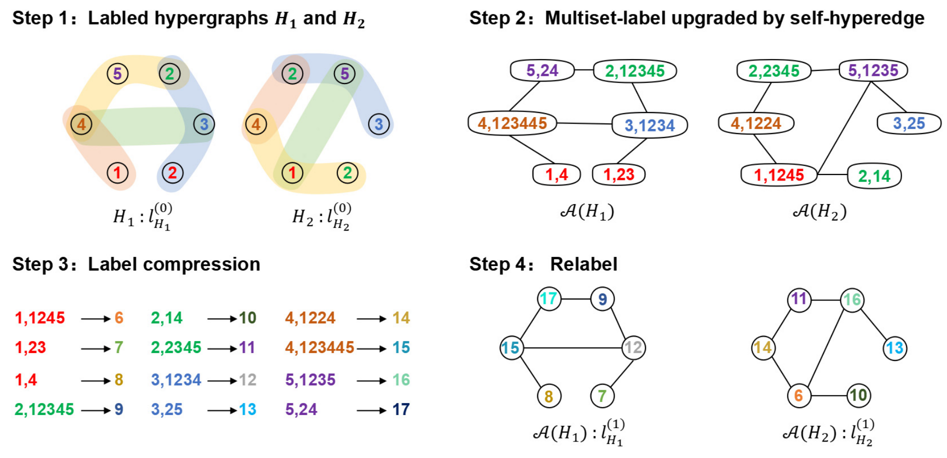


Fig. 2. Illustrations of the hypergraph WL test for distinguishing non-isomorphic hypergraphs H_1 and H_2. Step 1: Label the nodes of two hypergraphs based on the observed labels $l_{H_1}^{(0)} \to H_1$, $l_{H_2}^{(0)} \to H_2$. Step 2: Map $\mathcal{A} : \mathcal{H} \to \mathcal{R}^d$, update the multiset-label information for each node by aggregating according to the hyperedge relations. Step 3: Compress the multiset labels and assign them as new node labels. Step 4: Relabel the nodes with new labels $l_{H_1}^{(1)} \to \mathcal{A}(H_1), l_{H_2}^{(1)} \to \mathcal{A}(H_2)$.

By the inductive hypothesis, we obtain

$$\begin{aligned}&\{\{(h_{H_1,v_1}^{(k-1)}, \{\{h_{H_1,u_1}^{(k-1)}\}\}_{u_1\in E_{H_1,v_1}})\}\}\\ &\quad = \{\{(h_{H_2,v_2}^{(k-1)}, \{\{h_{H_2,u_2}^{(k-1)}\}\}_{u_2\in E_{H_2,v_2}})\}\}. \end{aligned} \tag{12}$$

Note that $\mathcal{A}$ is injective, and the HGNN function's aggregation function generates the same output $h_{H_1,v_1}^{(k))} = h_{H_2,v_2}^{(k)}$. □

Lemma 3. *Let $\mathcal{A}: \mathcal{H} \to \mathcal{R}^d$ be a HGNN with k layers. If $\mathcal{A}$ maps two non-isomorphic hypergraphs H_1 and H_2 such that $\mathcal{A}(H_1) \neq \mathcal{A}(H_2)$, then H_1 and H_2 are non-isomorphic decided by WL test.*

Proof. Suppose there exists $k \geq 0$ such that after k iterations, $\mathcal{A}(H_1) \neq \mathcal{A}(H_2)$, that is, $h_{H_1}^{(k)} \neq h_{H_2}^{(k)}$, but the graph WL test does not indicate that H_1 and H_2 are non-isomorphic, i.e., $l_{H_1}^{(k)} = l_{H_2}^{(k)}$.

For $k = 0$, then $l_{H_1}^{(0)} = l_{H_2}^{(0)}$ implies $h_{H_1}^{(0)} = h_{H_2}^{(0)}$. This means that both the WL test and $\mathcal{A}$ start with the same labels and features, which contradicts our assumption. For $k \geq 0$, then $l_{H_1}^{(k-1)} = l_{H_2}^{(k-1)}$ and $h_{H_1}^{(k-1)} = h_{H_2}^{(k-1)}$. Given that $l_{H_1}^{(k)} = l_{H_2}^{(k)}$, by Lemma 2, we obtain

$$l_{H_1,v_1}^{(k)} = l_{H_2,v_2}^{(k)}, \quad h_{H_1,v_1}^{(k)} = h_{H_2,v_2}^{(k)}. \tag{13}$$

This implies the existence of a mapping $\mathcal{B}$ such that $\mathcal{B}(l_{H,v}^{(k-1)}) \to h_{H,v}^{(k-1)}$. Since $l_{H_1}^{(k)} = l_{H_2}^{(k)}$, we have

$$\{\{l_{H_1,v_1}^{(k)}\}\}_{v_1\in V_1} = \{\{l_{H_2,v_2}^{(k)}\}\}_{v_2\in V_2}, \tag{14}$$

s.t.

