HyperGVL: Benchmarking and Improving Large Vision-Language Models in Hypergraph Understanding and Reasoning

Anonymous ACL submission

Abstract

Large Vision-Language Models (LVLMs) consistently require new arenas to guide their expanding boundaries, yet their capabilities with hypergraphs remain unexplored. In the real world, hypergraphs have significant practical applications in areas such as life sciences and social communities. Recent advancements in LVLMs have shown promise in understanding complex topologies, yet there remains a lack of a benchmark to delineate the capabilities of LVLMs with hypergraphs, leaving the boundaries of their abilities unclear. To fill this gap, in this paper, we introduce HyperGVL, the first benchmark to evaluate the proficiency of LVLMs in hypergraph understanding and reasoning. HyperGVL provides a comprehensive assessment of 12 advanced LVLMs across 84,000 vision-language question-answering (QA) samples spanning 12 tasks, ranging from basic component counting to complex NP-hard problem reasoning. The involved hypergraphs contain multiscale synthetic structures and real-world citation and protein networks. Moreover, we examine the effects of 12 textual and visual hypergraph representations and introduce a generalizable router WiseHyGR that improves LVLMs in hypergraph via learning adaptive representations. We believe that this work is a step forward in connecting hypergraphs with LVLMs.

1 Introduction

Graphs serve as a fundamental data structure for modeling the relationships between abstract concepts or tangible objects in the real world. Among the sub-categories of graphs, hypergraphs are significant because their hyperedges can effectively model high-order correlations among three or more entities. The applications of hypergraphs are prevalent in the real world. For example, in social networks, hypergraphs can naturally represent community interactions, where a hyperedge can connect an arbitrary number of vertices, reflecting the complex, high-order relationships within communities (Contisciani et al., 2022). Similarly, in life science, hypergraphs are adept at modeling interactions such as catalytic triplets in protein structures (Ravetz et al., 2019), where ordinary graphs that only focus on pair-wise relations fall short. Recent advances also shows the promising capability of hypergraph in modeling complex relationships among information in retrieved-augmented generation (Feng et al., 2025a) more accurately.

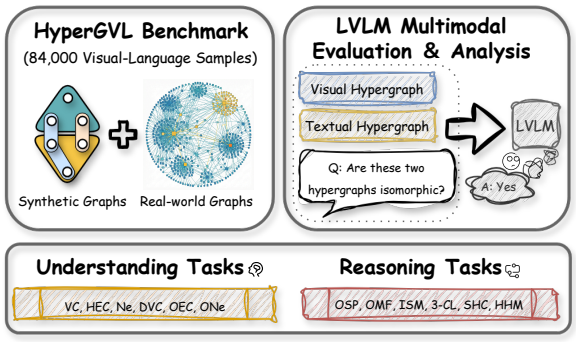

Figure 1: Overview of the HyperGVL benchmark.

On the other hand, large vision-language models (LVLMs) exhibit outstanding performance across a wide range of downstream tasks with human-like understanding and reasoning abilities (Li et al., 2025b). This triggers a growing interest in employing LVLMs for graph learning problems, as the vision modality offers a natural way for comprehending structural information and facilitating graph-related reasoning, where GVLQA (Wei et al., 2024), VisionGraph (Li et al., 2024), and VGCure (Zhu et al., 2025b) are among the first batch of such methods. However, they limit their scopes to ordinary graphs, and do not explore the potential of LVLMs on high-order relationships in hypergraphs.

To address this gap, we introduce HyperGVL (Fig. 1), the first comprehensive benchmark dataset designed to evaluate the capabilities of LVLMs on hypergraphs. HyperGVL consists of 84,000 vision-language question-answering (QA) pairs, cover-

Benchmark	Evaluation Capability	Graph Source	Structure Perception	High-Order	#Tasks	#Samples
GVLQA	Reasoning	Synthetic	Visual & Textual	✗	7	157,896
VisionGraph	Understanding & Reasoning	Synthetic	Visual	✗	10	3,000
VGCure	Understanding & Reasoning	Synthetic & Real-world	Visual	✗	22	223,646
LLM4Hypergraph	Mainly Understanding	Synthetic & Real-world	Textual	✓	15	21,500
HyperGVL (Ours)	Understanding & Reasoning	Synthetic & Real-world	Visual & Textual	✓	12	84,000

Table 1: Comparisons between HyperGVL and related graph analysis benchmarks for LVLMs/LLMs. *#Tasks*: number of response types; *#Samples*: total number of test samples; ✓/✗: support/not support high-order relationships.

ing both multiscale synthetic hypergraphs and real-world hypergraphs from citation and protein networks. The evaluation spans 12 tasks of varying difficulty levels, from fundamental hypergraph component understanding to the challenging NP-hard problems reasoning. Additionally, HyperGVL integrates seven textual and five visual representations of hypergraphs, offering insights into task preferences and model capability boundaries across these diverse representations.

Based on the performances of LVLMs under different hypergraph representations, we train WiseHyGR, a generalizable router that can choose proper hypergraph representations for given hypergraph problems. Experimental results validate that WiseHyGR generally enhances the hypergraph understanding and reasoning abilities of LVLMs, and the benefits can be generalized to downstream out-of-domain tasks.

