# Reinterpreting Hypergraph Kernels: Insights Through Homomorphism Analysis

Yifan Zhang, Shaoyi Du, Yifan Feng, Shihui Ying, *Member, IEEE*, and Yue Gao, *Senior Member, IEEE*

***Abstract*—Designing expressive hypergraph kernels that can effectively capture high-order structural information is a fundamental challenge in hypergraph learning. In this paper, we propose a novel comparison framework based on hypergraph homomorphisms to evaluate and compare the expressive ability of existing hypergraph kernels. We revisit classical kernels such as Hypergraph Weisfeiler-Lehman (HG WL) and Hypergraph Rooted kernels, providing theoretical conditions under which they fail to distinguish non-isomorphic hypergraphs. Motivated by these insights, we introduce the Hypergraph Subtree-Cycle Kernel, which augments subtree-based features with cycle-based structural patterns to enhance expressiveness. We propose two variants: HG SCKernelv1 and HG SCKernelv2. Extensive experiments on five graph and ten hypergraph classification benchmarks demonstrate the superior performance of our methods, confirming the effectiveness of integrating homomorphism-guided design into hypergraph kernels.**

***Index Terms*—Hypergraph kernel, expressive ability, hypergraph homomorphism, Weisfeiler-Lehman, cycle modeling.**

## I. Introduction

HYPERGRAPH is the generalization of the graph, where each hyperedge has the flexibility to connect any number of vertices. This unique characteristic allows hypergraphs to capture and model complex relationships and interactions that are often difficult to represent via simple graphs. Hypergraph computation has been applied in multiple fields, including bioinformatics [1], [2], image processing [3], [4], [5], and community network analysis [6], [7] in recent years. The growing interest in hypergraphs suggests a promising future for their application in solving complex problems in various disciplines.

Hypergraph structures are ubiquitous in daily life, and exacting structural information is a fundamental task in pattern recognition. Kernel-based methods, such as Weisferiler-Lehman [8], Shortest Path [9], offer both interpretability and strong performance in structure-level classification and clustering tasks. Moreover, with the deepening of graph theory research, classical concepts such as Lovasz-$\vartheta$ function [10], and persistent homology [11] have been introduced to enrich the capacity to extract structural information from graphs. Meanwhile, advanced (hyper)graph embedding and neural network methods can be viewed as extensions of (hyper)graph kernels. However, because of the inherent complexity of hypergraph structures, explorations of hypergraph kernels remain relatively scarce. The researchers originally addressed similarity issues by projecting hypergraphs onto ordinary graphs and then applying graph kernel methods [12], [13], [14], [15]. For example, Bai et al. [12] propose a Directed Line graph conversion algorithm through the Perron-Frobenius operator, enabling Weisfeiler-Lehman (WL) method to efficiently address hypergraph isomorphism problems. However, these methods cannot directly handle hypergraph structures, and the conversion from hypergraphs to graphs not only increases the computational complexity in terms of time and space, but also results in the loss of high-order correlation information. Wachman et al. [16] propose a kernel on ordered hypergraph and introduce the concept of a root. Starting from the root, Hypergraph Rooted kernel gradually expands adjacent vertices, recording the labels of hyperedges along the traversal path and the positions of the visited vertices. However, this approach has two fundamental limitations: it essentially employs a random walk sampling strategy, with a great deal of randomness; the kernel's heavily dependent on root-vertex definitions and strict ordering constraints significantly limits its scalability. Jan Boker [17] defines hypergraph color refinement and also points out that in hypergraphs, the adjacency of vertices depends not only on their neighbors but also on the way they are connected. Lee et al. [18] first define hypergraph motifs, begins by analyzing 26 cases corresponding to three connected hyperedges, and count them in any specific hypergraph. However, MoCHy merely counts a predefined set of motifs, and cannot adapt to novel structural patterns. Feng et al. [19] propose Hypergraph WL algorithm, a two-stage label set remapping framework, where hyperedge labels are enhanced by merging the sorted set of labels from neighboring vertices and then compressing these augmented labels into new representations, and then vertex

Received 9 October 2024; revised 22 July 2025; accepted 4 September 2025. Date of publication 11 September 2025; date of current version 3 December 2025. This work was supported in part by the National Natural Science Foundation of China under Grant U24A20252, Grant 62327808, and Grant 623B2066, in part by the Fundamental Research Funds for the Central Universities under Grant xtr062025010, in part by the National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Xi'an Jiaotong University under Grant HMHAI-202412, in part by Beijing Natural Science Foundation under Grant L242167, and in part by the Key Research and Development Program of Shaanxi Province of China under Grant 2024PT-ZCK-66 and Grant 2024CY2-GJHX-48. Recommended for acceptance by Y. J. Lee. *(Corresponding author: Shaoyi Du.)*

Yifan Zhang and Shaoyi Du are with the Department of Ultrasound, The Second Affiliated Hospital of Xi'an Jiaotong University, State Key Laboratory of Human-Machine Hybrid Augmented Intelligence and Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: dxny2nd@gmail.com; dushaoyi@xjtu.edu.cn).

Yifan Feng and Yue Gao are with the School of Software, BNRist, THUIBCS, BLBCI, Tsinghua University, Beijing 100084, China (e-mail: evanfeng97@gmail.com; kevin.gaoy@gmail.com).

Shihui Ying is with the Shanghai Institute of Applied Mathematics and Mechanics, School of Mechanics and Engineering Science, Shanghai University, Shanghai 200072, China (e-mail: shying@shu.edu.cn).

Digital Object Identifier 10.1109/TPAMI.2025.3608902

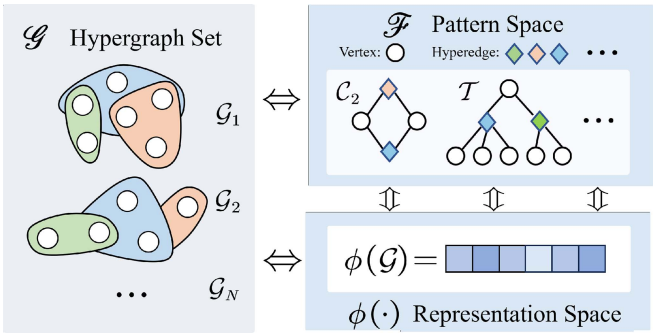


Fig. 1. The illustration of the relationship between hypergraph kernels and homomorphisms. Pattern space is defined as the complete set of all subhypergraphs, while representation space is the feature space into which these substructures are embedded. The process of extracting structural information via a hypergraph kernel amounts to encoding a sequence of specific motifs from the pattern space into this representation space.

labels are updated in the same way. They further instantiate their framework through two specific methods: Hypergraph Subtree kernel and Hyedge kernel. Experimental results demonstrate that both of them achieve excellent performance on various (hyper)graph classification benchmarks. However, these approaches primarily concentrate on the design of hypergraph kernels themselves, while overlooking the critical analysis of their expressive ability. And they develop hypergraph kernels based on distinct principles-such as neighborhoods, paths, or specific subhypergraph motifs-yet lack a unified framework for evaluating the expressiveness across different kernel types. The following issues remain to be addressed: ① *How to compare the expressive ability of these hypergraph kernels?* ②: *How to enhance their expressiveness?*

To address these issues, this paper revisits the underlying principles of hypergraph kernel expressiveness to distinguish high-order structures. From the perspective of homomorphism, we analyze the underlying mechanisms of of hypergraph kernels and point out that these kernels essentially map hypergraph structures from the original domain into a representation space, guided by a specific pattern space associated with each kernel, as illustrated in Fig. 1. Based on this framework, we analyze two representative hypergraph kernels that directly handle hypergraph structures: the Hypergraph Rooted kernel and the Hypergraph Weisfeiler-Lehman kernel, focusing on the structure of their respective pattern spaces. From the homomorphism perspective, we explain that the Hypergraph Weisfeiler-Lehman kernel currently has more powerful expressive ability with the pattern space $\mathscr{T}$, where $\mathscr{T}$ denotes the set of all hypergraph subtrees. The condition under which it fails to distinguish between hypergraphs is: if and only if, for every hypergraph subtree, the number of homomorphisms into the two hypergraphs is identical. In other words, the kernel cannot distinguish two hypergraphs when they appear structurally identical across all subtree-level mappings. This conclusion also highlights the limitations of Hypergraph WL kernels, offering theoretical insights for designing more powerful hypergraph kernels. We propose two strategies to enhance the expressive ability of the Hypergraph WL kernel by leveraging homomorphism count of hypergraph cycles and the return probabilities of hypergraph random walks, aiming to strike a balance between expressiveness and computational efficiency, as **HyperGraph Subtree-Cycle Kernelv1** and **v2** which expanding its corresponding pattern space to $\mathscr{T} \cup \mathscr{C}_L$, where $\mathscr{C}_L$ denotes the set hypergraph cycles of length at most $L$, and details are shown in Fig. 2. Extensive experiments and ablation studies on five graph datasets and ten hypergraph datasets demonstrate the effectiveness of our proposed methods. The main contributions of this work are summarized as follows:

- We propose a comparative framework to analyze the expressive ability of hypergraph kernels and derive conditions under which classical kernels, such as the Hypergraph WL kernels, fail to distinguish certain hypergraph structures. This provides theoretical guidance for enhancing the expressiveness of kernel.
- We introduce the Hypergraph Subtree-Cycle Kernel (HG SCKernel), designed to capture both tree-based and cycle-based motif patterns in hypergraphs. This kernel offers enhanced expressive ability, and we present two specific variants, HG SCKernelv1 and HG SCKernelv2.
- Comprehensive experiments on five graph and ten hypergraph datasets, together with detailed ablation studies, demonstrate the effectiveness and superiority of the proposed methods.

## II. Related Work

### A. Graph and Hypergraph Kernels

Graph and hypergraph kernels are essential tools for addressing similarity problems at the structure level. Research on graph kernels can generally be categorized into three main types: neighborhood or subgraph-based graph kernels, path-based graph kernels, and topology-based graph kernels. Weisfeiler-Lehman (WL) kernel [8] is one of the most widely used graph kernel methods. It is an improvement over the color refinement algorithm, where vertex labels are iteratively updated based on the neighborhood structure. The WWL [20] and WL-OA [21] kernels further enhance the WL kernel by using the Wasserstein distance kernel and the optimal assignment kernel, respectively, providing a more valid remap labels of similarity and achieving notable results. Glocalized WL [22] introduces the concept of high-dimensional global and local neighborhoods which theoretically improves the expressive ability of the WL kernel but also increases computational complexity. Another type of method is based on the theory of random walks on graphs. RWGK [23], [24] count the number of common random walks in the direct product graph. Shorted Path kernel [9], [25] calculates the shortest path matrix between pairs, and derives by comparing shortest paths between graphs, while position-aware GNN [26] utilize Bourgain's theorem [27] to inform the selection of anchor sets, ensuring that the resulting representation maintains low distortion. The last category of methods focuses on extracting global topological information. The Lovász-$\vartheta$ kernel and SVM-$\vartheta$[10] kernel utilize the Lovász number and its associated orthonormal representation to capture this information. P-WL [11]

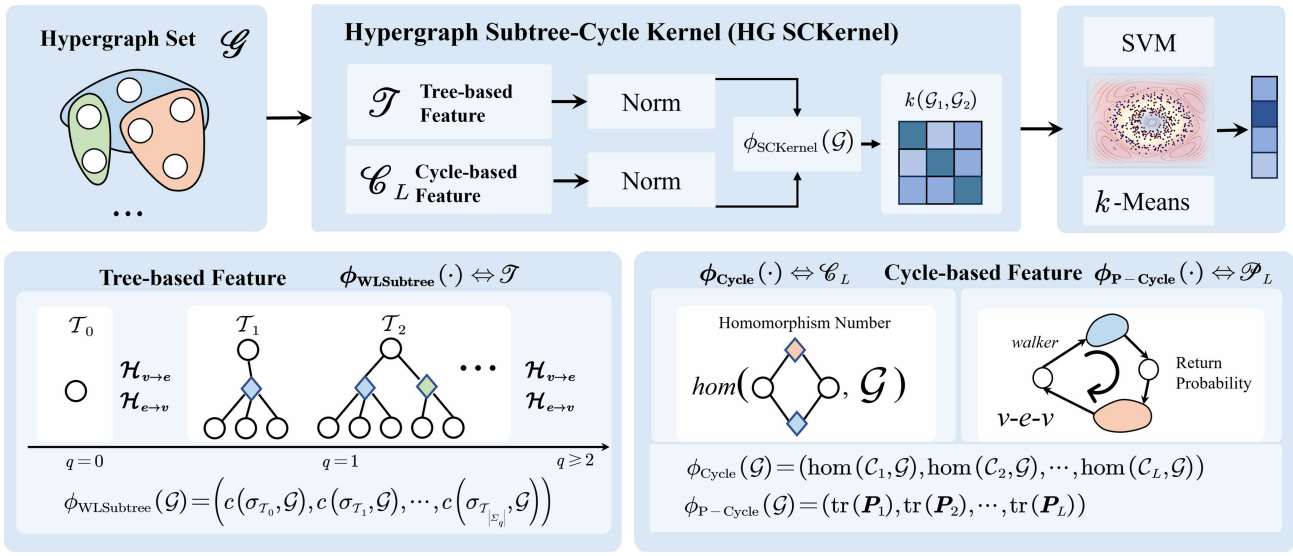


Fig. 2. The illustration of the proposed Hypergraph Subtree-Cycle Kernel.

applies persistence homology to extract connectivity and cycle characteristics.

These algorithms consider various aspects of graph structure information extraction methods, when processing hypergraph structures, they have to expand the hypergraph into a graph for further processing, which increases time consumption and also results in loss of structural information. Hypergraph kernels are methods that can directly process hypergraph structures. Hypergraph Rooted kernel [16] proposes a sampling-based method to extract vertex-hyperedge sequences from the original hypergraph. However, this method may lead to information loss and often fails in common scenarios. Hypergraph Directed Line kernel [12] proposes a transformation-based method that converts hypergraph structures into directed line graphs via the Perron-Frobenius operator, facilitating the application of the WL kernel to hypergraphs. AdE [15] designs a novel adaptive expansion-based methods to maintain high-level structure information. Feng et al. [19] propose a novel and generic WL isomorphism test algorithm that can capture more complex hypergraph structures via extracting high-order subtree count vector. MoCHy and HO-MA [18], [28] are methods based on hypergraph motifs, which constructed general hypergraph motifs and obtained motif count vectors in the hypergraph structure. MoCHy defines 26 h-motifs describing connectivity patterns of three connected hyperedges. HO-MA distinguishes higher-order motifs at the local scale, and proposes measures to analyze the nested structure of hyperedges.