$$\{\{\mathcal{B}(l_{H_1,v_1}^{(k)})\}\}_{v_1\in V_1} = \{\{\mathcal{B}(l_{H_2,v_2}^{(k)})\}\}_{v_2\in V_2}. \tag{15}$$

Consequently,

$$\begin{aligned}h_{H_1}^{(k)} &= \{\{h_{H_1,v_1}^{(k)}\}\}_{v_1\in V_1} = \{\{\mathcal{B}(l_{H_1,v_1}^{(k)})\}\}_{v_1\in V_1}\\ &= \{\{\mathcal{B}(l_{H_2,v_2}^{(k)})\}\}_{v_2\in V_2} = \{\{h_{H_2,v_2}^{(k)}\}\}_{v_2\in V_2} = h_{H_2}^{(k)}.\end{aligned} \tag{16}$$

This conclusion contradicts the original assumption, thus proving that the original conclusion holds. □

Theorem 2. *Let $\mathcal{A}: \mathcal{H} \to \mathcal{R}^d$ be a HGNN with k layers. If $\mathcal{A}$ maps two hypergraphs H_1 and H_2 that are determined to be non-isomorphic by the WL test to different embeddings, then $\mathcal{A}$ satisfies the following conditions*:

1. *Iterative Aggregation and Update of Node Features*:

$$h_{H,v}^{(k)} = \phi_1(\{\{(h_{H,v}^{(k-1)}, \phi_2(\{h_{H,u}^{(k-1)}\}_{u\in e}))\}\}_{e\in E_v}), \tag{17}$$

where ϕ_1 and ϕ_2 are injective functions.

2. *Injective Graph-Level Readout Function*: *The graph-level readout function for $\mathcal{A}$ operates injectively on the multiset of node embedding features.*

Proof. Suppose there exists an injective mapping $\mathcal{B}^{(k)}$ such that $\mathcal{B}^{(k)}(l_{H,v}^{(k)}) \to h_{H,v}^{(k)}$. For $k = 0$, $\mathcal{B}^{(0)}$ is the identity mapping in the case of $l_{H,v}^{(0)}$ and $h_{H,v}^{(0)}$ are the same. For $k \geq 0$, assuming the injective mapping $\mathcal{B}^{(k)}$ exists, we show that it holds at iterations $k+1$,

$$h_{H,v}^{(k+1)} = \phi_2(\{\{(h_{H,v}^{(k)}, \phi_1(\{\{h_{H,u}^{(k)}\}\}_{u\in e}))\}\}_{e\in E_v}). \tag{18}$$

Since the composition of injective functions ϕ_1 and ϕ_2 is also injective, we can rewrite this as

$$\begin{aligned}h_{H,v}^{(k+1)} &= \varphi(\{\{(h_{H,v}^{(k)}, \{\{h_{H,u}^{(k)}\}\}_{u\in e})\}\}_{e\in E_v})\\ &= \varphi(\{\{(\mathcal{B}^{(k)}(l_{H,v}^{(k)}), \{\{\mathcal{B}^{(k)}(l_{H,u}^{(k)})\}\}_{u\in e})\}\}_{e\in E_v})\\ &= \mathcal{B}^{(k+1)}(\{\{(l_{H,v}^{(k)}, \{\{l_{H,u}^{(k)}\}\}_{u\in e})\}\}_{e\in E_v})\\ &= \mathcal{B}^{(k+1)}(l_{H,v}^{(k+1)}).\end{aligned} \tag{19}$$

Here, φ is the injective function induced by ϕ_1 and ϕ_2, and $\mathcal{B}^{(k+1)}$ is the injective function induced by φ and $\mathcal{B}^{(k)}$. Thus, by induction, there also exists a injective mapping $\mathcal{B}^{(k)}$ such that $\mathcal{B}^{(k)}(l_{H,v}^{(k)}) \to h_{H,v}^{(k)}$. Consequently, if at iterations k, $l_{H_1}^{(k)} \neq l_{H_2}^{(k)}$, then $\{\{l_{H_1,v_1}^{(k)}\}\}_{v_1\in V_1} \neq \{\{l_{H_2,v_2}^{(k)}\}\}_{v_2\in V_2}$. Therefore, by the injectivity, we can derive similarly that

$$\begin{aligned}h_{H_1}^{(k)} &= \{\{l_{H_1,v_1}^{(k)}\}\}_{v_1\in V_1} = \{\{\mathcal{B}^{(k)}(l_{H_1,v_1}^{(k)})\}\}_{v_1\in V_1}\\ &\neq \{\{\mathcal{B}^{(k)}(l_{H_2,v_2}^{(k)})\}\}_{v_2\in V_2} = \{\{l_{H_2,v_2}^{(k)}\}\}_{v_2\in V_2} = h_{H_2}^{(k)}.\end{aligned} \tag{20}$$