The contributions of this work are threefold.

- We construct the HyperGVL benchmark, a new arena to assess LVLMs' abilities on hypergraph understanding and reasoning.
- We extensively evaluate 12 leading LVLMs on HyperGVL and expose their actual capabilities. The dedicated evaluations from various perspectives contribute 14 valuable observations.
- Based on the model performance across hypergraph representations, we train WiseHyGR, a generalizable router to boost LVLMs on hypergraph understanding and reasoning tasks.

2 The HyperGVL Benchmark

In this section, we introduce the HyperGVL Benchmark, which is designed to delineate the ability boundaries of LVLMs in handling higher-order structures of hypergraphs.

2.1 Benchmark Uniqueness

Table 1 underscores the distinct role of HyperGVL within the landscape of existing benchmarks. Unlike conventional graph-related LVLM benchmarks, HyperGVL delves into higher-order relationships inherent in hypergraphs, surpassing the limitations of ordinary graphs. Additionally, in contrast to text-only hypergraph benchmarks for large language models (LLMs), HyperGVL integrates visual perception effects unique to LVLMs, as well as enhances task diversity and complexity through the inclusion of intricate reasoning challenges. More related works are introduced in Appendix A.

2.2 Hypergraph Organization

The hypergraphs in HyperGVL involve meticulous considerations to ensure a reasonable organization. First, the HyperGVL benchmark comprises an equal proportion of synthetic and real-world hypergraphs. Synthetic hypergraphs are generated using both random and regular-structured methods, providing a controlled environment for testing. In contrast, real-world hypergraphs are from anonymized citation and protein networks, offering practical insights into real-world applications.

To ensure balanced complexity for comprehensive evaluation, we employed the scale partition protocol from Feng et al. (2025b), and organized hypergraphs by vertex count into three scale groups: small, medium, and large, with a distribution ratio of 1:2:1. This categorization facilitates the assessment of model performance across varying levels of complexity. Detailed descriptions of the processes used to obtain these hypergraphs are provided in Appendix B.

2.3 Benchmark Tasks

In this section, we introduce the tasks with design considerations in HyperGVL.

2.3.1 Design Principle

The tasks in HyperGVL are designed around two core dimensions: **assessed capability** and **response type**.

For **assessed capabilities**, tasks are divided into two main categories: understanding and reasoning. The understanding tasks evaluate three key atomic abilities: (1) *basic element capture*, which

involves recognizing vertices and hyperedges; (2) *adjacency perception*, which entails understanding adjacency relationships among vertices; and (3) *heuristic computation*, which includes computing heuristics such as vertex degree and hyperedge order (i.e., the number of vertices in a hyperedge). On the other hand, reasoning tasks assess model abilities in terms of (1) *algorithms*, which involve solving problems with definitive algorithms, and (2) *planning*, where problems are NP-hard and lack definitive algorithms, requiring models to actively plan and devise valid solutions.

Based on these assessed capabilities, all tasks are categorized into a four-level **difficulty** hierarchy: Level-1 (querying single atomic capability), Level-2 (combining compound atomic capabilities), Level-3 (polynomial-solvable algorithms), and Level-4 (NP-hard planning). This stratification aligns with task complexity (Bylander, 1994), and we aim to verify whether it is consistent with the actual capability spectrum of LVLMs.

For **response types**, tasks are categorized into four types: (1) *counting*, (2) *computing*, (3) *decision*, and (4) *descriptive* tasks. This taxonomy enables a comprehensive evaluation of LVLMs across diverse cognitive processes.

Unlike `LLM4Hypergraph` (Feng et al., 2025b), which presents relatively simple tasks for recent models (e.g., Gemini-3 Flash achieved over 90% zero-shot accuracy in its 13 out of 15 tasks in our testing), the proposed benchmark introduces more challenging tasks that require reasoning beyond structural understanding. This design aligns with evolving model capabilities and leaves ample room for their future improvement. Overall, the task distribution in `HyperGVL` encompasses a wider range of difficulty and diversity, establishing a comprehensive evaluation framework for LVLMs.

2.3.2 Task Descriptions

All tasks in `HyperGVL` are introduced briefly in this section. More details are in Appendix C.

Hypergraph understanding tasks are designed to evaluate in terms of the composition, topology, and fundamental heuristics of hypergraphs. Those tasks are mainly categorized into six types as follows.

- *Vertex Counting (VC)*: Counting the number of vertices in a given hypergraph.
- *Hyperedge Counting (HEC)*: Counting the number of hyperedges in a given hypergraph.
- *Neighbors (Ne)*: Identifying direct neighbors of a specified vertex connected by hyperedges.
- *Degree-specified Vertex Counting (DVC)*: Counting vertices with a specific degree value in the hypergraph.
- *Order-specified Hyperedge Counting (OEC)*: Counting hyperedges with a specific order in the hypergraph.
- *Order-filtered Neighbors (ONe)*: Identifying neighbors of a vertex when only considering hyperedges with their orders no smaller than a specified threshold.

The associated **assessed capabilities**, **difficulty levels**, and **response types** of those tasks are detailed at the top of Tab. 2, along with examples.