### B. Expression Ability Analysis

According to Xu et al. [29], all GNNs based on message passing frameworks are at most as powerful as the 1-WL test to distinguish non-isomorphic graphs. Since then, WL has become a benchmark for evaluating the expressive ability of various graph kernels and graph neural networks, numerous studies have proposed more expressive methods by constructing counterexamples that the 1-WL or even $k$-WL tests fail to distinguish, such as ID-GNN [30], KP-GNN [14]. Holger et al. [31] establishes a connection between graph WL algorithm and Lovász theory. They demonstrate that two graphs are indistinguishable by the $k$-Weisfeiler-Lehman ($k$-WL) test if and only if every tree with treewidth at most $k$ has the same homomorphism count in both graphs. Graph homomorphism convolution (GHC) [32] categorizes common cases of $\mathcal{F}$ classes corresponding to $\mathcal{F}$-indistinguishability and proposes using homomorphism counts as isomorphism-invariant embeddings. SSWL [33] constructs a comprehensive hierarchy of the subgraph Weisfeiler-Lehman framework (SWL), summarizing existing subgraph WL and GNN methods, and demonstrates that they all grouped into six distinct SWL equivalence classes. Zhang et al. [34] propose a promising view that all GNNs and WL algorithms are equivalent to encoding a certain class of structures, with their expressive ability being positively correlated with the generality of that class. They also provide a quantitative framework for analysis the expressiveness of GNNs and graph kernels [34]. Feng et al. [19] rigorously prove that Hypergraph WL has the same expressive ability as 1-WL when processing simple graphs, and exhibits powerful expressiveness when applied to hypergraphs. MoCHy [18] and HO-MA [28] only consider predefined motifs (e.g., three connected hyperedges or 3-order hyperedges), lacking systematic analysis of the relationship between hypergraph motifs and kernel expressiveness.

## III. Preliminary

In this section, we introduce the background of hypergraph kernels and homomorphism, and present the Hypergraph WL algorithm in detail. Finally, we summarize the mathematical notions used in this paper for ease of reference in Table I.

TABLE I
BASIC NOTATIONS THROUGHOUT THIS PAPER

| Notation | Description |
|---|---|
| $\mathcal{G}, \mathscr{G}$ | Hypergraph and hypergraph set |
| $\mathcal{T}, \mathscr{T}$ | HG subtree and the set of HG subtree |
| $\mathcal{W}, \mathscr{W}$ | HG walk path and the set of HG walk path |
| $\mathcal{C}, \mathscr{C}$ | HG cycle and the set HG cycle |
| $\mathcal{F}, \mathscr{F}$ | sub-HG and the set of sub-HG |
| $\mathcal{T}^G, \mathscr{T}^G$ | Graph subtree and the set of graph subtree |
| $\mathcal{C}^G, \mathscr{C}^G$ | Graph cycle and the set of graph cycle |
| $\hom(\cdot,\cdot)$ | The number of homomorphism. |
| $\mathrm{Aut}(\cdot)$ | The number of automorphisms |
| $\mathrm{Surj}(\cdot,\cdot)$ | The number of surjective homomorphism |
| $\mathrm{Lolnjhom}(\cdot,\cdot)$ | The number of locally bijective homomorphism |
| $\sigma, \Sigma$ | Label and label space |
| $c(\cdot,\cdot)$ | Label count function |
| $\pi(\cdot)$ | Project function |
| $\mathbb{I}(\cdot)$ | Iverson braket |
| $\delta(\cdot,\cdot)$ | Dirac kernel |
| $k(\cdot,\cdot), \phi(\cdot)$ | Hypergraph kernel function |

### A. Hypergraph and Hypergraph Kernel

Formally, a hypergraph is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{I}^e, \mathcal{I}^v\}$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ is the set of hyperedges, and $\mathcal{I}^v$ and $\mathcal{I}^e$ denote the label sets for vertices and hyperedges, respectively. Each hyperedge $e \in \mathcal{E}$ in a hypergraph may contain any subset of vertices from $\mathcal{V}$. And hypergraphs are commonly represented using an incidence matrix $\boldsymbol{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$, where $\boldsymbol{H}(v,e) = 1$ indicates that vertex $v$ is included in hyperedge $e$. For any vertex $v \in \mathcal{V}$, we define $\mathcal{N}_e(v)$ as the set of hyperedges incident to $v$; likewise, for any hyperedge $e \in \mathcal{E}$, we define $\mathcal{N}_v(e)$ as the set of vertices it contains:

$$\begin{aligned}\mathcal{N}_v(d) &= \{u|\boldsymbol{H}(u,d) = 1, u \in \mathcal{V}\}, d \in \mathcal{E},\\ \mathcal{N}_e(u) &= \{d|\boldsymbol{H}(u,d) = 1, d \in \mathcal{E}\}, u \in \mathcal{V}.\end{aligned} \tag{1}$$

The diagonal vertex degree matrix and hyperedge degree matrix are $\boldsymbol{D}_v = \mathrm{diag}(\sum_{e\in\mathcal{E}} \boldsymbol{H}(v,e)) \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\boldsymbol{D}_e = \mathrm{diag}(\sum_{v\in\mathcal{V}} \boldsymbol{H}(v,e)) \in \mathbb{N}^{|\mathcal{E}|\times|\mathcal{E}|}$, respectively.

Hypergraph kernels are a class of approaches that operate by comparing pairs of hypergraphs through defined similarity measures. Let $\mathscr{G}$ represent the hypergraph set of $\{\mathcal{G}_i\}_{i=1}^N$ and $N$ denote the number of hypergraphs in $\mathscr{G}$. And for any $\mathscr{G}$, hypergraph kernel can be defined as $k : \mathscr{G} \times \mathscr{G} \to \mathbb{R}$, which maps any pair $\mathcal{G}_1$ and $\mathcal{G}_2$ to a real number representing a real-valued similarity score. A typical formulation of such a kernel is based on an implicit feature map $\phi_* : \mathscr{G} \to \mathcal{H}_*$ into a reproducing kernel Hilbert space (RKHS),

$$k(\mathcal{G}_1, \mathcal{G}_2) = \phi_*(\mathcal{G}_1)\phi_*(\mathcal{G}_2)^\top, \tag{2}$$

where the feature map $\phi_*(\cdot)$ depends on the specific kernel design. Hypergraph kernels are typically required to be isomorphism invariant if and only if its mapping results remain unchanged for isomorphic hypergraphs.

### B. Homomorphism Numbers on Hypergraphs

Hypergraph homomorphism is a structure-preserving map between two hypergraphs, and as described below [17]:

*Definition 1:* For any two hypergraphs $\mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{E}_1, \mathcal{I}_1^v, \mathcal{I}_1^e\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \mathcal{E}_2, \mathcal{I}_2^v, \mathcal{I}_2^e\}$, hypergraph homomorphism function $\pi : (\mathcal{V}_1, \mathcal{E}_1) \to (\mathcal{V}_2, \mathcal{E}_2)$ are defined as follows:

$$\begin{aligned}\pi\begin{pmatrix}v_1\\ e_1\end{pmatrix} &= \begin{pmatrix}\pi_{\mathcal{V}}(v_1)\\ \pi_{\mathcal{E}}(e_1)\end{pmatrix} \in (\mathcal{V}_2, \mathcal{E}_2),\\ \pi_{\mathcal{V}}(v_1) &\in \mathcal{N}_v(\pi_{\mathcal{E}}(e_1)), v_1 \in \mathcal{N}_v(e_1).\end{aligned} \tag{3}$$

A homomorphism between hypergraphs only map a hyperedge $e_1$ to a subset of $e_2' \in \mathcal{E}_2$ and for example,

$$\begin{aligned}\mathcal{G}_1 &= \{\{a,b\}, \{e_1 = \{a,b\}\}\},\\ \mathcal{G}_2 &= \{\{x,y,z\}, \{e_1' = \{x,y\}, e_2' = \{y,z\}\}\}.\end{aligned} \tag{4}$$

We can set a homomorphism from $\mathcal{G}_1$ to $\mathcal{G}_2$ as $\pi_{\mathcal{V}}(a,b) = (x,y), \pi_{\mathcal{E}}(e_1) = e_1'$. And $\hom(\mathcal{G}_1, \mathcal{G}_2)$ denotes the number of hypergraph homomorphisms from $\mathcal{G}_1$ and $\mathcal{G}_2$ as shown in (5):

$$\hom(\mathcal{G}_1, \mathcal{G}_2) = \sum_{\pi_{\mathcal{V}}:\mathcal{V}_1\to\mathcal{V}_2} \prod_{v\in\mathcal{N}_v(e)} \mathbb{I}\left[\pi_{\mathcal{V}}(v) \in \mathcal{N}_v(\pi_{\mathcal{E}}(e))\right], \tag{5}$$

where $\mathcal{G}_1$ and $\mathcal{G}_2$ are hypergraphs, $\mathbb{I}(\cdot)$ is the Iverson bracket.

### C. Hypergraph Weisfeiler-Lehman Kernel

The Hypergraph Weisfeiler-Lehman kernel [19] is an efficient hypergraph kernel derived from the Hypergraph Weisfeiler-Lehman algorithm, which simply iterates over each vertex and hyperedge in a hypergraph, and aggregates the labels from their neighbors. The aggregated multiset labels are then used to update the labels of vertices and hyperedges for the next iteration. For $\forall\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{I}^v, \mathcal{I}^e\} \in \mathscr{G}$, the hypergraph Weisfeiler-Lehman algorithm updates the hyperedge and vertex labels separately according to the injective hash functions $\mathcal{H}_{v\to e}$ and $\mathcal{H}_{e\to v}$, and the update process for each tag is shown in Algorithm 1. The updated vertex labels can be regarded as the labels of the corresponding subtree, then the algorithm can be viewed as the process of extracting subtree bases from the hypergraph set $\mathscr{G}$, while the subtree label space denotes as $\Sigma$. And any $\mathcal{G}_1$ and $\mathcal{G}_2$ in $\mathscr{G}$, the hypergraph Weisfeiler-Lehman subtree kernel as an instance of hypergraph WL kernel, can be further written as:

$$\begin{aligned}k_{\mathrm{WLsubtree}}^{(h)}(\mathcal{G}_1, \mathcal{G}_2) &= \phi_{\mathrm{WLSubtree}}(\mathcal{G}_1)\,\phi_{\mathrm{WLSubtree}}^\top(\mathcal{G}_2)\\ &= \left[\|_0^h \phi_{\mathrm{WLSubtree}}^{(q)}(\mathcal{G}_1)\right]\left[\|_0^h \phi_{\mathrm{WLSubtree}}^{(q)}(\mathcal{G}_2)\right]^\top\\ &= \sum_{q=0}^{h}\sum_{r=1}^{|\Sigma_q|} c(\sigma_q^r, \mathcal{G}_1)\cdot c\left(\sigma_q^r, \mathcal{G}_2\right),\end{aligned} \tag{6}$$

where $\Sigma_q \subseteq \Sigma$ as the updated vertex label set that occurs in iteration $q$ of the hypergraph Weisfeiler-Lehman algorithm, $c(\sigma_q^r, \mathcal{G})$ is the number of occurrences of the label $\sigma_q^r$ in the hypergraph $\mathcal{G}$.

**Algorithm 1:** Hypergraph Weisfeiler-Lehman Algorithm.

Hypergraph set $\mathscr{G}$, pre-defined injective hash function $\mathcal{H}_{v\to e}$ and $\mathcal{H}_{e\to v}$, $\Sigma \leftarrow \oslash$, the number of iterations $h$.
**for** $\mathcal{G} \in \mathscr{G}$ **do**
  $l^e(e_j) \leftarrow l_0^e(e_j) \in \mathcal{I}^e, l^v(v_i) \leftarrow l_0^v(v_i) \in \mathcal{I}^v$
**end for**
$\Sigma \leftarrow \Sigma \cup \mathcal{I}^v$
**for** $q \leftarrow 1$ to $h$ **do**
  $\Sigma_q \leftarrow \oslash$
  **for** $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{I}^v, \mathcal{I}^e\} \in \mathscr{G}$ **do**
    **for** $d \in \mathcal{E}$ **do**
      $l_q^e(d) \leftarrow \mathcal{H}_{v\to e}(l_{q-1}^e(d), \{l_{q-1}^v(u) : u \in \mathcal{N}_v(d)\})$
    **end for**
    **for** $u \in \mathcal{V}$ **do**
      $l_q^v(u) \leftarrow \mathcal{H}_{e\to v}(l_{q-1}^v(u), \{l_q^e(d) : d \in \mathcal{N}_e(u)\})$
    **end for**
    $\mathcal{I}^e \leftarrow \{l_q^e(d), d \in \mathcal{E}\}, \mathcal{I}^v \leftarrow \{l_q^v(u), u \in \mathcal{V}\}$
    $\Sigma_q \leftarrow \Sigma_q \cup \mathcal{I}^v$
  **end for**
  $\Sigma \leftarrow \Sigma \cup \Sigma_q$
**end for**
**return** $\Sigma$

## IV. Homomorphism-Based Expressive Ability Comparison Framework

In this section, we first propose a general framework for analyzing the expressive ability of hypergraph kernels via hypergraph homomorphism. Then we compare the framework with the existing hypergraph kernels such as Hypergraph Weisfeiler-Lehman kernels and Hypergraph Rooted kernels.

### A. Homomorphism-Guided Expressiveness Framework

Lovász [35] have shown a classic result between isomorphism and homomorphism to finite algebraic structures, and this conclusion also holds on hypergraphs.

*Theorem 1:* For any two hypergraphs $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic, if and only if all hypergraph structure $\mathcal{F} \in \mathscr{F}_{\text{all}}$, $\hom(\mathcal{F}, \mathcal{G}_1) = \hom(\mathcal{F}, \mathcal{G}_2)$, and $[\hom(\mathcal{F}, \mathcal{G}_1) : \mathcal{F} \in \mathscr{F}_{\text{all}}]$ is the Lovász vector of $\mathcal{G}_1$.