□

3.3. *SHGIN algorithm*

To leverage the powerful expressive capabilities of the WL test on hypergraphs, which employs an injective aggregation update process, the incorporation of node-hyperedge structures is utilized to enhance the capture of relationships among nodes in hypergraphs, resulting in improved model performance. In current HGNN learning methods, the construction of hyperedges typically relies on node neighborhoods, with hyperedge sizes often being $|e| \geq 2$. However, the features of nodes themselves within a hypergraph influence the flow of "node-hyperedge-node" information during the training of neural networks. Analogously, in GNN training, directly combining node features is a common strategy to address challenges such as gradient issues. Consequently, the hypergraph is preprocessed by introducing self-loop to consider the information and self-dependence of nodes. This involves assigning each node $v_i \in V$ to a hyperedge $e = \{v_i\} \in \tilde{E}$,

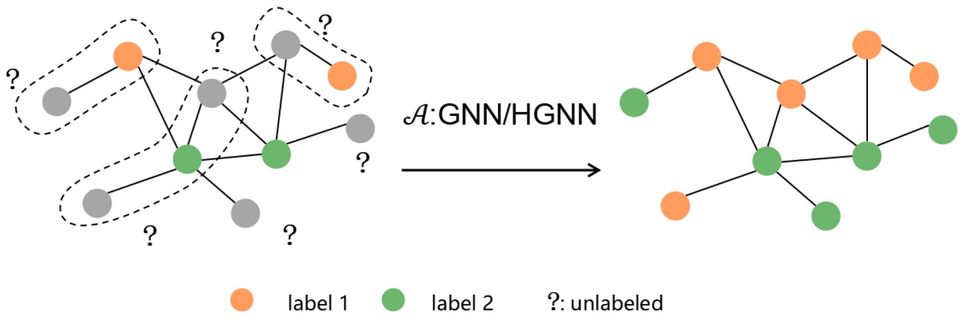


Fig. 3. Illustration of the node classification task. Mapping function $\mathcal{A}$: GNN/HGNN learns the entire structure and incomplete labels of the observed nodes and predicts unobserved labels of nodes.

effectively treating each node as a hyperedge, allowing for the integration of its own information. Inspired by the Graph Isomorphism Network (GIN),[30] the framework of the Self-hypergraph Graph Isomorphism Network consists of the two-phase message passing mechanism process on hyperedges, as follows:

Step 1: Aggregate hyperedges' information from connected nodes:

$$h_{H,e}^{(k-1)} = \sum_{u\in\Gamma(e)} h_{H,u}^{(k-1)}, \tag{21}$$

where $\Gamma(e)$ denotes the set of nodes connected by hyperedges e.

Step 2: Update node feature using hyperedges aggregation:

$$h_{H,v}^{(k)} = \mathrm{MLP}^{(k)}\left((1+\epsilon^{(k)})h_{H,v}^{(k-1)} + \sum_{e\in\tilde{E}_v} h_{H,e}^{(k-1)}\right), \tag{22}$$

where $\epsilon^{(k)}$ is a training parameter, $h_{H,v}^{(k)}$ denotes the representation of node v at layer k, $h_{H,v}^{(k-1)}$ is the representation at the previous layer, $\tilde{E}_v$ represents the subset of edges related to node v, including the hyperedge formed by the node itself, and $h_{H,e}^{(k-1)}$ is the representation of hyperedge e at the previous layer.

Equivalently, this can be defined as

$$\tilde{x}_v = W\left((1+\epsilon)x_v + \sum_{e\in\tilde{E}_v} h_e\right), \tag{23}$$

where x_v represents the input signal of node v, and W is a weight matrix used to transform the aggregated node and hyperedge representations. Unlike the neighborhood aggregation process in GNNs, where information is aggregated from immediate neighboring nodes, the SHGIN model leverages higher-order interactions to aggregate node features over the entirety of hyperedge e. This approach allows for a comprehensive integration of information from multiple nodes simultaneously, capturing richer structural dependencies.

3.4. *Hypernode classification*

In the task of semi-supervised node classification on hypergraphs, the labels y_v of some nodes in the hypergraph can be observed, and some nodes are unlabeled as shown in Fig. 3. SHGIN model aggregates node information under high-order relationships through hyperedges to predict missing labels of nodes. The objective function can be defined as

$$\min R_{\text{emp}}(\mathcal{G}) + \lambda\Omega(\mathcal{G}), \tag{24}$$

where $\mathcal{G}$ denotes the predicted node embeddings or class scores $\hat{y} = \mathcal{G}(H, X)$ generated by the hypergraph node classification model, $R_{\text{emp}}(\mathcal{G})$ represents the empirical supervised loss, denotes $\frac{1}{|V_{\text{train}}|}\sum_{v\in V_{\text{train}}}\mathrm{CrossEntropy}(\hat{y}_v, y_v)$, and λ denotes regularization parameter, $\Omega(\mathcal{G})$ represents the penalty term for the normalization process. Specifically, the definition of $\Omega(\mathcal{G})$ is given by $\Omega(\mathcal{G}) = \mathcal{G}^T\Delta$, where Δ is the hypergraph Laplacian operator defined as $\Delta = I - D_v^{-1/2}HD_e^{-1}H^TD_v^{-1/2}$, where $H \in \mathbb{R}^{n\times m}$ represents the hypergraph adjacency matrix, $D_v \in \mathbb{R}^{n\times n}$ is the degree matrix of the hypergraph nodes, and $D_e \in \mathbb{R}^{m\times m}$ is the degree matrix of the hyperedge e. To illustrate how the SHGIN model performs on the node classification task through the hypergraph WL test, the detailed process is presented in Algorithm 1.