Hypergraph reasoning tasks are designed to tackle complex, multi-step inferential challenges within hypergraphs. Beyond understanding hypergraph structures and computing heuristics, these tasks require organizing the atom capabilities into a sophisticated iterative process to tackle complex hypergraph problems. Those tasks are mainly classified into six types as follows.

- *Order-weighted Shortest Path (OSP)*: Computing the shortest path length between two vertices, where the hyperedge order serves as the distances.
- *Order-weighted Maximum Flow (OMF)*: Computing the maximum flow between two vertices, where the hyperedge order determines the edge capacity.
- *Isomorphism Recognition (ISM)*: Determining whether two hypergraphs are isomorphic.
- *Hypergraph 3-Coloring (3-CL)*: Providing a valid 3-coloring, where each hyperedge contains at least two distinct colors.
- *Strict Hypercycle (SHC)*: Searching a strict hypercycle in the hypergraph, where adjacent hyperedges share exactly one vertex.
- *Hypergraph Hamilton Path (HHM)*: Planning a Hamiltonian path, which visits all the vertices in the hypergraph exactly once, with a given vertex as the starting point of the path and another one as the ending point.

The associated **assessed capabilities**, **difficulty levels**, and **response types** of those tasks are detailed in the bottom of Tab. 2.

Task	Capability	Response Type	Difficulty	Example	#Sample
Understanding Tasks					42,000
VC	Element	Counting	Level-1	Q: How many vertices are in the hypergraph G? A: 15.	7,000
HEC	Element	Counting	Level-1	Q: How many vertices are in the hypergraph G? A: 23.	7,000
Ne	Adjacency	Descriptive	Level-1	Q: What are the direct neighbors of vertex v4 in hypergraph G? A: v0, v3, v5.	7,000
DVC	Heuristic&Element	Counting	Level-2	Q: How many vertices have degree 3 in hypergraph G? A: 7.	7,000
OEC	Heuristic&Element	Counting	Level-2	Q: How many hyperedges have order 4 in hypergraph G? A: 8.	7,000
ONe	Heuristic&Adjacency	Descriptive	Level-2	Q: What are the neighbors of vertex v5 when only considering hyperedges with order >= 2 in hypergraph G? A: v0, v3.	7,000
Reasoning Tasks					42,000
OSP	Algorithm	Computing	Level-3	Q: What is the order-weighted shortest path length from v4 to v8? A: 8.	7,000
OMF	Algorithm	Computing	Level-3	Q: What is the order-weighted maximum flow from v4 to v8? A: 19.	7,000
ISM	Algorithm	Decision	Level-3	Q: Are these two hypergraphs isomorphic? A: Yes.	7,000
3-CL	Planning	Descriptive	Level-4 (NP-hard)	Q: Please provide a 3-coloring strategy such that each hyperedge contains nodes with at least 2 different colors. A: Coloring:[v0:c0,v1:c1,v2:c2,v3:c0,v4:c1,v5:c2].	7,000
SHC	Planning	Descriptive	Level-4 (NP-hard)	Q: Please identify a strict hypercycle in the hypergraph G. A: Cycle:[e0,e3,e2,e4,e1].	7,000
HHM	Planning	Descriptive	Level-4 (NP-hard)	Q: Please provide a valid Hamiltonian path from v1 to v0. (Hamiltonian path = path visiting all vertices exactly once). A: Path:[e0,e1,e2,e3].	7,000

Table 2: Properties, statistics, and examples of all hypergraph understanding and reasoning tasks in HyperGVL.

2.4 Hypergraph Representations

Hypergraph representations are crucial for evaluating the capabilities of LVLMs within hypergraphs, as distinct representations introduce unique perceptual biases (Wei et al., 2024; Feng et al., 2025b). Unlike LLMs that rely solely on text, LVLMs benefit from a synergistic perception of both visual and textual information. Therefore, testing LVLMs on HyperGVL should not only consider textual or visual hypergraph representations individually, but also study their combinations. To comprehensively assess LVLMs across different representations, HyperGVL utilizes seven textual representations and five visual representations, resulting in 35 combinations of hypergraph representations (7 textual $\times$ 5 visual). To clarify, the 84,000 vision-language QA samples in HyperGVL are related to 2,400 meta problems (each task 200), where each meta problem p corresponds to 35 vision-language QA samples with varying hypergraph representations.

In the following, we introduce these hypergraph representations, and provide examples in Fig. 2.

Textual Hypergraph Representations. By following (Feng et al., 2025b), we use the following 7 textual hypergraph representations in HyperGVL. Their prompts are demonstrated in Appendix E.1:

- *Low-Order Incidence Description (LO-Inc)*: Listing pairwise vertex connections, e.g., "Vertex v_1 is linked to vertices v_2 and v_3".
- *Neighbor-Pair Description (N-Pair)*: Enumerating pairs of vertices sharing a hyperedge, e.g., "$(v_1, v_2), (v_1, v_3)$".
- *Raw Adjacency Matrix Description (Adj-Mat)*: Employing a matrix where the binary entry indicates the connection between vertex pairs.
- *High-Order Neighbor Description (HO-Neigh)*: Detailing neighborhood information in two parts, outlining vertex-to-hyperedge and hyperedge-to-vertex connections.
- *High-Order Incidence Description (HO-Inc)*: Incorporating higher-order relationships into LO-Inc, e.g., "Vertex v_1 is connected to vertices v_2 and v_3 via hyperedge e_1".
- *Neighbor-Set Description (N-Set)*: Listing complete vertex sets associated with each hyperedge, e.g., "(v_1, v_2, v_3)".
- *Raw Incidence Matrix Description (Inc-Mat)*: Uses a matrix where the binary entry signifies the inclusion of a vertex in a hyperedge.