As Theorem 1, $\mathscr{F}_{\text{all}}$ represents all hypergraph substructures. And for any two hypergraphs $\mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{E}_1, \mathcal{I}_1^v, \mathcal{I}_1^e\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \mathcal{E}_2, \mathcal{I}_2^v, \mathcal{I}_2^e\}$, Obviously, Lovász vector is capable of distinguishing all non-isomorphic hypergraphs and thus possesses the most powerful expression ability. If a hypergraph kernel can distinguish all hypergraphs, it effectively encodes complete structural information and solves the hypergraph isomorphism problem in its entirety [8], [19], [35]. However, existing kernel methods inherently face an upper bound on structural expressive ability, because the isomorphism problem is NP-hard. In other words, they can only capture partial structural information and unable to achieve perfect discrimination power. To formally analyze and compare the expressive capacity of hypergraph kernels, we introduce a unified comparison framework based on hypergraph homomorphisms, as described below:

*Definition 2:* The general hypergraph kernel A's pattern space $\mathscr{F}_{\mathsf{A}}$ is usually subset of $\mathscr{F}_{\text{all}}$, and satisfy:

- For any $\mathcal{G}_1$ and $\mathcal{G}_2$, $\phi_{\mathsf{A}}(\mathcal{G}_1) = \phi_{\mathsf{A}}(\mathcal{G}_2)$ if and only if $\forall \mathcal{F} \in \mathscr{F}_{\mathsf{A}}, \hom(\mathcal{F}, \mathcal{G}_1) = \hom(\mathcal{F}, \mathcal{G}_2)$.
- There exists $\mathcal{G}_1$ and $\mathcal{G}_2$, $\phi_{\mathsf{A}}(\mathcal{G}_1) = \phi_{\mathsf{A}}(\mathcal{G}_2)$ but for $\forall \mathcal{F} \in \mathscr{F}_{\text{all}}/\mathscr{F}_{\mathsf{A}}, \hom(\mathcal{F}, \mathcal{G}_1) \neq \hom(\mathcal{F}, \mathcal{G}_2)$.

This framework essentially establishes an equivalence between the expressive ability of a hypergraph kernel and the homomorphism number vector over its associated pattern space. In doing so, the comparison of expressiveness is reduced to a comparison of pattern spaces. For example, if two hypergraph kernel methods A and B correpond to the pattern spaces of $\mathscr{F}_{\mathsf{A}}$ and $\mathscr{F}_{\mathsf{B}}$, then if $\mathscr{F}_{\mathsf{B}} \subset \mathscr{F}_{\mathsf{A}}$, the expressiveness of hypergraph kernel A is more powerful than hypergraph kernel B.

### B. Expressiveness Analysis of Classical Hypergraph Kernels

*1) Expressiveness Analysis of the Hypergraph Weisfeiler-Lehman Kernel:* The pattern space of hypergraph Weisfeiler-Lehman is the set of hypergraph subtree $\mathscr{T}$.

*Theorem 2:* For all hypergraphs $\mathcal{G}_1$ and $\mathcal{G}_2$, Hypergraph Weisfeiler-Lehman Kernel can't distinguish them is equivalent to $\forall \mathcal{T} \in \mathscr{T}, \hom(\mathcal{T}, \mathcal{G}_1) = \hom(\mathcal{T}, \mathcal{G}_2)$.

*Proof:* As states in existing works [17], [31], [34], the number of homomorphism from $\mathcal{T}$ to $\mathcal{G}$ is:

$$\hom(\mathcal{T}, \mathcal{G}) = \sum_{\mathcal{T}' \in \mathscr{T}} \text{Surj}(\mathcal{T}, \mathcal{T}') \frac{\text{Lolnjhom}(\mathcal{T}', \mathcal{G})}{\text{Aut}(\mathcal{T}')} \tag{7}$$

where $\text{Lolnjhom}(\mathcal{T}', \mathcal{G})$ is the number of locally injective homomorphisms, $\text{Aut}(\mathcal{T}')$ is the number of automorphisms of $\mathcal{T}'$, $\text{Surj}(\mathcal{T}, \mathcal{T}')$ is number of homomorphisms from $\mathcal{T}$ to $\mathcal{T}'$ that are surjective on both vertices and hyperedges. So $\text{Surj}(\mathcal{T}', \mathcal{T}) = 0$ if $\mathcal{T}$ has more $v$-$e$ pairs than $\mathcal{T}'$ in any depth.

For hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{I}^v, \mathcal{I}^e\}$, according to the $v$-$e$-$v$ traversal rule, it is possible to traverse all vertices applying a tree with a depth of at most $q^{\max} = \min(2|\mathcal{E}| + 1, 2|\mathcal{V}| - 1)$. We first establish that the proposition holds: for any two hypergraphs $\mathcal{G}_1$ and $\mathcal{G}_2$, Hypergraph Weisfeiler-Lehman kernel can't distinguish them is equivalent to $\forall \mathcal{T} \in \mathscr{T}_{\text{HGWL}}, \text{Lolnjhom}(\mathcal{T}, \mathcal{G}_1) = \text{Lolnjhom}(\mathcal{T}, \mathcal{G}_2)$. Let $\mathcal{T}_q(v, \mathcal{G}_1)$ denote any hypergraph subtree with the root $v$ and depth of $q$ on the $\mathcal{G}_1$. We devise a bijection $\pi_{\text{bi}}(\cdot)$ between possible label $\sigma_q$ and rooted tree $\mathcal{T}_q(v, \mathcal{G})$. For $q = 0$, the only allowed label is $\sigma_0^0$ and, the only subtree is $\pi_{\text{bi}}(\mathcal{T}_0) = \sigma_0^0$. For $q > 0$, let $\sigma_q$ be any label that could appear as the $\sigma_q^r = \pi_{\text{bi}}(\mathcal{T}(v, \mathcal{G}))$ for any hypergraph $\mathcal{G}$. The $\sigma_q^r$ is a set $\{\sigma_{q-1}^0, \ldots, \sigma_{q-1}^{|\Sigma_{q-1}|}\}$ consisting of $\Sigma_{q-1}$ labels to create in the $q-1$-th iteration. Let $\pi_{\text{bi}}(\mathcal{T}_{q-1}^{\sigma_{q-1}^r}) = \sigma_{q-1}^r$ denote the corresponding subtree with depth $q-1$. Then as the Lemma 2.5 [36],

at the $q$-th iteration,

$$\begin{aligned}\sigma_q^r &= \mathcal{H}_{e-v}\left(\left\{l_{q-1}^d | d \in \mathcal{N}_e(v)\right\}\right), \\ &= \mathcal{H}_{e-v}\left(\left\{\mathcal{H}_{v-e}\left(\left\{\sigma_{q-1}^r = \sigma\left(l_{q-1}^u\right) | u \in \mathcal{N}_v(d)\right\}\right) | \right.\right. \\ &\quad \left.\left. \times d \in \mathcal{N}_e(u)\right\}\right), \\ &= \mathcal{H}_{e-v}\left(\left\{\mathcal{H}_{v-e}\left(\left\{\pi_{\text{bi}}\left(\mathcal{T}_{q-1}(v,\mathcal{G})\right) | u \in \mathcal{N}_v(d)\right\}\right) | \right.\right. \\ &\quad \left.\left. \times d \in \mathcal{N}_e(u)\right\}\right), \\ &= \pi_{\text{bi}}\left(\left\{\left\{\left(\mathcal{T}_{q-1}(v,\mathcal{G})\right)|, v \in \mathcal{N}_v(d)\right\}_d | d \in \mathcal{N}_e(v)\right\}\right), \\ &= \pi_{\text{bi}}\left(\mathcal{T}_q(v,\mathcal{G})\right).\end{aligned}$$

According to [19]'s, hypergraph WL kernel cannot distinguish them is: for $q$-th iteration, $\phi^{(q)}_{\text{WLSubtree}}(\mathcal{G}_1) \equiv \phi^{(q)}_{\text{WLSubtree}}(\mathcal{G}_2)$, and is equivalent to:

$$\left[c(\sigma_q^0,\mathcal{G}_1),\ldots,c\left(\sigma_q^{|\Sigma_q|},\mathcal{G}_1\right)\right] \equiv \left[c(\sigma_q^0,\mathcal{G}_2),\ldots,c\left(\sigma_q^{|\Sigma^q|},\mathcal{G}_2\right)\right].$$

and then,

$$\begin{aligned}\text{Lolnjhom}(\mathcal{T},\mathcal{G}_1) &= \text{Lolnjhom}\left(\mathcal{T}_q(v,\mathcal{G}_1),\mathcal{G}_1\right) \\ &= \text{Lolnjhom}\left(\pi^{-1}_{\text{Lolnj}}(\sigma_q^r),\mathcal{G}_1\right) \\ &= \text{Lolnjhom}\left(\mathcal{T}_q(v,\mathcal{G}_2),\mathcal{G}_1\right) \\ &= \text{Lolnjhom}\left(\mathcal{T}_q(v,\mathcal{G}_2),\mathcal{G}_2\right) \\ &= \text{Lolnjhom}\left(\mathcal{T},\mathcal{G}_2\right)\end{aligned}$$

Then there exist other bijection that transform $\text{Lolnjhom}(\mathcal{T},\mathcal{G})$ to $c(\pi_{\text{bi}}(\mathcal{T}),\mathcal{G})$, and for any $\mathcal{T} \in \mathscr{T}_{\text{HGWL}}$,

$$\text{Lolnjhom}\left(\mathcal{T},\mathcal{G}_1\right) \equiv \text{Lolnjhom}\left(\mathcal{T},\mathcal{G}_2\right). \tag{8}$$

Then, We first prove the sufficiency, that is, for two hypergraphs $\mathcal{G}_1$ and $\mathcal{G}_2$, if the hypergraph Weisfeiler-Lehman cannot distinguish them, then $\forall \mathcal{T} \in \mathscr{T}$, the homomorphism numbers $\hom(\mathcal{T},\mathcal{G}_1)$ and $\hom(\mathcal{T},\mathcal{G}_2)$ are the same. According to the formula above, for any subtree $\mathcal{T} \in \mathscr{T}_{\text{HGWL}}$, $\hom(\mathcal{T},\mathcal{G}_1) = \hom(\mathcal{T},\mathcal{G}_2)$. If its depth exceeds the maximum traversable depth, it is no longer possible to construct a locallly injective homomoprhism onto to the original hypergraph. In this case, the number of such homomorphism is 0 for both $\mathcal{T} \notin \mathscr{T}_{\text{HGWL}}$. Therefore:

$$\begin{aligned}\hom(\mathcal{T},\mathcal{G}_1) &= \sum_{\mathcal{T}' \in \mathscr{T}_{\text{HGWL}}} \text{Surj}\left(\mathcal{T},\mathcal{T}'\right) \frac{\text{Lolnjhom}(\mathcal{T}',\mathcal{G}_1)}{\text{Aut}(\mathcal{T}')} + 0 \\ &= \sum_{\mathcal{T}' \in \mathscr{T}_{\text{HGWL}}} \text{Surj}\left(\mathcal{T},\mathcal{T}'\right) \frac{\text{Lolnjhom}(\mathcal{T}',\mathcal{G}_2)}{\text{Aut}(\mathcal{T}')} \\ &= \hom\left(\mathcal{T},\mathcal{G}_2\right)\end{aligned}$$

and the sufficiency is proven.

Finally, we prove the necessity. For $\mathcal{G}_1$ and $\mathcal{G}_2$, if $\forall \mathcal{T} \in \mathscr{T}, \hom(\mathcal{T},\mathcal{G}_1) = \hom(\mathcal{T},\mathcal{G}_2)$, Hypergraph Weisfeiler-Lehman kernel cannot distinguish between two hypergraphs.

$$\begin{aligned}0 &= \hom\left(\mathcal{T},\mathcal{G}_1\right) - \hom\left(\mathcal{T},\mathcal{G}_2\right) \\ &= \sum_{\mathcal{T}' \in \mathscr{T}} \frac{\text{Surj}\left(\mathcal{T},\mathcal{T}'\right)}{\text{Aut}\left(\mathcal{T}'\right)} \left(\text{Lojhom}\left(\mathcal{T}',\mathcal{G}_1\right) - \text{Lojhom}\left(\mathcal{T}',\mathcal{G}_2\right)\right) \\ &= \sum_{\mathcal{T}' \in \mathscr{T}_{\text{HGWL}}} \frac{\text{Surj}\left(\mathcal{T},\mathcal{T}'\right)}{\text{Aut}\left(\mathcal{T}'\right)} \pi\left(c\left(\pi_{\text{bi}}\left(\mathcal{T}'\right),\mathcal{G}_1\right) - c\left(\pi_{\text{bi}}\left(\mathcal{T}'\right),\mathcal{G}_2\right)\right) \\ &= \sum_{\sigma' \in \Sigma} \frac{\text{Surj}\left(\pi_{\text{bi}}^{-1}(\sigma),\pi_{\text{bi}}^{-1}\left(\sigma'\right)\right)}{\text{Aut}(\pi_{\text{bi}}^{-1}(\sigma'))} \pi\left(c\left(\sigma',\mathcal{G}_1\right) - c\left(\sigma',\mathcal{G}_2\right)\right) \\ &= \boldsymbol{A}_{\mathcal{T}}\left(\phi_{\text{WLSubtree}}(\mathcal{G}_1) - \phi_{\text{WLSubtree}}\left(\mathcal{G}_2\right)\right)\end{aligned}$$

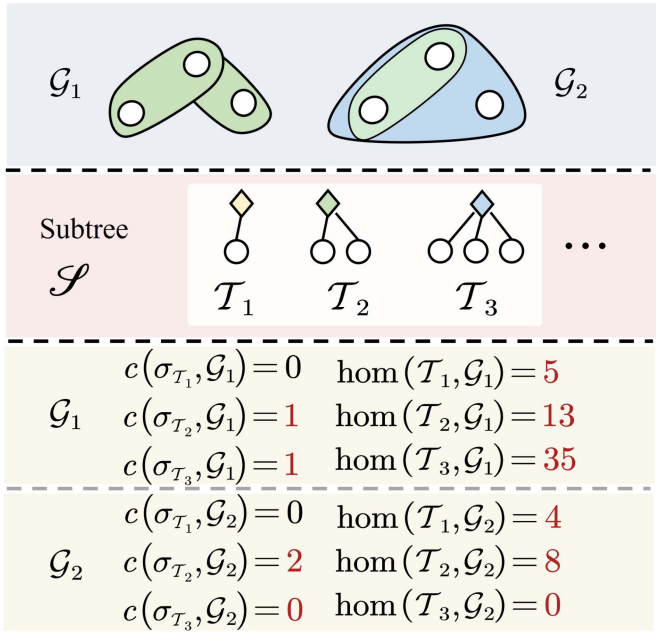


Fig. 3. A calculation example of hypergraph WL kernel and homomorphism number.

where $\pi$ is other bijective from $\text{Lolnjhom}(\mathcal{T},\mathcal{G})$ to $c(\sigma,\mathcal{G})$, and $\text{Surj}(\mathcal{T},\mathcal{T}') \neq 0$ iff $\mathcal{T}' \subseteq \mathcal{T}$, so this can formulate as a linear system as $\boldsymbol{A}_{\mathcal{T}}x = 0$, and then $\sigma \in \Sigma, c(\sigma,\mathcal{G}_1) \equiv c(\sigma,\mathcal{G}_2)$, and the necessity is proven.