Algorithm 1. Self-hypergraph Graph Isomorphism Network (SHGIN) with WL Test for Node Classification Task

```
1: Input: Hypergraph $H = (V, E)$, features $h_{H,v}$, labels $y_{H,v}$, iterations number $T$, training/test/validation set $V_{\text{train}}$, $V_{\text{test}}$,$V_{\text{val}}$, learning rate $\eta$, epochs number $N_{\text{epochs}}$
2: Output: Node embeddings and test accuracy
3: Initialization:
4: for each node $v \in V$ do
5:     Initialize node feature $h^0_{H,v}$ input feature matrix or uniform initialization
6:     Initialize labels $l^0_{H,v}$ to $y_v$ ▷ For labeled nodes in training set
7:     Add self-hyperedge $e_v = (v, v)$ to $E$ ▷ Capture self-dependencies
8: end for
9: Initialize model $\mathcal{G}$ as an multilayer perceptron and Adam optimizer
10: Set stopping criteria: max number of epochs $N_{\text{epochs}}$, patience for early stopping $p$, and loss threshold $\varepsilon$
11: for epoch = 1 to $N_{\text{epochs}}$ do
12:     for $t = 1$ to $T$ do ▷ Run T iterations of the WL test during each epoch
13:         ▷ Hypergraph WL Test
14:         for each node $v \in V$ do
15:             Aggregate hyperedges' features from nodes connecting to hyperedges:
                $$h^{(k-1)}_{H,e} = \sum_{u \in \Gamma(e)} h^{(k-1)}_{H,u}, \forall u \in e$$
                ▷ Aggregation as Equation.(21)
16:             Update node $v$ feature using hyperedges aggregation
                $$h^{(k)}_{H,v} = \text{MLP}^{(k)}\left((1+\epsilon^{(k)})h^{(k-1)}_{H,v} + \sum_{e \in \tilde{E}_v} h^{(k-1)}_{H,e}\right),$$
                ▷ Process as Eq. (22), where $\epsilon^{(k)}$ denotes training parameter
17:         end for
18:     end for
19:     ▷ Loss computation and backpropagation
20:     Forward pass: $\hat{y} = \mathcal{G}(H, X)$
21:     Combine loss terms: empirical loss and regularization ▷ as Eq. (24)
        $$\mathcal{L}_{\text{train}} = \frac{1}{|V_{\text{train}}|} \sum_{v \in V_{\text{train}}} \text{CrossEntropy}(\hat{y}_v, y_v) + \lambda\Omega(\mathcal{G}),$$
22:     Backpropagate loss: $\mathcal{L}_{\text{train}}$.backward()
23:     Update model parameters: optimizer.step()
24:     ▷ Early stopping criteria based on validation loss
25:     Compute validation loss $\mathcal{L}_{\text{val}}$ on $V_{\text{val}}$
26:     if $\mathcal{L}_{\text{val}}$ does not improve for $p$ epochs or $\mathcal{L}_{\text{val}} < \varepsilon$ then
27:         Early stopping triggered
28:         break
29:     end if
30: end for
31: Evaluate the SHGIN model on the test node set $V_{\text{test}}$
32: Return Node embeddings $h_{H,v}$ and test accuracy
```

4. Experiments

Datasets. To demonstrate the effectiveness of high-order neural network models and evaluate the advantages and limitations of various approaches. Academic co-citation network and co-authorship network datasets are considered for testing node classification tasks in networks of cooperation that take into account high-order relationships in Table 1. The co-citation network datasets consist of Cora ($n = 2708$, $m = 5429$),[31] Citeseer ($n = 3327$, $m = 9464$), and Pubmed ($n = 19,717$, $m = 88,676$),[32] their hyperedges represent all documents referenced by the authors, and co-authorship networks, such as Cora ($n = 2708$, $m = 5429$)[31] and DBLP ($n = 4095$, $m = 14,328$),[33] its hyperedges connect all documents co-authored by an author.

Implementation details: HGNN models are trained using the cross-entropy loss function, utilizing the Adam optimizer. The batch size is 64, and training is terminated prematurely at 200 epochs if no loss reduction is detected on the validation set, with a maximum of 300 epochs allowed. Train for 300 epochs, repeat the node classification task three times by selecting models from the validation results within each epoch, and calculate the mean and variance of the results.