Visual Hypergraph Representations. For the vision branch, HyperGVL incorporates five visual hypergraph representations as follows, with details of their generations in Appendix D and examples in Fig. 2.

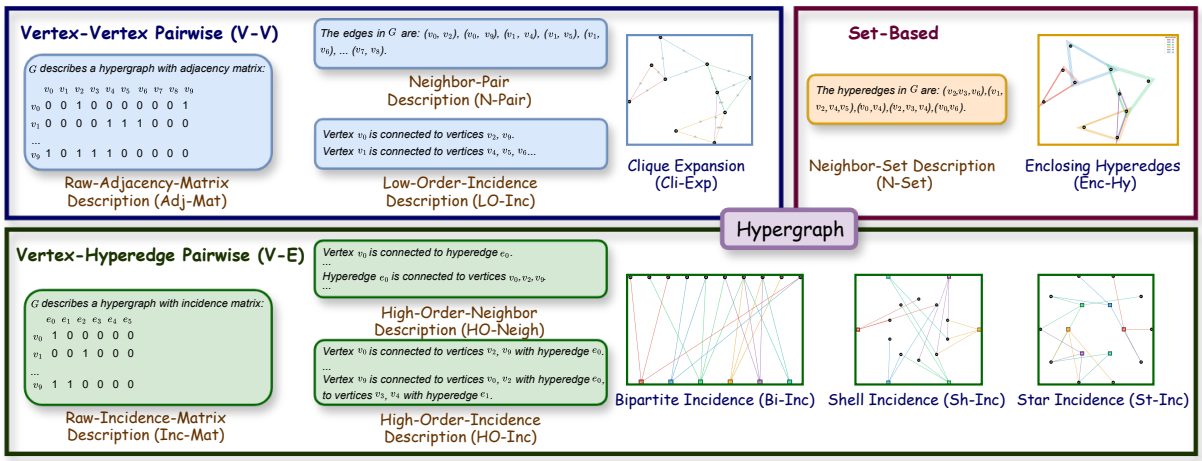

Figure 2: The 7 textual hypergraph representations and 5 visual hypergraph representations in HyperGVL.

- *Enclosing Hyperedges (Enc-Hy)*: Visualizing hyperedges as geometric enclosures, with vertices inside enclosures indicating membership in the corresponding hyperedge.

- *Bipartite Incidence (Bi-Inc)*: Modeling hypergraphs as bipartite structures, with vertices and hyperedges as two disjoint sets arranged in separate layers. Straight lines between layers depict vertex-hyperedge membership.

- *Shell Incidence (Sh-Inc)*: Similar to the *Bipartite Incidence* representation, but arranging vertices on an outer circle and places hyperedges internally, with radial connections depicting the membership.

- *Star Incidence (St-Inc)*: Similar to Bipartite Incidence, but treating each hyperedge as a star center connected to its member vertices.

- *Clique Expansion (Cli-Exp)*: Transforming hyperedges into vertex-vertex pairwise connections, with hyperedge IDs labeled to preserve hypergraph semantics.

Based on the above encoding approaches, those textual and visual hypergraph representations are categorized into three representation categories: **(1) Vertex-Vertex Pairwise (V-V)** category, which includes *LO-Inc, N-Pair, Adj-Mat*, and *Cli-Exp*, focuses on vertex-vertex neighborships with hyperedge semantics implied through identifiers; **(2) Vertex-Hyperedge Pairwise (V-E)** category, comprising *HO-Neigh, HO-Inc, Inc-Mat, Bi-Inc, Sh-Inc*, and *St-Inc*, explicitly decomposes high-order relationships into pairwise mappings from vertices to their associated hyperedges; **(3) Set-Based** category, featuring *N-Set* and *Enc-Hy*, treats hyperedges as holistic units, introducing their constituent vertices together.

3 Benchmarking LVLMs on HyperGVL

3.1 Experimental Setup

We conduct evaluations on 12 leading LVLMs, including open-source LVLMs: LLaVA1.5-7B (Liu et al., 2023), LLaVA-Next-7B (Liu et al., 2024), Molmo-7B-D-0924 (Deitke et al., 2025), InternVL2.5-8B (Chen et al., 2024), InternVL3-8B (Zhu et al., 2025a), GLM-4V-9B (GLM et al., 2024), Qwen2.5-VL-7B (Bai et al., 2025b), Qwen3-VL-8B (Bai et al., 2025a), Gemma3-12B (Team et al., 2025), and Deepseek-VL2-27B (Wu et al., 2024) as well as GPT-4o (OpenAI, 2024) and Gemini-3 Flash (Google, 2025), which are strong closed-source LVLMs. For all LVLMs, the **zero-shot** setting is adopted for the evaluation. The LVLMs are required to provide answers in predefined formats, but the factually accurate answers with incorrect formats are also considered to be correct after manual review. Prompts used in the evaluation are detailed in Appendix E.2. More details of experimental settings are in Appendix F.