Consequently, we establish a theoretical connection between the homomorphism counts over hypergraph subtree patterns, the expressive power of HG WL kernels. Fig. 3 shows a calculation example on hypergraph WL kernel and homomorphism number between two small hypergraphs. And hypergraph WL algorithm find subtree heuristically as $\mathcal{T}_1,\mathcal{T}_2,\cdots$, remap subtrees as new labels, and count the new labels as the final features, while the calculation of homomorphism follows (7). And the result shows both hypergraph WL kernel and homomorphism number can distinguish these two hypergraphs. Theorem 2 shows that Hypergraph Weisfeiler-Lehman only focuses on the tree-based group $\mathscr{T}$, and this also explains why Hypergraph Weisfeiler-Lehman kernel cannot distinguish all non-isomorphic hypergraphs. Therefore, when there is another method that can extract structures as $\mathcal{F}$ and $\mathcal{F} \notin \mathscr{T}$, the expression ability of this method will be strictly more powerful than Hypergraph Weisfeiler-Lehman.

*2) Expressiveness Analysis of the Hypergraph Rooted Kernel:* To ensure a fair comparison of the similarities between all methods, we initially set the labels of all vertices and hyperedges to be identical. hypergraph Rooted Kernel rooted kernel stores the relative exit and entry positions in the random walk paths within a string, and counts the random walk paths from each given hypergraph. This means that the structural information extraction of the method is based on the set of random walk paths $\mathscr{W}$, where each $\mathcal{W}_l \in \mathscr{W}$ retains only the label attributes and the relative position properties of each visited point within the hyperedge. During the random walk, no constraints are placed on the IDs of the visited points. Therefore, any path $\mathcal{W}$ can be viewed as a cycle-free chain structure, where vertex hyperedge labels alternate. Thus, $\mathscr{W} \subset \mathscr{T}$, Hypergraph Weisfeiler-Lehman is more powerful than Hypergraph Rooted Kernel under this setting.

## V. Enhancing Hypergraph WL Kernels via Cycle Modeling

In this section, we introduce supplementary information to obtain a hypergraph kernel with more expressive ability.

### A. Incorporating Cycle-Based Structural Information

Since Section IV-B shows that hypergraph WL captures only acyclic subtree information, cycle-related features can complement its limitations. We define a cycle on a hypergraph as follows:

*Definition 3:* For a closed path $p_l^v$ on a hypergraph, if all parts except the starting and ending points form a simple path, it is defined as a cycle on the hypergraph, as $\mathcal{C}_l(v)$.

It notes that in Definition 3, multiple vertices within the same hyperedge do not constitute a hypergraph cycle. The set of hypergraph cycles with a length not greater than $L$ is denoted as $\mathscr{C}_L$. For any $\mathcal{G}$, we can directly apply $\mathscr{C}_L$ as a template and obtain the homomorphism number vector as an enhanced hypergraph-level feature, as follows:

$$\phi_{\text{Cycle}}^{(L)}(\mathcal{G}) = [hom(\mathcal{C}_l, \mathcal{G}) : \mathcal{C}_l \in \mathscr{C}_L] \in \mathbb{R}^L. \tag{9}$$

For $\forall l \leq L$, the runtime complexity for calculating the cycle homomorphism number $\hom(\mathcal{C}_l, \mathcal{G})$ is $\mathcal{O}((|\mathcal{V}| + |\mathcal{E}|)^{2\,l})$ [37]. This exponential growth in complexity not only demands significant computational resources, but may also lead to considerable runtime, thereby limiting the applicability of this method in time-sensitive tasks. To address this, we next propose an efficient approximation using random walk.

Here, we use the probability of forming a closed path of length $l$ via a random walk on a hypergraph as an alternative to the cycle homomorphism count $\hom(\mathcal{C}_l, \mathcal{G})$. A hypergraph without isolated vertices is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{I}^v, \mathcal{I}^e\}$, the random walk on hypergraph is typically defined following the $v$-$e$-$v$ pattern [38], [39]. That is, starting from a vertex $v_s$, a random walker performs the following steps:

- $v$-$e$ Stage: Choose $e_s \in \mathcal{N}_e(v_s)$, with the probability $1/|\mathcal{N}_e(v_s)|$ at random.
- $e$-$v$ Stage: Choose $v_d \in \mathcal{N}_v(e_s)$, with the probability $(1-\alpha)/(|\mathcal{N}_v(e_s)|-1)$ at other vertice $v_d \neq v_s$, and the probability $\alpha$ return to $v_s$.

and the parameter $\alpha \in [0,1]$ controls the probability of backtracking. Based on this transition mechanism, the one-step transition probability from $v_s$ to $v_d$ is defined as:

$$p_{v_s,v_d} = \begin{cases} \sum_{e_s \in \mathcal{N}_e(v_s)} \frac{1}{|\mathcal{N}_e(v_s)|} \frac{1-\alpha}{|\mathcal{N}_v(e_s)|-1}, & \text{if } v_d \neq v_s \\ \alpha, & \text{others} \end{cases}. \tag{10}$$

And the transition matrix $\boldsymbol{P} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of a random walk on the hypergraph $\mathcal{G}$:

$$\boldsymbol{P} = \boldsymbol{D}_v^{-1}(1-\alpha)\boldsymbol{H}\left(\boldsymbol{D}_e - \boldsymbol{I}\right)^{-1}\boldsymbol{H}^\top \odot (\boldsymbol{1} - \boldsymbol{I}) + \alpha \boldsymbol{I}, \tag{11}$$

where $\odot$ denotes Hamamard product, to increase universality, the return probability of isolated vertex can be set to 1. Let (11) short for $\boldsymbol{P} = \boldsymbol{D}_v^{-1/2}\boldsymbol{R}$, and then $l$-step random walk matrix can be denoted as:

$$\boldsymbol{P}^l = \boldsymbol{D}_v^{-1/2}(\boldsymbol{R})^l \boldsymbol{D}_v^{1/2}, \tag{12}$$

Let $\{\lambda_v, \boldsymbol{u}_v\}_{v\in\mathcal{V}}$ denote eigenvalues and eigenvectors, respectively, so the probability of the path becoming a closed path after $l$-step walk is

$$\boldsymbol{P}_{vv}^l = \boldsymbol{R}_{vv}^l = \sum_{v\in\mathcal{V}} \lambda_v^l \left[(\boldsymbol{u}_v)(\boldsymbol{u}_v)^\top\right], \tag{13}$$

$\boldsymbol{P}_{vv}^l$ is the probability of forming a closed hypergraph path through $l$-steps random walks on hypergraph. Subsequently, the following formula is applied as a substitute for $\phi_{\text{Cycle}}(\mathcal{G})$:

$$\phi_{\text{P-Cycle}}^{(L)}(\mathcal{G}) = \left[\sum_{v\in\mathcal{V}} \boldsymbol{P}_{vv}^l : l \in [1, L]\right] \in \mathbb{R}^L, \tag{14}$$

*Theorem 3:* $\phi_{\text{P-Cycle}}^{(L)}(\mathcal{G})$ is isomorphism invariant.

*Proof:* Any two hypergraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same size $|\mathcal{V}_1| = |\mathcal{V}_2|, |\mathcal{E}_1| = |\mathcal{E}_2|$, let $\{(\lambda_v, \boldsymbol{u}_v)\}_{v\in\mathcal{V}_1}$ and $\{(\mu_v, \boldsymbol{v}_v)\}_{v\in\mathcal{V}_2}$ be eigenvaules and eigenvectors of $\boldsymbol{P}_{\mathcal{G}_1}$ and $\boldsymbol{P}_{\mathcal{G}_2}$. if $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphism, then there exists a permutation $\pi_{\mathcal{V}}: \mathcal{V}_1 \to \mathcal{V}_2$: $\boldsymbol{H}_i = \boldsymbol{H}_{\pi_{\mathcal{V}}(i)}$. And as (11), we obtain $\forall l \in [1, L]$, $(\boldsymbol{P}_{\mathcal{G}_1})_{vv}^l = (\boldsymbol{P}_{\mathcal{G}_2})_{\pi_{\mathcal{V}}(v)\pi_{\mathcal{V}}(v)}^l$. For (14), due to the fact that the addition operation satisfies permutation invariance, $\phi_{\text{P-Cycle}}^{(L)}(\mathcal{G}_1) = \phi_{\text{P-Cycle}}^{(L)}(\mathcal{G}_2)$. Therefore, the proposition is proved.

Similarly, $\phi_{\text{Cycle}}(\mathcal{G})$ is also isomorphism invariant, and both can be used for the further design of hypergraph kernels.

### B. Design of the Hypergraph Subtree-Cycle Kernel

In this subsection, we integrate tree-based and cycle-based structure infrastructure features of hypergraphs and introduce a more powerful hypergraph kernel, the Hypergraph Subtree-Cycle Kernel.

*Definition 4:* Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two hypergraphs, the Hypergraph Subtree-Cycle Kernel on them is defined as:

$$k_{\text{SCKernel}}(\mathcal{G}_1, \mathcal{G}_2) = \phi_{\text{SCKernel}}(\mathcal{G}_1)\phi^{\top}_{\text{SCKernel}}(\mathcal{G}_2), \quad (15)$$

The feature map $\phi_{\text{SCKernel}}(\cdot)$ is derived via cooperate with hypergraph subtree and hypergraph cycle feature, defined as follows:

$$\phi_{\text{SCKernelv1}}(\mathcal{G}) = \left[\|_{q=0}^{h}\phi^{(q)}_{\text{WLsubtree}}(\mathcal{G}), \phi^{(L)}_{\text{Cycle}}(\mathcal{G})\right]. \quad (16)$$

$$\phi_{\text{SCKernelv2}}(\mathcal{G}) = \left[\|_{q=0}^{h}\phi^{(q)}_{\text{WLsubtree}}(\mathcal{G}), \phi^{(L)}_{\text{P-Cycle}}(\mathcal{G})\right]. \quad (17)$$

while $\|_{q=1}^{h}\phi_{\text{WLSubtree}}(\cdot)$ follows Hypergraph WL Subtree algorithm with $h$ iterations, which is used to extract subtree-based information, and calculate $\phi^{(L)}_{\text{Cycle}}(\cdot)$ or $\phi^{(L)}_{\text{P-Cycle}}(\cdot)$ via (9) or (14) to obtain cycle-based information. Two types of kernels are named HG SCKernelv1 and HG SCKernelv2.

It is evident that the HG SCKernel is isomorphism invariant, as each of its components, such as $\phi^{(L)}_{\text{Cycle}}$, is also isomorphism invariant. The condition of the Hypergraph Subtree-Cycle Kernel $k_{\text{SCKernel}}(\mathcal{G}_1, \mathcal{G}_2)$ cannot distinguish $\mathcal{G}_1$ and $\mathcal{G}_2$ is if and only if the homomorphism numbers are the same for any $\mathcal{F} \in \mathscr{T} \cup \mathscr{C}_L$. Obviously, $\mathscr{C}_L \cap \mathscr{T} = \oslash$, so $\mathscr{T} \subset \mathscr{C}_L \cup \mathscr{T}, \mathscr{C}_L \subset \mathscr{C}_L \cup \mathscr{T}$ and our method is more powerful than any tree-based or cycle-based hypergraph kernels.

*Complexity:* For a hypergraph set $\mathscr{G}$ with $N$ hypergraphs, the complexity of $\phi_{\text{WLSubtree}}(\cdot)$ is $\mathcal{O}(Nh\bar{m})$, where $\bar{m}$ is the average hypergraph capacity as stated in [8], [19]. For $\phi_{\text{Cycle}}(\cdot)$, the complexity of runtime is $\mathcal{O}(N(|\bar{\mathcal{V}}| + |\bar{\mathcal{E}}|)^{2\,L})$. For $\phi_{\text{P-Cycle}}(\cdot)$, the eigen-decomposition of $\boldsymbol{R}^{s}_{vv}$ requires time $\mathcal{O}(|\bar{\mathcal{V}}|^3)$. So the total time complexity of the above computational method is $\mathcal{O}(|\bar{\mathcal{V}}|^3 + (L+1)|\bar{\mathcal{V}}|^2)$. Our method runtime will cost $\mathcal{O}(Nh\bar{m} + N|\bar{\mathcal{V}}|^3 + N(L+1)|\bar{\mathcal{V}}|^2 + N^2\,d)$, where $d$ is the dimension of each feature vector.

### C. Expressiveness Analysis of Hypergraph Subtree-Cycle Kernel

We now analyze the expressiveness of HG SCKernelv2 in relation to HG SCKernelv1. We first prove that the expressiveness of SCKernelv1 is not weaker than that of SCKernelv2. We then demonstrate that under certain conditions, the expressiveness of SCKernelv2 can reach the upper bound of SCKernelv1's expressiveness. All proofs in this subsection that the pruning module only serves as early stopping and does not affect the expressiveness of Hypergraph WL Subtree Kernel.

*1) Hypergraph SCKernelv2 is at Most as Powerful as SCKernelv1:* In previous discussions, we have learned that HG SCKernelv1 cannot distinguish between two hypergraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ if and only if for any $\mathcal{F} \in \mathscr{T} \cup \mathscr{C}_L$, $\hom(\mathcal{F}, \mathcal{G}_1) = \hom(\mathcal{F}, \mathcal{G}_2)$. The set $\mathscr{T} \cup \mathscr{C}_L$ includes all hypergraph subtrees $\mathscr{T}$ and hypergraph cycles within the length of $L$ as $\mathscr{C}_L$. On the other hand, HG SCKernelv2's information comes from all closed paths, and we denote these paths set as $\mathscr{P}_L$, the paths are simplified as $v$-$e$-$v$, while $v$ denotes the starting and ending point, and $\{p\}$ denotes the sets of hyperedges and vertices for pathways. We categorize and discuss all situations of $\mathscr{P}_L$:

1) If the path $\mathcal{P}_l \in \mathscr{P}_L$, excluding the endpoint, is a simple path, then origin path is a hypergraph cycle $\mathcal{P}_l \in \mathscr{C}_L$.
2) Otherwise, if $\{p\}$ does not include the starting point, there must exist two repeated positions $v_m$ or $e_m$, meaning there is a segment of the path that is repeated from the starting point to that position. However, compared to a typical random walk, a closed path only imposes restrictions on the starting and ending points, without distinguishing the identities of the vertices in the path. Therefore, from the perspective of hypergraph cores, $\mathcal{P}_l$ and the corresponding $\mathcal{C}_l$ are equivalent.
3) If the starting point appears in the path, we split the path into two closed paths and analyze each segment separately.