Results. The results indicate that the SHGIN model, leveraging the information of self-loop edges formed by individual nodes, outperforms traditional GNN and HGNN models in terms of classification accuracy. As shown in Table 2, in the co-authorship network datasets, on Cora, SHGIN displays superior accuracy of 76.6%, better classifying the attributes of authors within co-authorship networks, and outperforming other High-order neural network models significantly. On the DBLP dataset, SHGIN achieves an accuracy of 89.1%, outperforming models such as UniGIN and UniSAGE, which have accuracies of 88.5% and 88.6%, respectively.

As illustrated in Table 3, in the co-citation networks, UniGCN achieves the highest performance with an

Table 1. Details of datasets with high-order interactions.

Dataset		Nodes	Edges	Features	High-order interactions
Co-authorship	Cora	2708	5429	1433	Co-authored papers forming group relationships
	DBLP	31,546	115,515	334	Multi-author collaborations in academic circles
Co-citation	Cora	2708	5255	1433	Papers co-cited, revealing related topics
	Citeseer	3327	4732	3703	Co-cited papers highlighting related research
	Pubmed	19,717	44,338	500	Co-citations forming cohesive themes

Table 2. Accuracy (%) of semi-supervised hypernode classification of hypergraph models in co-authorship datasets.

		Co-authorship	
Category	Methods	Cora	DBLP
GNN*	GCN*	70.3 ± 4.75	88.2 ± 0.57
	GraphSAGE*	71.8 ± 3.59	87.3 ± 0.31
	GIN*	61.2 ± 2.93	76.5 ± 0.54
	MAGAE*	75.6 ± 0.41	87.9 ± 0.62
	Graphite*	74.8 ± 0.31	88.1 ± 0.73
HGNN	HGNN	59.2 ± 3.19	76.4 ± 0.39
	HGNN $^+$	63.2 ± 2.73	77.1 ± 0.57
	HyperGCN	54.5 ± 1.17	75.9 ± 0.56
	HyperSAGE	68.4 ± 1.39	78.1 ± 0.42
	HNHN	69.3 ± 2.13	85.1 ± 0.49
	HGCN	73.6 ± 1.02	86.9 ± 0.73
	UniGCN	69.9 ± 0.55	87.5 ± 0.41
	UniGCNII	68.7 ± 0.91	88.4 ± 0.78
	UniGAT	71.1 ± 0.78	88.4 ± 0.57
	UniSAGE	73.9 ± 1.19	88.6 ± 0.81
	UniGIN	74.3 ± 1.29	88.5 ± 0.14
	SHGIN (ours)	**76.6 ± 1.12**	**89.1 ± 0.27**

Notes: The best or competitive results are highlighted below. All experiments are implemented in PyTorch and executed on NVIDIA A6000 GPUs.

Table 3. Accuracy (%) of semi-supervised hypernode classification of hypergraph models in co-citation datasets.

		Co-citation		
Category	Methods	Cora	Citeseer	PubMed
GNN*	GCN*	82.3 ± 0.78	71.7 ± 0.65	77.4 ± 0.91
	GraphSAGE*	80.9 ± 0.23	70.6 ± 0.29	76.4 ± 0.75
	GIN*	57.1 ± 0.48	53.8 ± 0.37	74.7 ± 0.41
	MTGAE*	79.0 ± 0.32	71.8 ± 0.12	77.5 ± 0.34
	Graphite*	82.1 ± 0.06	71.0 ± 0.07	78.3 ± 0.03
HGNN	HGNN	81.9 ± 0.24	54.7 ± 0.52	78.3 ± 0.57
	HGNN $^+$	80.6 ± 0.36	60.6 ± 0.47	**78.5 ± 0.30**
	HyperGCN	70.5 ± 0.42	62.7 ± 0.61	75.9 ± 0.43
	HyperSAGE	71.3 ± 0.12	69.3 ± 0.19	78.5 ± 0.61
	HNHN	51.8 ± 0.28	64.8 ± 0.35	40.7 ± 0.13
	HGCN	79.9 ± 0.43	71.9 ± 0.18	78.3 ± 0.49
	UniGCN	**82.7 ± 0.59**	71.2 ± 0.47	78.1 ± 0.57
	UniGCNII	76.3 ± 0.74	68.9 ± 0.69	72.6 ± 1.03
	UniGAT	81.8 ± 0.85	70.9 ± 0.68	78.2 ± 0.48
	UniSAGE	79.4 ± 1.49	71.1 ± 0.94	78.1 ± 0.93
	UniGIN	78.1 ± 1.93	72.5 ± 0.86	77.3 ± 0.74
	SHGIN(ours)	79.6 ± 1.21	**72.7 ± 0.13**	78.4 ± 0.47

Notes: The best or competitive results are highlighted below. All experiments are implemented in PyTorch and executed on NVIDIA A6000 GPUs.

accuracy of 82.7% on the Cora dataset. Compared to UniGIN, our model demonstrates a slight improvement in classification accuracy, reaching 79.6%, while the classification accuracy of the UniGIN model was 78.1%, and the GCN model has a slightly lower accuracy of 82.3%. On the Citeseer dataset, SHGIN exhibits an accuracy of 72.7%, surpassing other baseline models and exceeding the 72.5% accuracy of UniGIN. On the PubMed dataset, our model outperforms most High-order neural network models with an accuracy of 78.3%. The HGNN and HGNN$^+$ models also demonstrate strong classification performance, achieving accuracies of 78.3% and 78.8%, respectively. These results highlight that our model outperforms traditional GNN and high-order neural network models.