3.2 Main Results

Tab. 3 shows the performance of various LVLMs on HyperGVL, with results averaged across all hypergraph representation combinations. Key observations include:

Observation 1: Basic hypergraph components and neighborhood understanding (Level-1 VC, HEC, and Ne) could be challenging for open-source LVLMs. Advanced open-source LVLMs like Qwen3-VL and Gemma3 excel in vertex counting (VC), but they are still unsatisfactory in capturing hyperedges (HEC) and neighborhoods (Ne). In

Models	Understanding							Reasoning						
	VC	HEC	Ne	DVC	OEC	ONe	Avg.U	OSP	OMF	ISM	3-CL	SHC	HHM	Avg.R
LLaVA-1.5-7B	14.10	8.63	2.64	10.74	16.57	1.61	9.05	5.57	4.51	44.53	10.04	8.44	0.57	12.28
LLaVA-Next-7B	19.56	11.17	6.07	8.83	14.59	8.39	11.44	6.83	3.84	42.27	15.84	2.83	0.31	11.99
Molmo-7B-D-0924	42.24	17.94	7.96	13.24	15.81	14.40	18.60	10.53	6.74	45.90	6.63	6.80	0.40	12.83
InternVL2.5-8B	73.34	22.93	10.73	15.93	18.26	22.00	27.20	10.94	6.70	51.06	22.76	5.94	0.69	16.35
GLM-4V-9B	91.34	39.37	15.46	17.89	23.33	18.90	34.38	13.51	8.89	55.54	20.57	14.19	0.67	18.90
InternVL3-8B	88.93	47.93	39.17	27.47	35.71	30.54	44.96	16.23	11.69	50.54	38.04	5.24	1.41	20.52
Qwen2.5-VL-7B	95.41	41.94	36.19	24.63	29.01	27.17	42.39	15.73	14.53	59.23	26.31	10.51	1.50	21.30
Qwen3-VL-8B	99.60	59.41	55.20	52.34	49.14	43.13	59.80	32.10	25.09	42.79	57.07	46.57	2.93	34.43
Gemma3-12B	99.80	57.91	73.34	51.92	47.35	50.87	63.53	29.41	22.71	51.29	33.69	13.39	2.67	25.53
Deepseek-VL2-27B	48.59	17.73	2.79	9.81	10.60	5.11	15.77	3.54	3.07	40.76	1.40	0.43	0.03	8.21
GPT-4o	98.29	80.29	83.43	56.57	50.57	82.86	66.91	39.43	26.29	59.14	56.86	25.14	3.13	35.00
Gemini-3 Flash	99.73	98.31	88.20	77.11	77.83	97.13	89.72	82.09	76.29	64.60	79.34	66.44	3.50	62.04
Average	72.58	41.96	37.71	30.54	32.40	30.90	40.31	22.16	17.53	50.64	30.71	17.16	1.48	23.28

Table 3: Model performance (Acc) on HyperGVL benchmark, where **Avg.U** and **Avg.R** represent the average accuracy across understanding and reasoning tasks, respectively. Open- and closed-source LVLMs are colored in . and . backgrounds. The best and worst results are colored in red and blue, respectively.

contrast, closed-source LVLMs perform robustly across these foundational capabilities.

Observation 2: Level-2 tasks (i.e., DVC, OEC, ONe) introduce complexity beyond Level-1 tasks by incorporating heuristic computations as constraints, leading to lower performance in DVC, OEC, and ONe than VC, HEC, and Ne as expected.

Observation 3: Reasoning tasks pose significant challenges for all LVLMs, with Avg.R across models (23.28%) notably lower than Avg.U (40.31%). Even the most powerful LVLM, Gemini-3 Flash, achieves only 62.04% in terms of *Avg.R*, while most open-source LVLMs score below 25%.

Observation 4: The hypergraph reasoning performance of LVLMs generally aligns with task complexity. Overall, Level-4 NP-hard tasks (i.e., 3-CL, SHC, HHM) are more challenging than Level-3 tasks, with HHM as the most intractable, that is, no model exceeds 6% in terms of the accuracy on this task. This demonstrates that planning constitutes a complex capability that requires substantial improvement for current LVLMs. 3-CL is a bit easier due to its abundance of valid solutions, which can be derived via local constraints. In contrast, SHC and HHM require global consistency, feature far fewer valid solutions, and are highly susceptible to intermediate inference errors, rendering them significantly challenging for LVLMs.

Observation 5: All LVLMs perform well in ISM, which may attributes to the visual representations directly reveals similar patterns among isomorphic hypergraphs (illustrated in Appendix D.3).

Observation 6: Closed-source LVLMs significantly outperform open-source counterparts in both understanding and reasoning tasks, with the highest Avg.U reaching 89.72% for closed-source LVLMs (Gemini-3 Flash) compared to 63.53% for open-source LVLMs (Gemma3). For reasoning tasks, the disparity is 62.04% (Gemini-3 Flash) vs. 34.43% (Qwen3-VL) in terms of Avg.R. This highlights substantial room for improvement in the hypergraph understanding and reasoning capabilities of open-source LVLMs.