Therefore, $\forall$ closed path $\mathcal{P}_l \in \mathscr{P}_L$ in closed paths on hypergraphs, satisfying that $\mathcal{P}_l \in \mathscr{T} \cup \mathscr{C}_L$, meaning $\mathscr{P}_L \subseteq \mathscr{T} \cup \mathscr{C}_L$. Thus, the expressiveness of HG SCKernelv1 is not weaker than that of HG SCKernelv2. For HG SCKernelv2, due to not extracting homomorphism number directly, its expressiveness not only relies on the motif pattern space, but also relies on the transition probability from vertex $v_s$ to $e_s$, and then from $e_s$ to $v_t$. Compared to homomorphism counts, this approach carries a risk of information loss. For example, when $\alpha = 1$, and $\boldsymbol{P} = \boldsymbol{I}$, which becomes ineffective. However, when $\boldsymbol{P} \propto \boldsymbol{H}\boldsymbol{H}^{\top}$, then:

$$\phi_{\text{P-Cycle}}(\mathcal{G}) \propto \left[tr\left(\left(\boldsymbol{H}\boldsymbol{H}^{\top}\right)^{l}\right) : l \in [1, L]\right], \quad (18)$$

Referencing Theorem 4 and (14), $\phi_{\text{Cycle}}(\mathcal{G}) \propto \hat{\phi}_{\text{Cycle}}(\mathcal{G})$, Hypergraph SCKernelv2 has the same expressiveness as Hypergraph SCKernelv1.

*Theorem 4:* We have $\hom(\mathcal{C}_l, \mathcal{G}) \propto tr((\boldsymbol{H}\boldsymbol{H}^{\top})^l)$, where $\mathcal{C}_l \in \mathscr{C}_L$ is a length $l$ cycle and $\boldsymbol{H}$ is the incidence matrix of hypergraph $\mathcal{G}$.

*Proof:* For all $l \geq 0$, the number of homomorphisms from cycle $\mathcal{C}_l$ of length $l$ to a hypergraph $\mathcal{G}$ with incidence matrix $\boldsymbol{H}$ is equal to the number of closed length $l$ path in $\mathcal{G}$, which in turn is equal to the trace of $(\boldsymbol{H}\boldsymbol{H}^{\top})^l$. Thus for hypergraph $\mathcal{G}_1$ and $\mathcal{G}_2$ with incidence matices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, we have $\hom(\mathcal{C}_l, \mathcal{G}_1) = \hom(\mathcal{C}_l, \mathcal{G}_2)$ if and only if $tr((\boldsymbol{H}_1\boldsymbol{H}_1^{\top}))$ is equal to $tr((\boldsymbol{H}_2\boldsymbol{H}_2^{\top}))$, when $l \geq 0$. Thus we have $hom(\mathcal{C}_l, \mathcal{G}) \propto tr((\boldsymbol{H}\boldsymbol{H}^{\top})^l)$.

*2) There Exists a Subhypergraph-Based Kernel That is More Powerful Than HG SCKernel:* Fig. 4 shows the expressiveness comparison of different hypergraph kernels, which reveals their hierarchical relationships. Since the pattern space of the Hypergraph Subtree-cycle kernel is only $\mathscr{T} \cup \mathscr{C}_L$, we denote it as $\mathscr{F}_{\text{HG SCKernel}}$. Its expressiveness upper limit is consistent with the expressiveness of hypergraph kernels based on all cycles or $v$-$e$-$v$ random walk collaborative hypergraph WL algorithm, whereas for Definition 2, there exists $\mathcal{G}_1$ and $\mathcal{G}_2$, $\phi_{\text{HG SCKernel}}(\mathcal{G}_1) = \phi_{\text{HG SCKernel}}(\mathcal{G}_2)$ but for $\forall \mathcal{F} \in \mathscr{F}_{\text{all}}/\mathscr{F}_{\text{HG SCKernel}}$, $\hom(\mathcal{F}, \mathcal{G}_1) \neq \hom(\mathcal{F}, \mathcal{G}_2)$. Similarly, we introduce corresponding subgraphs into the pattern space of

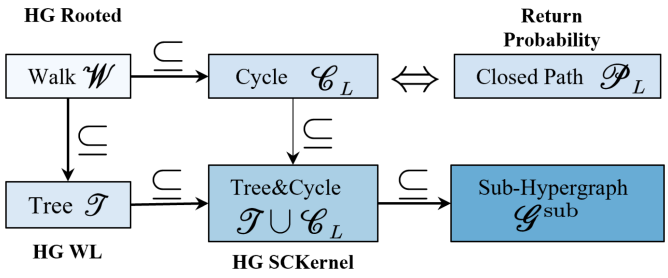


Fig. 4. Expressiveness comparison of hypergraph kernels.

the new hypergraph kernel to further enhance its expressiveness. This also highlights the advantage of employing homomorphisms to examine hypergraph kernels; it does not require searching for counterexamples but instead allows us to understand the mechanisms of complex hypergraph kernels from the perspective of simple pattern spaces.

## VI. Experiments

In this section, we first introduce the experiment settings, including datasets, compared methods, and other implementation details. Then we conduct experiments on graph/hypergraph classification tasks, and other ablation studies to validate the effectiveness of our methods.

### A. Experiments Settings

*1) Graph and Hypergraph Datasets:* We used the following fifteen benchmark datasets used in graph and hypergraph classification tasks: They include five graph datasets, RG-Sub, RG-Macro, MUTAG, PROTEINS, IMDB-BINARY, and ten hypergraph datasets, include RHG-3, RHG-10, Steam-Player, IMDB-Wri-Form, IMDB-Wri-Genre, IMDB-Dir-Form, IMDB-Dir-Genre, Twitter-Friend, IMDB-Wri-Genre-M, IMDB-Dir-Genre-M [19]. RG-Macro, RG-Sub, RHG-3 and RHG-10 are synthetic datasets combining six canonical graphs types and ten hypergraph factors, RG-Macro is labeled by macro-structure and RG-Sub uses the subgraph types as the label. RHG-3 and RHG-10 are labeled by their basic hypergraph factors. MUTAG [40], PROTEINS [41] are bioinformatics datasets. MUTAG contains 188 mutagenic aromatic and heteroaromatic nitro compounds with prediction task focused on their mutagenicity in Salmonella typhimurium. Each graph represents a molecule, where the vertices correspond to atoms, and the edges are the chemical bonds between these atoms. PROTEINS models protein structures as graphs: vertices are secondary structure elements (SSE) such as $\alpha$-helices, $\beta$-sheets, or turns, and edges are drawn between SSEs that are adjacent either in the amino acid sequence or spatially in 3D. The prefix "IMDB-" in the dataset refers to movie datasets. Each director/writer forms a hypergraph. The dataset name contains "Form", which means that film categories are classified by their form, such as animation, documentary, and drama. "Genre" means that films are classified by their genre, such as adventure, crime, and family. The Steam-Player is a game player dataset where each player is a hypergraph. The vertices denote games and the hyperedges are the relations linking the games with the same tags. The label of dataset is to identify each player's preference: single-player-game or multi-player-game. The Twitter-Friend is a social media dataset. Hypergraphs are constructed by users who are friends with the specified user, and hyperedges are grouped according to the friend relationships among users within each hypergraph. To fairly compare the structure extraction capabilities of different hypergraph kernels, we set the initial features of all vertices and hyperedges to the current degrees of the vertices and hyperedges. In the ablation study, an in-depth analysis is performed on four hypergraph datasets, including Steam-Player, IMDB-Wri-Form, IMDB-Wri-Genre, and IMDB-Dir-Genre. More details are shown in Table II.

*2) Comparison With Existing Methods:* We compare our method with four classic graph kernel methods such as GraphLet [42], WL Subtree [8], Shortest Path [9], GHC-T&C [32], eight classic hypergraph kernel methods, like HG Rooted kernel [16], Directed Line kernel [12], HG WL Subtree and HG WL Hyperedge [19], and hypergraph motifs-based method HO-MA [28]. The following is a detailed introduction to each comparison method:

- *GraphLet:* GraphLet counts small induced subgraphs and captures local structural patterns for each graph.
- *WL Subtree:* Weisfeiler-Lehman Subtree kernel works by iteratively updating vertex labels using the sorted sets of neighbor and compressing these into new labels.
- *Shortest Path:* This kernel utilizes Floyd algorithm to obtain shortest path matrix for each graph, and the matrices are then summed to form a graph-level representation.
- *GHC-T&C:* Graph homomorphism convolution is applied using specific trees and cycles (with size $\leq 6$) to calculate the homomorphism counts.
- *Vertex Hist:* This method employs histograms of vertex labels' distribution histogram as the representation of the hypergraph level.
- *H-SVD:* We refer to the idea of spectral clustering, perform singular value decomposition on the $\boldsymbol{H}$ matrix, and obtain the eigenvectors of the left feature matrix corresponding to the first 10 smallest singular values as the features of the hypergraph.
- *HO-MA:* Higher-Order Motifs Analysis extract higher-order profilesm and put forward a series of measures for investigating the nested structure of hypergraphs, thus offering a principle-based approach to extracting higher-order fingerprints within hypergraphs.
- *Directed Line:* Directed Line kernel is a transformation-based method that transforms a hypergraph into a directed graph. Then, the directed WL Subtree is applied to extract the feature map of a given hypergraph.
- *HG Rooted:* A sampling-based hypergraph kernel that counts paths within the hypergraph to create its embedding.
- *HG WL Subtree:* HG WL subtree kernel is a classical type of instance to compress a unique rooted subtree as enhanced labels and counts them to capture the high-order information.
- *HG WL Hyedge:* HG WL Hyperedge kernel utilizes the hyperedge as the root and counts the number of different hyperedge subtrees to capture the high-order structure information.

TABLE II
DATASET STATISTICS OF TEN GRAPH/HYPERGRAPH DATASETS

| Dataset | Types | #Hypergraph | #Classes | Averaged #Vertices | Averaged #Hyperedges | Averaged Degree of hyperedges |
|---|---|---|---|---|---|---|
| RG_Sub | Graph | 1000 | 6 | 27.5 | 47.7 | 2.0 |
| RG_Macro | Graph | 1000 | 6 | 27.5 | 47.7 | 2.0 |
| MUTAG | Graph | 188 | 2 | 17.9 | 19.8 | 2.0 |
| PROTEINS | Graph | 1113 | 2 | 39.1 | 72.8 | 2.0 |
| IMDB-BINARY | Graph | 1000 | 2 | 19.8 | 96.5 | 2.0 |
| RHG-3 | Hypergraph | 2000 | 10 | 35.5 | 29.8 | 5.2 |
| RHG-10 | Hypergraph | 1500 | 3 | 31.3 | 17.9 | 6.9 |
| Steam-Player | Hypergraph | 2048 | 2 | 13.8 | 46.4 | 4.5 |
| Twitter-Friend | Hypergraph | 1310 | 2 | 21.6 | 84.3 | 4.3 |
| IMDB-Wri-Form | Hypergraph | 374 | 4 | 10.1 | 3.7 | 5.0 |
| IMDB-Wri-Genre | Hypergraph | 1172 | 6 | 12.8 | 4.4 | 5.2 |
| IMDB-Dir-Form | Hypergraph | 1869 | 3 | 15.7 | 39.2 | 3.7 |
| IMDB-Dir-Genre | Hypergraph | 3393 | 3 | 17.3 | 36.4 | 3.8 |
| IMDB-Dir-Genre-M | Hypergraph | 1554 | 7 | 15.7 | 40.8 | 5.0 |
| IMDB-Wri-Genre-M | Hypergraph | 2048 | 6 | 10.3 | 3.7 | 5.0 |

"#" denotes "number of".

TABLE III
GRAPH CLASSIFICATION TASK RESULTS ONTHE REAL-WORLD GRAPH

| Method | RG-Macro | | RG-Sub | | MUTAG | | PROTEINS | | IMDB-BINARY | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma |
| GraphLet | 0.3800 | 0.3195 | 0.6020 | 0.5919 | 0.8087 | 0.7699 | 0.6864 | 0.6236 | 0.6010 | 0.5959 |
| WL Subtree | 0.6860 | 0.6720 | 0.9190 | 0.9218 | 0.8245 | 0.7822 | 0.7196 | 0.6709 | 0.7250 | 0.7244 |
| Shortest Path | 0.6430 | 0.6362 | 0.5910 | 0.5737 | 0.8565 | 0.8426 | 0.7107 | 0.6648 | 0.7250 | 0.7230 |
| GHC-T&C | 0.4380 | 0.4103 | 0.7140 | 0.7125 | 0.8543 | 0.8458 | 0.7222 | 0.7039 | 0.6890 | 0.6867 |
| Vertex Hist | 0.3090 | 0.2630 | 0.7910 | 0.7912 | 0.7451 | 0.6478 | 0.6252 | 0.5017 | 0.7220 | 0.7216 |
| H-SVD | 0.1590 | 0.0575 | 0.3060 | 0.2149 | 0.6648 | 0.3982 | 0.7115 | 0.6644 | 0.4850 | 0.3834 |
| HO-MA | 0.3340 | 0.2412 | 0.5600 | 0.5529 | 0.6648 | 0.3983 | 0.6252 | 0.5017 | 0.6496 | 0.6496 |
| HG Rooted | 0.1790 | 0.0933 | 0.1850 | 0.0866 | 0.6649 | 0.3983 | 0.5956 | 0.3731 | 0.4950 | 0.4200 |
| Directed Line | 0.6040 | 0.5441 | 0.6820 | 0.6851 | 0.6650 | 0.3986 | 0.6738 | 0.6493 | 0.6370 | 0.6361 |
| HG WL Subtree | 0.6860 | 0.6720 | 0.9190 | 0.9218 | 0.8245 | 0.7822 | 0.7196 | 0.6709 | 0.7250 | 0.7244 |
| HG WL Hyedge | 0.6490 | 0.6347 | 0.9250 | 0.9268 | 0.8141 | 0.7738 | 0.7071 | 0.6482 | 0.7440 | 0.7432 |
| HG WL Tree-OA | 0.6630 | 0.6442 | 0.9250 | 0.9251 | 0.8561 | 0.8381 | 0.6936 | 0.6831 | 0.7330 | 0.7314 |
| HG SCKernelv1 | 0.7160 | 0.7057 | 0.9420 | 0.9426 | 0.8673 | 0.8524 | 0.7187 | 0.6731 | 0.7400 | 0.7394 |
| HG SCKernelv2 | **0.8440** | **0.8397** | **0.9840** | **0.9840** | **0.8831** | **0.8699** | **0.7304** | **0.7042** | **0.7470** | **0.7462** |

The best results are marked in bold type.