For the hypergraph node classification task, the two-stage information propagation process of model SHGIN, which aggregates feature information on hyperedges (self-hyperedge) and hypernodes, demonstrates stable performance with good accuracy on the hypergraph. This performance surpasses traditional GNN models. The high-order neural network model allows for the investigation of intricate high-order connections within cooperative networks, facilitating accurate node classification even in scenarios, where node labels are not explicitly distinguished in the hypergraph. Meanwhile, the experimental results demonstrate that extending graph isomorphism network to hypergraphs enables more flexible and effective aggregation of interaction information on hyperedges, capturing richer structural features. This approach distinguishes the hyperedge structures of isomorphic hypergraphs, enhancing the model's representation learning for nodes within hyperedges, and yielding better performance and generalizability.

5. Conclusion

In the study of different cooperative behaviors of individuals within complex systems, analyzing high-order cooperative relationships can better capture the characteristics of network nodes, such as in co-citation networks and co-authorship networks. Traditional neural network models struggle to explore latent high-order cooperative relationships within these networks, whereas

HGNN models exhibit satisfactory learning capabilities. Inspired by high-order neural network methods, we consider that graph isomorphism networks can be extended to hypergraphs and possess strong representation learning abilities. We propose SHGIN model, provide an explicit mathematical framework to describe the information transmission mechanism of high-order interactions and establish the equivalent conditions for HGNNs to the WL test. Additionally, the upper bound of the expressibility of HGNN is also demonstrated through theoretical analysis. Compared with the classification accuracy of HGNN models on the co-authorship networks, SHGIN outperforms other baseline models. Meanwhile, HGNN models achieve considerable accuracy on high-order networks, comparable to GNN models on general graphs.

In addition, traditional graph structures are limited in capturing higher-order interactions within heterogeneous data. Our model addresses these limitations by effectively learning high-order interactions and distinguishing non-isomorphic hypergraphs using the hypergraph WL test. This capability is crucial for comprehensively analyzing indirect influences and interactions among multiple intelligent communities, understanding individual behaviors, and predicting trends in collaboration and evolution. Recent advancements in high-order graphs have been applied to satellite cooperation and communication. For example, the multi-aspect expanded hypergraph[34] captures the roles of satellite communication and inter-domain resources by representing hyperedges with similar characteristics across different aspects, thereby reducing redundant connections. It models the multi-domain resource allocation problem as a mixed integer linear programming problem to maximize task completion rates. In the communication domain, hypergraphs have been used to address quality of transmission constraints in multi-lightpath communications.[35] It captures the coupling effects due to stimulated Raman scattering, enabling more accurate and efficient parallel virtual link mapping, thereby alleviating the long-range signal blocking problem in next-generation communications. In future work, our work can be further extended to apply the challenges of navigation control and swarm evolution, such as efficient communication patterns of base stations, heterogeneous unmanned swarm formation combination problems.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2022ZD0120004, in part by Zhishan Youth Scholar Program, in part by the National Natural Science Foundation of China under Grant 62233004, Grant 62273090, and Grant 62073076, and in part by the Jiangsu Provincial Scientific Research Center of Applied Mathematics under Grant BK20233002.

ORCID

Jiawen Chen https://orcid.org/0009-0003-0530-0387
Yanyan He https://orcid.org/0009-0006-0860-7992
Duxin Chen https://orcid.org/0000-0002-3194-2258
Wenwu Yu https://orcid.org/0000-0003-0301-9180