Observation 7: Parameter scale does not completely determine the performance of LVLMs. Smaller models are not always worse than larger ones. For example, Qwen3-VL-8B performs better at reasoning tasks than Gemma3-12B and Deepseek-VL2-27B. Moreover, there are significant differences even among models with comparable sizes (e.g., LLaVA-1.5-7B vs. Qwen3-VL-8B), suggesting that enhancing LVLM's hypergraph capabilities requires specialized approaches rather than merely increasing the model size.

3.3 Comparison of LVLMs and LLMs

We compare the performance of two representative LVLMs, GLM-4V and InternVL3, with their corresponding LLMs, GLM-4 and InternLM3.

Observation 8: As illustrated in Fig. 3, both LVLMs consistently outperform their LLMs on all understanding and reasoning tasks. This demonstrates the advantages of visual-textual synergy in LVLMs over text-only processing in LLMs on hypergraphs. These results further underscore the imperative of integrating LVLMs into hypergraph scenarios, a step that HyperGVL has initiated.

Representations		Understanding							Reasoning						
		VC	HEC	Ne	DVC	OEC	ONe	Avg.U	OSP	OMF	ISM	3-CL	SHC	HHM	Avg.R
V-V	LO-Inc	75.02	23.45	40.53	17.19	16.94	50.12	36.51	13.86	11.71	50.34	24.11	10.96	0.79	18.64
	N-Pair	75.00	14.90	32.92	18.52	17.27	33.64	31.34	17.15	13.43	50.54	19.12	15.26	0.80	19.40
	Adj-Mat	76.28	21.79	25.81	18.87	18.38	18.94	29.31	13.74	10.18	50.53	24.80	10.11	0.78	18.37
	Cli-Exp	70.39	40.04	38.00	31.31	32.29	31.02	39.81	22.98	17.76	56.17	31.15	18.06	1.55	24.62
	V-V Avg.	74.17	25.05	34.32	21.47	21.22	33.43	34.24	16.93	13.27	51.90	24.80	13.60	0.98	20.26
V-E	HO-Neigh	73.28	67.03	44.20	48.65	53.19	30.43	52.10	30.45	23.84	51.72	37.29	21.48	2.03	27.81
	HO-Inc	73.82	61.37	46.16	38.53	40.36	40.06	49.35	27.44	21.68	51.05	39.95	22.59	2.51	27.55
	Inc-Mat	63.80	40.42	27.78	36.59	26.64	10.05	33.51	19.80	17.15	50.05	30.29	16.65	1.58	22.60
	Bi-Inc	70.25	40.95	38.08	30.21	32.56	31.02	39.81	21.67	17.49	47.36	30.43	16.58	1.51	22.52
	Sh-Inc	74.43	41.57	37.20	29.73	31.63	30.74	40.18	21.88	17.28	47.58	29.77	16.30	1.39	22.38
	St-Inc	74.72	42.00	37.32	30.36	32.04	30.45	40.45	22.03	17.37	46.96	30.69	16.43	1.41	22.30
	V-E Avg.	71.72	48.89	38.46	35.68	36.07	28.79	42.57	23.88	19.14	49.12	33.07	18.34	1.74	24.19
Set-Based	N-Set	70.89	64.74	46.55	35.40	54.02	33.09	50.08	32.70	24.72	49.67	39.65	21.92	1.85	28.44
	Enc-Hy	73.11	45.23	37.92	31.07	33.48	31.29	41.32	22.26	17.76	55.72	31.26	19.61	1.53	24.70
	Set-Based Avg.	72.00	54.99	42.24	33.24	43.75	32.19	45.70	27.48	21.24	52.70	35.46	20.77	1.69	26.57

Table 4: Average performance (Acc) with different hypergraph representations. Textual and visual representations are distinguished by . and . backgrounds. The best and worst performance are colored in red and blue.

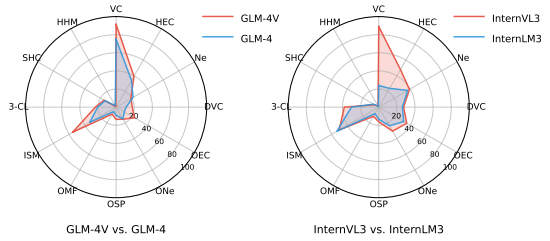

Figure 3: Comparison between the capability boundaries of LVLMs (GLM-4V & InternVL3) and corresponding LLMs (GLM-4 & InterLM3).

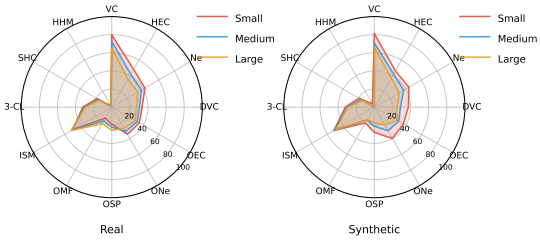

Figure 4: Performance for real-world/synthetic hypergraphs at different sizes.