- *HG WL Tree-OA:* This kernel extends the HG WL Subtree kernel by integrating the optimal assignment kernel to capture hierarchical features, as outlined in [21].
- *HG SCKernelv1 and HG SCKernelv2:* Both of them are the methods proposed in the article. SCKernelv1 employs the hypergraph homomorphism number of $\mathscr{C}_L$ as a supplementary feature, while SCKernelv2 utilizes an alternative method in Section V-A.

*3) Other Details:* The hypergraph kernel can directly treat simple graphs as 2-uniform hypergraphs. And to make the graph kernel method apply to hypergraph classification, we use clique expansion to convert the hypergraph into a graph. All the experiments are conducted on a machine equipped with 40 Intel(R) Xeon(R) E5-2640 v4 @ 2.40 GHz CPUs and a single NVIDIA GeForce RTX 2080Ti GPU.

### B. Graph/Hypergraph Classification

The subsection presents the experiments on five graph datasets and eight hypergraph datasets for structure-level prediction. The statistics of these datasets are gathered in Table II. We conduct experiments with 5 different seeds and report the average and variance of accuracy (Acc.) and F1-macro (F1_ma) score for single-label classification task, exact match ratio (EMR) and example-based accuracy/precision (EB-Acc, EB-Pre) for the multi-label classification task. Each dataset is divided into 5 folds. We utilize SVM as the classifier to compare these methods, while the standard SVM is initialized with the same hyperparameters $C = 1$ for all methods.

*1) Experiments on Graph Datasets:* Experimental result on five graph datasets on Table III. We have the following three observations. Firstly, our method also demonstrates good generalizability in handling graph classification tasks. Notably, the margin over the best baseline reaches 15.8% on RG-Macro (Acc.), indicating substantial representational improvements. Then, GHC-T&C incorporates cycle motifs but is limited to shallow structures, resulting in lower performance on complex datasets. Our approach overcomes this limitation by combining subtree-based embeddings, enabling the model to preserve long-range cyclic dependencies without sacrificing tree structure fidelity. Finally, The HG WL Subtree and WL Subtree are equivalent in handling graph classification tasks, while the HG WL Tree-OA, using the optimal assignment kernel, is able to adapt to the hierarchical structure generated by subtrees. It has outperformed the WL Subtree on several datasets such as RG-Sub. However, its fundamental representation basis is still

TABLE IV
HYPERGRAPH CLASSIFICATION TASK RESULTS ON REAL-WORLD HYPERGRAPH

| Method | Stream-Player | | Twitter-Friend | | IMDB-Wri-Form | | IMDB-Wri-Genre | | IMDB-Dir-Form | | IMDB-Dir-Genre | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma |
| GraphLet | 0.4805 | 0.3440 | 0.5878 | 0.3759 | 0.4654 | 0.2215 | 0.3353 | 0.2318 | 0.5265 | 0.3922 | 0.6086 | 0.5058 |
| WL Subtree | 0.5293 | 0.5189 | 0.5931 | 0.4467 | 0.4681 | 0.2994 | 0.4846 | 0.4202 | 0.6453 | 0.5967 | 0.7424 | 0.6619 |
| Shortest Path | 0.5215 | 0.5156 | 0.5961 | 0.4751 | 0.4681 | 0.2259 | 0.4403 | 0.3704 | 0.6362 | 0.5890 | 0.7176 | 0.6492 |
| GHC-T&C | 0.5244 | 0.5207 | 0.5969 | 0.3972 | 0.4654 | 0.2132 | 0.4266 | 0.3267 | 0.5987 | 0.5485 | 0.6614 | 0.5364 |
| Vertex Hist | 0.5610 | 0.5600 | 0.5877 | 0.3945 | 0.4304 | 0.2068 | 0.3106 | 0.1906 | 0.6115 | 0.5297 | 0.7061 | 0.6725 |
| H-SVD | 0.4873 | 0.3276 | 0.6076 | 0.4297 | 0.4574 | 0.1559 | 0.2619 | 0.0691 | 0.2892 | 0.1255 | 0.5125 | 0.2258 |
| HO-MA | 0.4970 | 0.3933 | 0.5927 | 0.3718 | 0.4575 | 0.1559 | 0.2892 | 0.1255 | 0.5393 | 0.3938 | 0.6749 | 0.4737 |
| HG Rooted | 0.4868 | 0.4855 | 0.5817 | 0.4635 | 0.4438 | 0.2471 | 0.4215 | 0.3755 | 0.5944 | 0.5247 | 0.6891 | 0.6095 |
| Directed Line | 0.5312 | 0.5206 | OOM | OOM | 0.4574 | 0.2604 | 0.4999 | 0.4269 | 0.6773 | 0.6316 | OOM | 0.OOM |
| HG WL Subtree | 0.5620 | 0.5467 | 0.5954 | 0.4800 | 0.4732 | 0.2990 | 0.5307 | 0.4738 | 0.6741 | 0.6227 | 0.7804 | 0.7379 |
| HG WL Hyedge | 0.5493 | 0.5474 | 0.5809 | 0.3971 | 0.4679 | 0.3335 | 0.5614 | 0.4953 | 0.6688 | 0.6197 | 0.7698 | 0.7165 |
| HG WL Tree-OA | 0.5585 | 0.5512 | 0.5816 | 0.4378 | 0.4866 | 0.2798 | 0.4939 | 0.3708 | 0.6655 | 0.6125 | 0.7645 | 0.7068 |
| HG SCKernelv1 | 0.5815 | 0.5805 | 0.6076 | 0.5443 | 0.4706 | 0.3412 | 0.5580 | 0.4931 | 0.6704 | 0.6203 | 0.7869 | 0.7439 |
| HG SCKernelv2 | **0.6079** | **0.6044** | **0.6389** | **0.6105** | **0.5079** | **0.3822** | **0.5785** | **0.5072** | **0.6774** | **0.6317** | **0.8034** | **0.7640** |

The best results are marked in bold type.

TABLE V
EXPERIMENTAL RESULTS OF MULTI-LABEL CLASSIFICATION ON REAL-WORLD HYPERGRAPH DATASETS

| Method | IMDB-Dir-Genre-M | | | IMDB-Wri-Genre-M | | |
|---|---|---|---|---|---|---|
| | EMR | EB_acc | EB_pre | EMR | EB_acc | EB_pre |
| GraphLet | 0.1795 | 0.4008 | 0.5663 | 0.0524 | 0.3306 | 0.5740 |
| WL Subtree | 0.3533 | 0.4823 | 0.5954 | 0.2122 | 0.4327 | 0.6199 |
| Shortest Path | 0.3622 | 0.4953 | 0.5963 | 0.1656 | 0.3958 | 0.5965 |
| GHC-T&C | 0.3346 | 0.4562 | 0.5544 | 0.1832 | 0.4033 | 0.5996 |
| Vertex Hist | 0.2503 | 0.4432 | 0.6017 | 0.0697 | 0.3519 | 0.5827 |
| H-SVD | 0.2560 | 0.3813 | 0.5117 | 0.1714 | 0.4269 | 0.5966 |
| HO-MA | 0.1833 | 0.2345 | 0.2646 | 0.1394 | 0.3867 | 0.5231 |
| HG Rooted | 0.2799 | 0.4159 | 0.5357 | 0.1626 | 0.3789 | 0.5762 |
| Directed Line | 0.3410 | 0.4779 | 0.6075 | 0.1684 | 0.4249 | 0.6131 |
| HG WL Subtree | 0.3739 | 0.5199 | 0.6500 | 0.2238 | 0.4463 | 0.6620 |
| HG WL Hyedge | 0.3539 | 0.5014 | 0.6210 | 0.2616 | 0.4668 | 0.6586 |
| HG WL Tree-OA | 0.3294 | 0.4814 | 0.6235 | 0.2150 | 0.4477 | 0.6759 |
| HG SCKernelv1 | 0.3616 | 0.5186 | 0.6469 | 0.2412 | 0.4600 | 0.6585 |
| HG SCKernelv2 | **0.3783** | **0.5363** | **0.6655** | **0.2789** | **0.4782** | **0.6615** |

The best results are marked in bold type.

$\mathcal{T}$, which is a subset of the representation basis of the HG SCKernel. As a result, its performance is still inferior to that of the methods we propose.

*2) Experiments on Hypergraph Datasets:* We evaluate our proposed HG SCKernel on eight real-world hypergraph classification datasets, including six single-label and two multi-label benchmarks. The results are presented in Tables IV and V. Based on those results, we derive the following observations. First, HG SCKernel achieves the best performance improvement across all datasets, demonstrating superior expressiveness and generalizability. For example, on Stream-Player, it outperforms HG WL Subtree by 8.16% and 10.55%, respectively. The results validate the effectiveness of incorporating both subtree-based and cycle-based structural representations. One of the main reasons for the limited improvement of HG SCKernelv1 on the IMDB-Dir-Genre dataset may be attributed to its method of directly extracting homomorphism counts as discrete features, while it is not as suitable for classification tasks as the continuous features of HG SCKernelv2. Second, HG Rooted, HO-MA, and Vertex Hist exhibit inferior performance due to their limited pattern spaces. HG Rooted, which relies on sampled acyclic paths, performs poorly on IMDB-Wri-Form and IMDB-Wri-Genre, with Acc. of 0.4438 and 0.4215, respectively. This suggests a high sensitivity to sampling noise and incomplete pattern extraction. HO-MA, which only captures 3-order nested motifs, achieves a macro-F1 of 0.2197 on IMDB-Wri-Form, which is 16.2% lower than HG SCKernelv2. Vertex Hist, with its shallow degree-based pattern space, also fails to capture high-order dependencies, resulting in relatively weak performance across datasets. Third, compared with path-based kernels(e.g. HG Rooted), subtree-based kernels(e.g. HG WL Subtree) deliver competitive results, but their expressive ability remains constrained. These kernels operate in a fixed subtree pattern space $\mathcal{T}$, which is inherently a subset of the structural space captured by HG SCKernels. For instance, on IMDB-Dir-Form, HG WL Tree-OA achieves a macro-F1 of 0.6125, whereas HG SCKernelv2 improves this to 0.6317 indicating that its broader structural coverage contributes to improved predictive performance. Finally, graph-based methods such as WL Subtree and Shortest Path underperform consistently. HG SCKernelv2 outperforms WL Subtree by a notable 8.7% in macro-F1 score. This performance gap demonstrates that naive clique expansions are unable to preserve the semantic integrity of hypergraph structures and often introduce spurious edges, leading to representation drift and degraded results.

### C. Ablation Analyses

In this subsection, we conduct further ablation studies on four hypergraph datasets. We implement seven sets of ablation experiments to evaluate our proposed Hypergraph Subtree-Cycle Kernel.

*1) Ablation Analysis on Hypergraph Cycle or Closed Path Length of L for $\mathscr{C}_L$:* As $L$ increases, the pattern space expands monotonically ($\mathscr{C}_L \subset \mathscr{C}_{L+1}$), leading to a gradual powerful of hypergraph Subtree-Cycle kernel. In this subsection, we investigate the performance of HG SCKernel series across different $L$ values, and the results are shown in Fig. 6. A consistent performance gain is observed with increasing $L$, which can

be attributed to the richer structural information captured by longer walk cycles and deeper subtree patterns. During our experiments, we found significant performance gains, particularly when the number of steps is in the range of 1 to 3. This suggests that a moderate $L$ allows the model to extract relevant features effectively while maintaining a manageable level of complexity. However, it is essential to recognize that as $L$ continues to increase beyond this optimal range, the model may extract excessive information, because $L$ step walks returning to the original vertex do not guarantee simple paths, and the probability of obtaining a simple path decreases exponentially as $L$ increases. This overload can hinder the classifier's ability to utilize the extracted features effectively, resulting in a deterioration of classification performance. Therefore, while increasing $L$ can enhance the model's capability, it is crucial to strike a balance to avoid diminishing returns. In summary, our findings underscore the importance of carefully selecting $L$ to maximize the effectiveness of hypergraph SCKernels in classification tasks.

*2) Ablation Analysis on Tree and Cycle Feature Map:* In this study, we further conduct ablation experiments on the information extraction capability of subtree, cycle, and subtree-cycle features with the results detailed in Fig. 7. We explore four settings: only $\phi_{\text{WLSubtree}}(\cdot)$, only $\phi_{\text{P-Cycle}}(\cdot)$, $\phi_{\text{SCKernelv1}}(\cdot)$ and $\phi_{\text{SCKernelv2}}(\cdot)$. This indicates that tree-based and cycle-based kernels can capture different types of high-order patterns of hypergraphs. Integrating two types of feature can enhance the ability of capture more nuanced high-order pattern features, potentially resulting in better performance in downstream tasks. Secondly, compared with using homomorphism counts directly, the $\phi_{\text{PCycle}}$ method transforms counting into continuous probabilistic features, yielding an average performance improvement of 4.3% over HG SCKernelv1 across four datasets.

*3) Ablation Analysis on Random Walk Strategies:* In this study, we evaluate the comprehensive performance of different hypergraph random walk methods, including naive random walk (RW), edge-dependent weight random walk (EDW-RW), and the random walk used in HG SCKernelv2, while naive random walk is implented the same as Zhou et al. [38], and for EDW-RW [43], [44], edge-dependent weights are calculated based on original vertex and edge degrees, transforming the random walk process into irreversible Markov chains. Our random walk in $\phi_{\text{P-Cycle}}$ is an improved version of the non-lazy random walk [43], intrducing rule-based constraints to the symmetry of the naive random walk. The results of different RW methods are summarized in the Table VII. While the edgeindependent method enhances the walk process by constructing an irreversible Markov chain, it over-relies on edgedependent vertex weights and thus neglects the inherent structural properties of hypergraphs. In this work, we focus on investigating the expressive capacity of hypergraph kernels for structure alone; therefore, our datasets lack vertex senmantic attributes, and the advantages of edgedependent vertex weights are not leveraged. Moreover, both methods operate within the same pattern space $\mathscr{T} \cup \mathscr{C}_L$, differing only in how they handle and aggregate specific structural information-a limitation we discuss in Section VII-B. Finally, our proposed $\phi_{\text{P-Cycle}}(\cdot)$ method captures cycle-based features, making the HG SCKernelv2 more powerful and effective.