References

1. Manlio De Domenico, More is Different in Real-world Multilayer Networks, *Nat. Phys.* **19**(9), 1247–1262, (2023), doi: org/10.1038/s41567-023-02132-1.
2. Wenjian Luo, Daofu Zhang, Li Ni *et al.*, Multiscale Local Community Detection in Social Networks, *IEEE Trans. Knowl. Data Eng.* **33**(3), 1102–1112 (2021), doi: 10.1109/TKDE.2019.2938173.
3. Renaud Lambiotte, Martin Rosvall and Ingo Scholtes, From Networks to Optimal Higher-order Models of Complex Systems, *Nat. Phys.* **15**(4), 313–320 (2019), doi: org/10.1038/s41567-019-0459-y.
4. Federico Battiston, Enrico Amico, Alain Barrat *et al.*, The Physics of Higher-order Interactions in Complex Systems, *Nat. Phys.* **17**(10), 1093–1098 (2021), doi: org/10.1038/s41567-021-01371-4.
5. Nasimeh Heydaribeni, Xinrui Zhan, Ruisi Zhang *et al.*, Distributed Constrained Combinatorial Optimization Leveraging Hypergraph Neural Networks, *Nat. Mach. Intell.* 1–9 (2024), doi: org/10.1038/s42256-024-00833-7.
6. Yujuan Wang and Yongduan Song, Leader-following Control of High-order Multi-agent Systems Under Directed Graphs: Pre-specified Finite Time Approach, *Automatica* **87**, 113–120 (2018), doi:doi: org/10.1016/j.automatica.2017.09.017.
7. Austin R. Benson, David F. Gleich and Jure Leskovec, Higher-order Organization of Complex Networks, *Science* **353**, 163–166 (2016), doi: 10.1126/science.aad9029.
8. Xiaoyi Liu, Duxin Chen, Wenjia Wei *et al.*, Interpretable Sparse System Identification: Beyond Recent Deep Learning Techniques on Time-series Prediction, in *International Conference on Learning Representations* (2024), https:/openreview.net/forum?id=aFWUY3E7ws.
9. Lingfei Wu, Peng Cui, Jian Pei *et al.*, Graph Neural Networks: Foundation, Frontiers and Applications, in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (2022), pp. 4840–4841, doi: org/10.1145/3534678.3542609.
10. Mingyu Kang, Ran Zhu, Duxin Chen *et al.*, CM-GAN: A Cross-modal Generative Adversarial Network for Imputing Completely Missing Data in Digital Industry, *IEEE Trans. Neural Netw. Learn. Syst.* **35**(3), 2917–2926 (2024), doi: 10.1109/TNNLS.2023.3284666.

11. Thomas N. Kipf and Max Welling, Variational Graph Auto-encoders (2016), doi:10.48550/arXiv.1611.07308.
12. Mingyu Kang, Ran Zhu, Duxin Chen *et al.*, A Cross-modal Generative Adversarial Network for Scenarios Generation of Renewable Energy, *IEEE Trans. Power Syst.* **39**(2), 2630–2640 (2023), doi: 10.1109/TPWRS.2023.3277698.
13. Thomas N. Kipf and Max Welling, Semi-supervised Classification with Graph Convolutional Networks (2016), doi:10.48550/arXiv.1609.02907.
14. Giannis Nikolentzos, Antoine Tixier and Michalis Vazirgiannis, Message Passing Attention Networks for Document Understanding, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34(05) (2020), pp. 8544–8551, doi: 10.1609/aaai.v34i05.6376.
15. Petar Veličković, Guillem Cucurull, Arantxa Casanova *et al.*, A. Romero, P. Lio and Y. Bengio, Graph Attention Networks (2017), doi:10.48550/arXiv.1710.10903.
16. Will Hamilton, Zhitao Ying and Jure Leskovec, Inductive Representation Learning on Large Graphs, *Adv. Neural Inf. Process. Syst.* **30** (2017), doi: 10.5555/3294771.3294869.
17. Yifan Feng, Haoxuan You, Zizhao Zhang *et al.*, Hypergraph Neural Networks, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33(01) (2019), pp. 3558–3565, doi: 10.1609/aaai.v33i01.33013558.
18. Naganand Yadati, Madhav Nimishakavi, Prateek Yadav *et al.*, HyperGCN: A New Method for Training Graph Convolutional Networks on Hypergraphs, *Adv. Neural Inf. Process. Syst.* **32** (2019), doi: 10.5555/3454287.3454422.
19. Anand Louis, Hypergraph Markov Operators, Eigenvalues and Approximation Algorithms, in *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing* (2015), pp. 713–722, doi: 10.1145/2746539.2746555.
20. T. H. Hubert Chan and Zhibin Liang, Generalizing the Hypergraph Laplacian Via a Diffusion Process with Mediators, *Theor. Comput. Sci.* **806**, 416–428 (2020), doi: 10.1016/j.tcs.2019.07.024.
21. Ke Tu, Peng Cui, Xiao Wang, Fei Wang and Wenwu Zhu, Structural Deep Embedding for Hyper-networks, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30(01) (2018), pp. 426–433, doi: 10.1609/aaai.v32i1.11266.
22. Ruochi Zhang, Yuesong Zou and Jian Ma, Hyper-SAGNN: A Self-attention Based Graph Neural Network for Hypergraphs (2019), doi:10.48550/arXiv.1911.02613.
23. S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang and Y. Gao, Dual Channel Hypergraph Collaborative Filtering, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2020), pp. 2020–2029, doi: 10.1145/3394486.3403253.
24. Yihe Dong, Will Sawin and Yoshua Bengio, HNHN: Hypergraph Networks with Hyperedge Neurons (2020), doi:10.48550/arXiv.2006.12278.
25. Devanshu Arya, Deepak K. Gupta, Stevan Rudinac *et al.*, HyperSAGE: Generalizing Inductive Representation Learning on Hypergraphs (2020), doi:10.48550/arXiv.2010.04558.
26. Jing Huang and Jie Yang, UniGNN: A Unified Framework for Graph and Hypergraph Neural Networks (2021), doi:10.48550/arXiv.2105.00956.
27. Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou *et al.*, Improving Graph Neural Network Expressivity Via Subgraph Isomorphism Counting, *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(01), 657–668 (2022), doi: 10.1109/TPAMI.2022.3154319.
28. Yifan Feng, Jiashu Han, Shihui Ying and Yue Gao, Hypergraph Isomorphism Computation, *IEEE Trans. Pattern Anal. Mach. Intell.* (2024), doi: 10.1109/TPAMI.2024.3353199.
29. László Babai, Graph Isomorphism in Quasipolynomial Time, in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing* (2016), pp. 684–697, doi: 10.1145/2897518.2897542.
30. Keyulu Xu, Weihua Hu, Jure Leskovec and Stefanie Jegelka, How Powerful are Graph Neural Networks? (2018), doi:10.48550/arXiv.1810.00826.
31. Prithviraj Sen, Galileo Namata, Mustafa Bilgic *et al.*, Collective Classification in Network Data Articles, *AI Mag.* **29**(3), 93–106 (2008), doi:10.1609/aimag.v29i3.2157.
32. Jérôme Kunegis, KONECT: The Koblenz Network Collection, in *International Conference on World Wide Web Companion* (2013), doi: 10.1145/2487788.2488173.
33. Jie Tang, Jing Zhang, Limin Yao *et al.*, ArnetMiner: Extraction and Mining of Academic Social Networks, in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008), pp. 990–998, doi: 10.1145/1401890.1402008.
34. Qi Hao, Min Sheng, Di Zhou and Yan Shi, A Multi-aspect Expanded Hypergraph Enabled Cross-domain Resource Management in Satellite Networks, *IEEE Trans. Commun.* **70**(7), 4687–4701 (2022), doi: 10.1109/TCOMM.2022.3174886.
35. Zeyuan Yang, Rentao Gu and Yuefeng Ji, Virtual Network Embedding Over Multi-band Elastic Optical Network Based on Cross-matching Mechanism and Hypergraph Theory, *IEEE Trans. Netw. Service Manag.* **20**(4), 4681–4697 (2023), doi: 10.1109/TNSM.2023.3259391.