3.4 Results on Hypergraph Representations

Table 4 presents the average performance across various hypergraph representations. The results for each textual representation are averaged over all LVLMs and visual representations, while the results for each visual representation are averaged over all LVLMs and textual representations. Key observations include:

Observation 9: Among textual representations, 'N-Set' demonstrates superior average performance in both understanding and reasoning tasks. Similarly, among visual representations, 'Enc-Hy' excels in both areas. Both representations are set-based, underscoring the overall importance of their holistic hyperedge expression.

Observation 10: Representation-induced variability differs substantially across modalities. **Textual** encodings yield a much wider performance range (higher ceiling and lower floor), while **visual** encodings lead to comparatively more moderate variations, indicating that LVLMs are more representation-sensitive in the text branch and more representation-robust in the visual branch.

Observation 11: The V–V pairwise encodings are comparatively more suitable for vertex-level primitives (e.g., VC and neighbor-related queries), but they consistently lag behind when tasks require explicit hyperedge semantics (e.g., hyperedge cardinality/order and planning with global constraints). This suggests that LVLMs remain limited in recovering high-order hyperedge information from purely vertex–vertex adjacency.

Observation 12: *HO-Inc* is the top-performing representation on planning tasks (e.g., 3-CL, SHC, HHM). We attribute this to its explicit vertex–hyperedge–vertex modeling with hyperedge IDs (e.g., "v_1 is connected to v_2, v_3 via hyperedge e_1"), which supports step-wise navigation and constraint checking during multi-step solution construction.

Observation 13: Within textual representations, matrix-style encodings (i.e., Adj-Mat and Inc-Mat) are generally *less competitive* than list/set-style encodings within the same family, likely due to the sparsity and indexing burdens of matrix representa-

tions hinder reliable extraction of vertex–hyperedge memberships.

3.5 Impact of Hypergraph Scale

Fig. 4 shows the performance of real-world and synthetic hypergraphs at different sizes. We have the following observation:

Observation 14: As the hypergraph size increases, the performance on synthetic hypergraphs always deteriorates. This trend is more pronounced for understanding tasks. For real-world hypergraphs, OSP and OMF have better performance on the larger hypergraphs, showing benefits from their more regular connectivity structures.

4 WiseHyGR Router

Inspired by our findings on the critical impacts of hypergraph representations, we develop `WiseHyGR`, a generalizable router aimed at enhancing the LVLM performance on hypergraph understanding and reasoning tasks by intelligently selecting suitable hypergraph representations.

First, we construct a Problem-Representation Mapping (PRM) dataset $\{(p, R_p)\}$, where R_p is the representation combination with the highest average accuracy for each vision-language QA set related to the meta problem p in `HyperGVL`. When multiple combinations yield the same accuracy, we include multiple (p, R_p) pairs.

Treating as a classification task, we use the PRM dataset to train a DeBERTaV3-base (He et al., 2021) router named `WiseHyGR`, with the training and validating split in a ratio of 8:2. The hypergraph in meta problem p is converted to the HO-Neigh representation, chosen for its superior performance in understanding tasks according to Tab. 4. This hypergraph HO-Neigh description, along with the problem question, serves as input to `WiseHyGR`, with the label being the optimal representation combination.

We evaluate `WiseHyGR` based on an open-source LVLM (i.e., Qwen3-VL-8B) and a closed-source LVLM (i.e., GPT-4o). Baselines include: (1) ***Rand***, which randomly samples one of the 35 representation combinations, simulating the unguided choice; and (2) ***Top***, which uses the most effective textual and visual representations according to Tab. 4 (i.e., N-Set and Enc-Hy). The evaluation involves two settings.

(1) In-domain hypergraph capabilities : We generate 1,000 problems per task to assess improvements by `WiseHyGR` in hypergraph understanding and reasoning. As shown in Fig. 5, representations chosen by `WiseHyGR` consistently outperform the baselines across all tasks on both LVLMs, demonstrating its effectiveness in enhancing LVLM capabilities in hypergraph tasks.

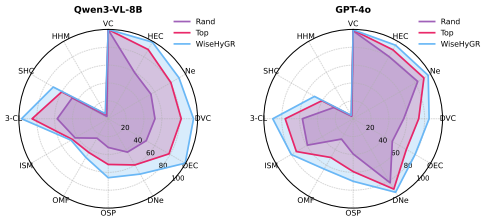

Figure 5: In-domain hypergraph capabilities of LVLMs with WiseHyGR.

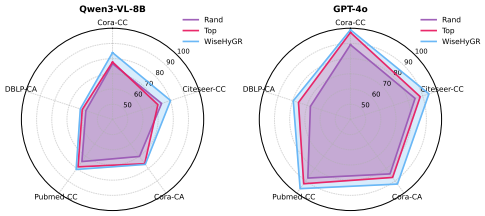

Figure 6: Out-of-domain hypergraph node classification of LVLMs with WiseHyGR.

(2) Out-of-domain hypergraph applications : We further evaluate LVLMs on hypergraph node classification, where `WiseHyGR` serves as a plug-and-play enhancer. Details of the experimental setting and datas are in Appendix G. As shown in Fig. 6, despite being trained solely on `HyperGVL` tasks and having no exposure to hypergraph node classification tasks, `WiseHyGR` consistently improves the zero-shot performance of both LVLMs. This shows that `WiseHyGR` captures the hidden representation preferences embedded in the semantic requirements of tasks with generalizability.