*4) Ablation Analysis on Different Classifiers:* To evaluate the flexibility of the proposed hypergraph kernels, we replace the standard SVM classifier with several commonly used machine learning models, as shown in Table VI. The results indicate that compared with the proposed hypergraph kernels and the method of origin kernels that only extract subtree or cycle patterns, performance improvements is achieved on different classifiers. For example, on the IMDB-Wri-Genre dataset, integrating with HG SCKernel results in a performance gain of over 10%. This improvement is primarily attributed to the enhanced expressive power of our method, which effectively overcomes the structural limitations of existing hypergraph kernels.

*5) Ablation Analysis on Runtime:* In this study, to comprehensively evaluate the runtime performance of different methods, we conduct an ablation analysis on the runtime performance of different methods. To ensure the fairness of the evaluation, HG WL Subtree and HG WL Hyedge Kernel are specifically selected as benchmark models, which demonstrate great balance in terms of time overhead, making them reliable references for measuring the performance of other kernels. Specifically, this study explores two directions: real-world dataset verification and scale scalability testing: we compare the F1-score and runtime of each method on steam-player dataset; we construct several random hypergraph datasets with vertice size ranging from 10 to 700.

The releval experimental results are shown in Fig. 5, and we draw the following four conclusions: First, our proposed HG SCKerenl series methods show excellent performance beyond existing methods. HG SCKernelv2 achieves the state-of-the-art performance with only a small increase in computation time, effectively verifying the excellent balance of this method between performance and efficiency. Second, graph kernels need to project hypergraphs into graph structures, which not only consumes a lot of computing resources but also inevitably causes the loss of high-order information, eventually leading to a significant decline in performance, highlighting the necessity of directly processing hypergraph structures. Third, due to the high time complexity of themselves, Directed Line Kernel and HG SCKernelv1 methods have increasing computation time when facing hypergraphs with more than 200 vertices, which is difficult to meet the timeliness requirements in practical applications. Finally, when the hypergraph scale does not exceed 500 vertices, the runtime of HG SCKernelv2 is similar to that of the benchmark model HG WL Subtree, but it shows greater structural expression ability; as the hypergraph scale further increases to 700 vertices, although the time-consuming growth of HG SCKernelv2 has increased, its overall processing time is still within the acceptable range of practical applications.

*6) Ablation Analysis on Hypergraph Clustering:* In this study, we tested the versatility of our methods in hypergraph clustering tasks on three datasets: RHG-3, RHG-10, and RHG-Table, which is a subset of RHG-10 consists of the two type of "table". To comprehensively verify the reliability of the clustering results from multiple perspectives, we selected five evaluation metrics, including external indices (NMI, ARI) and internal indices (SC, DBI, CHI). Compared with HG WL kernel,

TABLE VI
ABLATION STUDY ON HYPERGRAPH KERNELS WITH DIFFERENT CLASSIFIERS

| Dataset | Method | $k$ Nearest Neighbor | | Logistic Regression | | Naive Bayes | | Grad-Boosting | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma |
| Stream-Player | HG WL Subtree | 0.5269 | 0.5130 | 0.5684 | 0.5329 | 0.5395 | 0.5034 | 0.5400 | 0.5056 |
| | HG WL Hyedge | 0.5161 | 0.5158 | 0.5605 | 0.5536 | 0.5518 | 0.5502 | 0.5460 | 0.5447 |
| | HG SCKernelv2 | **0.5498** | **0.5480** | **0.6231** | **0.6164** | **0.5630** | **0.5617** | **0.5703** | **0.5695** |
| IMDB-Wri-Form | HG WL Subtree | 0.4143 | 0.2791 | 0.4759 | 0.3124 | 0.3607 | 0.3247 | 0.4679 | 0.3486 |
| | HG WL Hyedge | 0.3957 | 0.2767 | 0.4867 | 0.3228 | 0.2273 | 0.1953 | 0.4813 | 0.3435 |
| | HG SCKernelv2 | **0.4361** | **0.3095** | **0.5026** | **0.3715** | **0.3904** | **0.3595** | **0.4839** | **0.3666** |
| IMDB-Wri-Genre | HG WL Subtree | 0.4138 | 0.3654 | 0.5392 | 0.4806 | 0.3763 | 0.3242 | 0.5478 | 0.4841 |
| | HG WL Hyedge | 0.3780 | 0.3256 | 0.5674 | 0.4978 | 0.2730 | 0.2453 | 0.5580 | 0.4867 |
| | HG SCKernelv2 | **0.4667** | **0.4057** | **0.5708** | **0.5000** | **0.4641** | **0.3974** | **0.5653** | **0.5047** |
| IMDB-Dir-Genre | HG WL Subtree | 0.7504 | 0.6922 | 0.7754 | 0.7269 | 0.7185 | 0.6880 | 0.7728 | 0.7247 |
| | HG WL Hyedge | 0.7439 | 0.6799 | 0.7698 | 0.7143 | 0.7241 | 0.6939 | 0.7666 | 0.7161 |
| | HG SCKernelv2 | **0.7648** | **0.7119** | **0.7902** | **0.7436** | **0.7436** | **0.7087** | **0.7825** | **0.7398** |

The best results are marked in bold type.

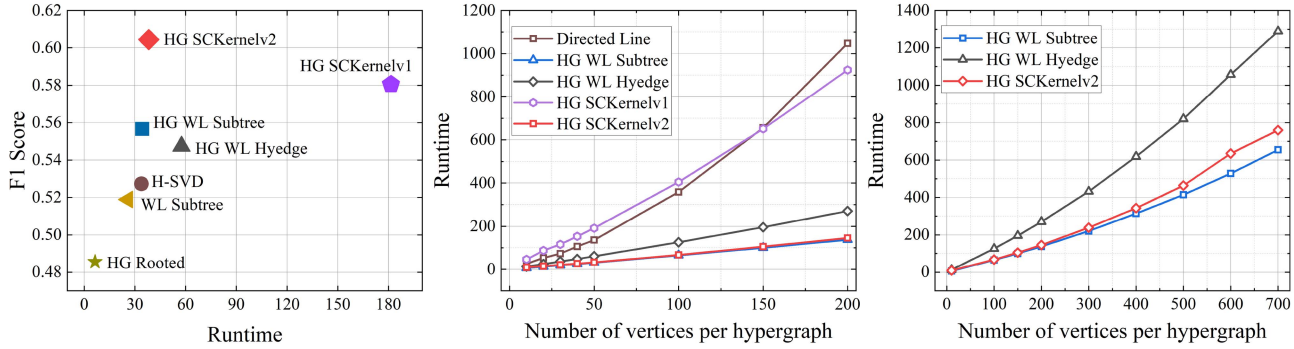


Fig. 5. Runtime comparison for kernel matrix computation on different datasets.

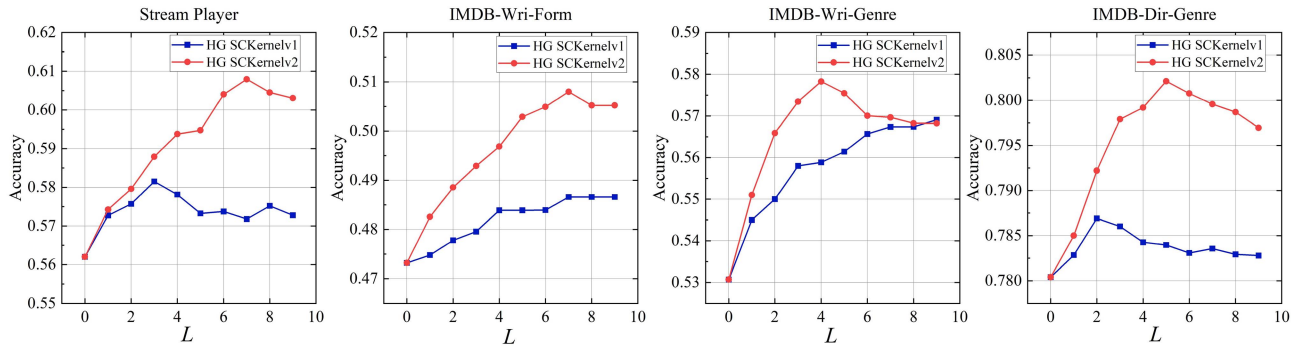


Fig. 6. Experimental results of ablation study on the hypergraph cycle length of $L$.

the experimental results demonstrate remarkable performance advantages. In terms of external metrics, consistently outperforms other baseline methods in all three datasets, indicating a significant degree of consistency between the clustering results and the ground-truth labels. Regarding internal metrics, SC, DBI, and CHI are solely based on the compactness within clusters and the separation between clusters. In Table VIII, compared with HG WL Subtree and HG SCKernelv2, HG WL Hyedge involves an additional vertex-to-hyperedge remapping process, which leads to more thorough separation of data points in the RHG-3 and RHG-10 datasets. By introducing cycle-based structural information, HG SCKernelv2 achieves higher separation and compactness than HG WL Subtree while ensuring more accurate clustering.

*7) Ablation Analysis on Extension of HGNN:* To assess the effectiveness of our proposed extensions to hypergraph neural networks, we utilize HGNN$^+$ as the baseline, where the original vertice features are one-hot encoded from original labels into

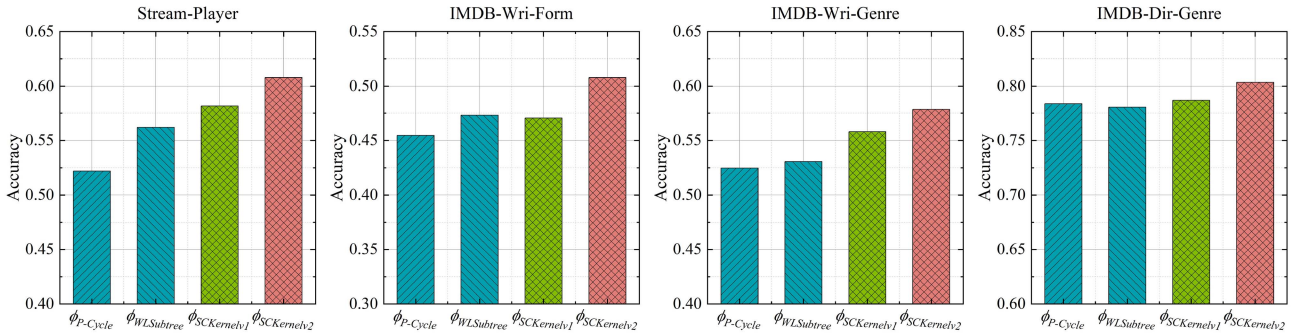


Fig. 7. Experimental results of ablation study on different Tree and Cycle kernels.

TABLE VII
EXPERIMENT RESULT OF ABLATION ON DIFFERENT HYPERGRAPH RANDOM WALK

| Dataset | Metrics | $\phi_{\text{RW}}$ | $\phi_{\text{EDW-RW}}$ | $\phi_{\text{P-Cycle}}$ |
|---|---|---|---|---|
| Stream-Player | Acc. | 0.5815 | 0.5903 | **0.6079** |
| | F1_ma | 0.5804 | 0.5897 | **0.6044** |
| IMDB-Wri-Form | Acc. | 0.4705 | 0.4732 | **0.5078** |
| | F1_ma | 0.3412 | 0.3416 | **0.3821** |
| IMDB-Wri-Genre | Acc. | 0.5579 | 0.5776 | **0.5784** |
| | F1_ma | 0.4930 | **0.5077** | 0.5071 |
| IMDB-Dir-Genre | Acc. | 0.6682 | 0.6704 | **0.6773** |
| | F1_ma | 0.6197 | 0.6183 | **0.6316** |

The best results are marked in bold type.

TABLE VIII
EXPERIMENT RESULTS OF ABLATION STUDY ON HYPERGRAPH CLUSTERING TASK

| RHG-Table | NMI | ARI | SC | DBI | CHI |
|---|---|---|---|---|---|
| HG WL Subtree | 0.5985 | 0.6111 | 0.5788 | 0.6345 | 343.061 |
| HG WL Hyedge | 0.6099 | 0.6266 | 0.6064 | 0.5995 | 330.150 |
| HG SCKernelv2 | **0.9417** | **0.9681** | **0.6527** | **0.5619** | **494.523** |
| RHG-3 | NMI | ARI | SC | DBI | CHI |
| HG WL Subtree | 0.5573 | 0.4673 | 0.5815 | 0.6566 | 423.515 |
| HG WL Hyedge | 0.6722 | 0.6219 | **0.7336** | **0.4026** | **992.522** |
| HG SCKernelv2 | **0.6768** | **0.6369** | 0.6388 | 0.5806 | 522.210 |
| RHG-10 | NMI | ARI | SC | DBI | CHI |
| HG WL Subtree | 0.6875 | 0.4702 | 0.3829 | 1.0355 | 172.072 |
| HG WL Hyedge | 0.5824 | 0.2007 | **0.4861** | **0.6267** | 171.560 |
| HG SCKernelv2 | **0.7065** | **0.5324** | 0.4012 | 1.1346 | **182.518** |

The best results are marked in bold type.

vectors of equal dimensions. For HG SCKernelv1, we concatenated the graph features obtained by HGNN$^+$ [45] to form the final hypergraph-level features. For HG SCKernelv2, we enhance vertex features with the probability of forming closed paths via $L$-step random walks for each vertex, concatenate them with original vertex features, and fed them into HGNN. Table IX shows that our HG SCKernel methods exhibit good scalability, serving as plug-and-play modules to enhance the expressiveness of HGNN, achieving an average 7.9% performance improvement over the baseline in these datasets.

TABLE IX
EXPERIMENT RESULTS OF ABLATION ON EXTENSION OF HGNN

| Dataset | Stream-Player | | IMDB-Wri-Genre | | IMDB-Dir-Genre | |
|---|---|---|---|---|---|---|
| | Acc. | F1_ma | Acc. | F1_ma | Acc. | F1_ma |
| HGNN | 0.5815 | 0.5736 | 0.4181 | 0.2867 | 0.7566 | 0.7118 |
| +$\phi_{\text{Cycle}}$ | 0.5839 | 0.5774 | 0.4232 | 0.3213 | 0.7733 | 0.7333 |
| +$\phi_{\text{P-Cycle}}$ | **0.5957** | **0.5888** | **0.4982** | **0.3827** | **0.7745** | **0.7403** |

The best results are marked in bold type.