Jiawen Chen received his Bachelor's degree in Information and Computing Science from the School of Mathematics, Southeast University, Nanjing, China, in 2022. He is currently pursuing Ph.D. at the School of Mathematics, Southeast University. His research focuses on complex networks, complex systems, graph representation learning, and higher-order neural networks.

Yanyan He received his Bachelor's degree in Mathematics from Shandong University of Science and Technology, Qingdao, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering at Southeast University, Nanjing, China. His research interests include graph representation learning, causal discovery, and sparse system identification.

Duxin Chen (Member, IEEE) received his B.S. degree in Automatic Control and Ph.D. degree in Control Science and Engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2013 and 2018, respectively. He is currently Associate Professor with the School of Mathematics, and Vice Director of the Jiangsu Provincial Scientific Research Center of Applied Mathematics, Southeast University, Nanjing, China. He is also one of the direction leaders of the Huawei-SEU Joint Innovation Lab of Networked Collective Intelligence. His research interests include causal inference, pre diction/generation, and system identification techniques for complex networks and systems science, artificial intelligence-related theory and applications.

Wenwu Yu (S07-M12-SM15, IEEE) received his B.Sc. degree in Information and Computing Science and M.Sc. degree in Applied Mathematics from the Department of Mathematics, Southeast University, Nanjing, China, in 2004 and 2007, respectively, and Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China, in 2010. Currently, he is Dean in the School of Mathematics, Deputy Associate Director of National Center of Applied Mathematics in Southeast University of Jiangsu, the Deputy Director of Jiangsu Provincial Scientific Research Center of Applied Mathematics, Deputy Associate Director of Jiangsu Provincial Key Laboratory of Networked Collective Intelligence, and Full Professor with the Endowed Chair Honor in Southeast University, China. Dr. Yu held several visiting positions in Australia, China, Germany, Italy, the Netherlands, and the USA. His research interests include multi-agent systems, complex networks and systems, disturbance control, distributed optimization, machine learning, game theory, cyberspace security, smart grids, intelligent transportation systems, big-data analysis, etc.

Dr. Yu serves as an Editorial Board Member of several flag journals, including IEEE Transactions on Circuits and Systems II, IEEE Transactions on Industrial Cyber-Physical Systems, IEEE Transactions on Industrial Informatics, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Science China Information Sciences, Science China Technological Sciences, etc. He was listed by Clarivate Analytics/Thomson Reuters Highly Cited Researchers in Engineering in 2014–2023. He published about 100 IEEE Transactions journal papers with more than 20,000 citations. Moreover, Dr. Yu is also the recipient of the Second Prize of State Natural Science Award of China in 2016. He is also the Cheung Kong Scholars Programmer of Ministry of Education of China (Artificial Intelligence).