## VII. DISCUSSION

### A. The Universality of Hypergraph Subtree-Cycle Kernel

As stated in [31], it is noted that the pattern space for graph Weisfeiler-Lehman method is the whole subtree space $\mathscr{T}^G$. In existing work [19], prove that the Hypergraph Weisfeiler-Lehman test generalizes the standard Weisfeiler-Lehman test when applied to simple graphs. This result can be further understood from the perspective of homomorphism numbers. when dealing with any 2-uniform hypergraph, there exists a bijection: $\omega_{\mathcal{T}}(\cdot) : \mathscr{T}_{\text{2-uni}} \to \mathscr{T}^G$, meaning that for every subtree in the 2-uniform hypergraph space, there is a correspondence between each subtree in the graph space, which $\forall \mathcal{T} \in \mathscr{T}_{\text{2-uni}}$:

$$\begin{aligned}\omega_{\mathcal{T}}\left(\mathcal{T}(v)\right) &= \omega_{\mathcal{T}}\left(\{v, \{\mathcal{T}(v), \mathcal{T}(v_1)\}, \ldots, \{\mathcal{T}(v), \mathcal{T}(v_m)\}\}\right), \\ &= \omega_{\mathcal{T}}(f_1(v, v_1, v_2, \ldots, v_m)), \\ &= \left\{v, \mathcal{T}^G(v_1), \mathcal{T}^G(v_2), \ldots, \mathcal{T}^G(v_m)\right\}, \\ &= \mathcal{T}^G(v) \in \mathscr{T}^G. \end{aligned} \tag{19}$$

Therefore, when HG WL algorithm processes ordinary graphs, the pattern space corresponding to the expressive ability is $\mathscr{T}^G$. Due to $\mathscr{T}_{\text{2-uni}} \cap \mathscr{C}_L = \varnothing$ and $\mathscr{T}^G \cap \mathscr{C}_L^G = \varnothing$, so $\omega(\mathcal{T} \cap \mathcal{C}_l) = \omega_{\mathcal{T}}(\mathcal{T}) \cap \omega_{\mathcal{C}}(\mathcal{C}_l)$. So, we can only consider the existence of $\omega_{\mathcal{C}}(\cdot) : \mathscr{C}_L \to \mathscr{C}_L^G$, that is, $\forall \mathcal{C}_l \in \mathscr{C}_L$:

$$\begin{aligned}\omega_{\mathcal{C}}\left(\mathcal{C}_l\right) &= \omega_{\mathcal{C}}\left((v, e, v_1, e, v_2, \ldots, v)\right), \\ &= \omega_{\mathcal{C}}\left((v, (e, v_1), (e, v_2), \ldots, (e, v))\right), \\ &= (v, v_1, v_2, \ldots, v_m, v), \\ &= \mathcal{C}_l^G \in \mathscr{C}_L^G. \end{aligned} \tag{20}$$

We prove that the hypergraph SCKernel can degenerate into the graph kernel with pattern space $\mathcal{T}^G \cup \mathcal{C}_L^G$ confronting the simple graph structure. In the realm of graph kernel research, several methods such as GHC-T&C [32] and ID-GNN [30] have explicitly incorporated cycle-based topological features to enhance discriminative capability. have demonstrated through constructive counterexamples that their expressive ability strictly surpasses that of the Weisfeiler-Lehman kernel. In particular, our theoretical framework can systematically explain such empirical results by analyzing how cycle-aware structural information expands the representational capacity beyond the limitations imposed by traditional WL-based reasoning. Our HG SCKernel has the same expressiveness upper bound as them and can be considered an extension of these methods to hypergraphs.

### B. The Limitation of Comparison Framework via Homoprohism

This paper proposes a framework focuses on analyzing the pattern space associated with hypergraph kernels. The expressive ability of defines the theoretical upper bound of expressiveness within a given pattern space. Therefore, expressive ability cannot be measured when the pattern spaces corresponding to the two methods do not have an obvious nested relationship. However, the actual expressiveness also depends on the mechaisms used to process structural information such as processed, such as the message function in HGNN, pooling operations, or the ordering mechanisms outlined in SSWL [33]. Furthermore, through systematic analysis, this study categorizes existing hypergraph kernels into four fundamental categories: path-based, tree-based, cycle-based, and subhypergraph-based. However, our analysis is constrained by the limited exploration of subhypergraph kernels, particularly the lack of in-depth investigation into the pattern space corresponding to specific methodologies employed in these kernels. These limitations point to promising directions for future research, especially in developing a deeper practical algorithms for subhypergraph-based kernel design.

### C. The Limitation of Hypergraph Subtree-Cycle Kernel

The computational complexity of HG SCKernelv1 for extracting homomorphism numbers is $\mathcal{O}(|\bar{\mathcal{V}}| + |\bar{\mathcal{E}}|)^{2\,L}$. In contrast, HG SCKernelv2 replaces the hypergraph cycle feature in HG SCKernelv1 with $L$-step closed paths, resulting in a running time comparable to that of the Hypergraph WL kernel on small- to medium-sized hypergraphs. However, the paths returned by $L$ steps to the original vertices are not necessarily simple paths, and the probability that these paths are simple decreases as the value of $L$ increases. Moreover, HG SCKernelv2 generates graph features with $L$ steps through sum-pooling, leading to information loss during the cycle feature extraction process compared to HG SCKernelv1.

## VIII. Conclusion

In conclusion, this paper establishes a significant connection between homomorphism and hypergraph kernels, introducing a comparative framework that analyzes the expressiveness hierarchy of classic hypergraph kernels. This framework provides valuable theoretical insights that can guide the enhancement of kernel expressiveness. We further present the Hypergraph Subtree-Cycle Kernel, which demonstrates a robust ability to extract both tree-based and cycle-based hypergraph motifs, supported by two specific instances, HG SCKernelv1 and HG SCKernelv2. Notably, HG SCKernel exhibits outstanding performance across multiple graph and hypergraph datasets. Overall, our comparative framework offers advanced theoretical support for the design of hypergraph kernels, paving the way for future research and applications in complex data analysis.

## References


[1] K. A. Murgas, E. Saucan, and R. Sandhu, "Hypergraph geometry reflects higher-order dynamics in protein interaction networks," *Sci. Rep.*, vol. 12, no. 1, 2022, Art. no. 20879.

[2] P. S. Chodrow, N. Veldt, and A. R. Benson, "Generative hypergraph clustering: From blockmodels to modularity," *Sci. Adv.*, vol. 7, no. 28, 2021, Art. no. eabh1303.

[3] Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, and X. Wu, "Visual-textual joint relevance learning for tag-based social image search," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 363–376, Jan. 2013.

[4] Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai, "3-D object retrieval and recognition with hypergraph analysis," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4290–4303, Sep. 2012.

[5] Y. Feng, S. Ji, Y.-S. Liu, S. Du, Q. Dai, and Y. Gao, "Hypergraph-based multi-modal representation for open-set 3D object retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 4, pp. 2206–2223, Apr. 2024.

[6] N. Ruggeri, M. Contisciani, F. Battiston, and C. De Bacco, "Community detection in large hypergraphs," *Sci. Adv.*, vol. 9, no. 28, 2023, Art. no. eadg9159.

[7] I. Chien, C.-Y. Lin, and I.-H. Wang, "Community detection in hypergraphs: Optimal statistical limit and efficient algorithms," in *Proc. 21st Int. Conf. Artif. Intell. Statist.*, Apr. 2018, pp. 871–879.

[8] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-Lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, Nov. 2011.

[9] K. M. Borgwardt and H. P. Kriegel, "Shortest-path kernels on graphs," in *Proc. 5th IEEE Int. Conf. Data Mining*, 2005, pp. 74–81.

[10] F. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya, "Global graph kernels using geometric embeddings," in *Proc. 31st Int. Conf. Mach. Learn.*, Bejing, China, Jun. 2014, pp. 694–702.

[11] B. Rieck, C. Bock, and K. Borgwardt, "A persistent Weisfeiler-Lehman procedure for graph classification," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 5448–5458.

[12] L. Bai, P. Ren, and E. R. Hancock, "A hypergraph kernel from isomorphism tests," in *Proc. 22nd Int. Conf. Pattern Recognit.*, 2014, pp. 3880–3885.

[13] Z. Chen, L. Chen, S. Villar, and J. Bruna, "Can graph neural networks count substructures?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 10383–10395.

[14] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, "How powerful are k-hop message passing graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 4776–4790.

[15] T. Ma, Y. Qian, S. Zhang, C. Zhang, and Y. Ye, "Adaptive expansion for hypergraph learning," 2025, *arXiv:2502.15564*.

[16] G. Wachman and R. Khardon, "Learning from interpretations: A rooted kernel for ordered hypergraphs," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 943–950.

[17] J. Böker, "Color refinement, homomorphisms, and hypergraphs," in *Proc. Int. Workshop Graph-Theoretic Concepts Comput. Sci.*, 2019, pp. 338–350.

[18] G. Lee, J. Ko, and K. Shin, "Hypergraph motifs: Concepts, algorithms, and discoveries," *Proc. VLDB Endowment*, vol. 13, no. 11, pp. 2256–2269, 2020.

[19] Y. Feng, J. Han, S. Ying, and Y. Gao, "Hypergraph isomorphism computation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 3880–3896, May 2024.

[20] M. Togninalli, E. Ghisu, F. Llinares-López, B. Rieck, and K. Borgwardt, "Wasserstein Weisfeiler-Lehman graph kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6439–6449.

[21] N. M. Kriege, P.-L. Giscard, and R. Wilson, "On valid optimal assignment kernels and applications to graph classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1623–1631.
[22] C. Morris, K. Kersting, and P. Mutzel, "Glocalized Weisfeiler-Lehman graph kernels: Global-local feature maps of graphs," in *Proc. 2017 IEEE Int. Conf. Data Mining*, 2017, pp. 327–336.
[23] U. Kang, H. Tong, and J. Sun, "Fast random walk graph kernel," in *Proc. 2012 SIAM Int. Conf. Data Mining*, 2012, pp. 828–838.
[24] Z. Zhang, M. Wang, Y. Xiang, Y. Huang, and A. Nehorai, "RetGK: Graph kernels based on return probabilities of random walks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3968–3978.
[25] L. Hermansson, F. D. Johansson, and O. Watanabe, "Generalized shortest path kernel on graphs," in *Proc. 18th Int. Conf. Discov. Sci.*, Banff, AB, Canada, 2015, pp. 78–85.
[26] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 7134–7143.
[27] J. Bourgain, "On Lipschitz embedding of finite metric spaces in hilbert space," *Isr. J. Math.*, vol. 52, pp. 46–52, 1985.
[28] Q. F. Lotito, F. Musciotto, A. Montresor, and F. Battiston, "Higher-order motif analysis in hypergraphs," *Commun. Phys.*, vol. 5, no. 1, 2022, Art. no. 79.
[29] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.
[30] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 10737–10745.
[31] H. Dell, M. Grohe, and G. Rattan, "Lovász meets weisfeiler and Leman," in *Proc. 45th Int. Colloq. Automata Lang. Program.*, Dagstuhl, Germany, 2018, pp. 1–14.
[32] H. Nguyen and T. Maehara, "Graph homomorphism convolution," in *Proc. 37th Int. Conf. Mach. Learn.*, Jul. 2020, pp. 7306–7316.
[33] B. Zhang, G. Feng, Y. Du, D. He, and L. Wang, "A complete expressiveness hierarchy for subgraph GNNs via subgraph Weisfeiler-Lehman tests," in *Proc. 40th Int. Conf. Mach. Learn.*, Jul. 2023, pp. 41019–41077.
[34] B. Zhang, J. Gai, Y. Du, Q. Ye, D. He, and L. Wang, "Beyond Weisfeiler-Lehman: A quantitative framework for GNN expressiveness," in *Proc. 12th Int. Conf. Learn. Representations*, 2024, pp. 1–73.
[35] L. Lovász, "Operations with structures," *Acta Mathematica Hungarica*, vol. 18, no. 3/4, pp. 321–328, 1967.
[36] A. Krebs and O. Verbitsky, "Universal covers, color refinement, and two-variable counting logic: Lower bounds for the depth," in *Proc. 30th Annu. ACM/IEEE Symp. Log. Comput. Sci.*, 2015, pp. 689–700.
[37] J. Dıáz, M. Serna, and D. M. Thilikos, "Counting H-colorings of partial k-trees," *Theor. Comput. Sci.*, vol. 281, no. 1/2, pp. 291–309, 2002.
[38] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1601–1608.
[39] U. Chitra and B. Raphael, "Random walks on hypergraphs with edge-dependent vertex weights," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1172–1181.
[40] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowl. Inf. Syst.*, vol. 14, pp. 347–375, 2008.
[41] K. M. Borgwardt et al., "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. 1, pp. i47–i56, 2005.
[42] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, Clearwater Beach, FL, USA, Apr. 2009, pp. 488–495.
[43] U. Chitra and B. Raphael, "Random walks on hypergraphs with edge-dependent vertex weights," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1172–1181.
[44] K. Hayashi, S. G. Aksoy, C. H. Park, and H. Park, "Hypergraph random walks, Laplacians, and clustering," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 495–504.
[45] Y. Gao, Y. Feng, S. Ji, and R. Ji, "HGNN$^+$: General hypergraph neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3181–3199, Mar. 2023.

**Yifan Zhang** received the BE degree in 2023 from Xi'an Jiaotong University, Xi'an, China, where he is currently working toward the master's degree in artificial intelligence.

**Shaoyi Du** received double Bachelor degrees in computational mathematics and in computer science in 2002, the MS degree in applied mathematics in 2005, and the PhD degree in pattern recognition and intelligence system from Xi'an Jiaotong University, China, in 2009. He is currently a professor with Xi'an Jiaotong University. His research interests include computer vision, machine learning, and pattern recognition.

**Yifan Feng** received the BE degree in computer science and technology from Xidian University, Xi'an, China, in 2018, and the MS degree from Xiamen University, Xiamen, China, in 2021. He is currently working toward the PhD degree with the School of Software, Tsinghua University, Beijing, China. His research interests include hypergraph neural networks, machine learning, and pattern recognition.

**Shihui Ying** (Member, IEEE) received the BEng degree in mechanical engineering and the PhD degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2008, respectively. He is currently a professor with the Department of Mathematics, School of Science, Shanghai University, Shanghai, China. His research interests include geometric theory and methods for machine intelligence and medical image analysis.

**Yue Gao** (Senior Member, IEEE) received the BS degree from the Harbin Institute of Technology, Harbin, China, and the ME and PhD degrees from Tsinghua University, Beijing, China. He is currently an Associate Professor with the School of Software, Tsinghua University.