5 Conclusion

We present `Hyper-GVL`, the pioneering benchmark for integrating LVLMs into hypergraph scenarios. `Hyper-GVL` includes 84,000 vision-language QA pairs on synthetic and real-world hypergraphs, spanning 12 tasks that test various capabilities and difficulty levels. Our comprehensive evaluation of 12 leading LVLMs provides insightful findings. Additionally, we investigate the impact of diverse textual and visual hypergraph representations and introduce `WiseHyGR`, a plug-and-play router that enhances LVLMs in hypergraph with generalizability. This work pushes the boundary of LVLMs, marking their initial foray into hypergraph processing.

Limitations

While HyperGVL encompasses 12 representative hypergraph tasks to establish robust capability benchmarks, it naturally cannot cover all hypergraph response types, specifically some domain-specific customized tasks, due to our focus on a general usable benchmark.

Additionally, resource limitations prevent us from evaluating extremely large-scale models such as Qwen3-VL-235B (Bai et al., 2025a) and InternVL3-78B (Zhu et al., 2025a), despite their potentially greater capabilities on hypergraph understanding and reasoning.

References

Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. 1994. *Network flows: theory, algorithms and applications*. Prentice hall.

Qihang Ai, Jiafan Li, Jincheng Dai, Jianwu Zhou, Lemao Liu, Haiyun Jiang, and Shuming Shi. 2024. Advancement in graph understanding: A multimodal benchmark and fine-tuning of vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7485–7501.

László Babai. 2016. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025b. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Claude Berge. 1985. *Graphs and hypergraphs*. Elsevier Science Ltd.

J-C Bermond. 1978. Hamiltonian decompositions of graphs, directed graphs and hypergraphs. In *Annals of Discrete Mathematics*, volume 3, pages 21–28. Elsevier.

Alain Bretto. 2013. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1:209–216.

Tom Bylander. 1994. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Zhixuan Chu, Yan Wang, Qing Cui, Longfei Li, Wenqing Chen, Zhan Qin, and Kui Ren. 2024. Llm-guided multi-view hypergraph learning for human-centric explainable recommendation. *arXiv preprint arXiv:2401.08217*.

Martina Contisciani, Federico Battiston, and Caterina De Bacco. 2022. Inference of hyperedges and overlapping communities in hypergraphs. *Nature communications*, 13(1):7229.

France Dacar. 1998. The cyclicity of a hypergraph. *Discrete Mathematics*, 182(1-3):53–67.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, and 1 others. 2025. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 91–104.

Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*.

Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264.

Yifan Feng, Hao Hu, Xingliang Hou, Shiquan Liu, Shihui Ying, Shaoyi Du, Han Hu, and Yue Gao. 2025a. Hyper-rag: Combating llm hallucinations using hypergraph-driven retrieval-augmented generation. *arXiv preprint arXiv:2504.08758*.

Yifan Feng, Chengwu Yang, Xingliang Hou, Shaoyi Du, Shihui Ying, Zongze Wu, and Yue Gao. 2025b. Beyond graphs: Can large language models comprehend hypergraphs? In *The Thirteenth International Conference on Learning Representations*.

Michael R Garey, David S Johnson, and 1 others. 1990. A guide to the theory of np-completeness. *Computers and intractability*, pages 37–79.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Google. 2025. Gemini-3 Pro. Technical report.

Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Ruochang Li, Xiao Luo, Zhiping Xiao, Wei Ju, and Ming Zhang. 2025a. Heal: Hybrid enhancement with llm-based agents for text-attributed hypergraph self-supervised representation learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6815–6829.

Yunxin Li, Baotian Hu, Haoyuan Shi, Wei Wang, Longyue Wang, and Min Zhang. 2024. Visiongraph: Leveraging large multimodal models for graph theory problems in visual context. *arXiv preprint arXiv:2405.04950*.

Zongxia Li, Xiyang Wu, Hongyang Du, Huy Nghiem, and Guangyao Shi. 2025b. Benchmark evaluations, applications, and challenges of large vision language models: A survey. *arXiv preprint arXiv:2501.02189*, 1.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024. Llava-next: Improved reasoning, ocr, and world knowledge.

OpenAI. 2024. GPT-4o. Technical report.

Benjamin D Ravetz, Andrew B Pun, Emily M Churchill, Daniel N Congreve, Tomislav Rovis, and Luis M Campos. 2019. Photoredox catalysis using infrared light via triplet fusion upconversion. *Nature*, 565(7739):343–346.

António J M Ribeiro, Gemma L Holliday, Nicholas Furnham, Jonathan D Tyzack, Katherine Ferris, and Janet M Thornton. 2018. Mechanism and catalytic site atlas (m-csa): a database of enzyme reaction mechanisms and active sites. *Nucleic acids research*, 46(D1):D618–D623.

Nicolò Ruggeri, Martina Contisciani, Federico Battiston, and Caterina De Bacco. 2023. Community detection in large hypergraphs. *Science Advances*, 9(28):eadg9159.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Sandro Tosi. 2009. *Matplotlib for Python developers*. Packt Publishing Ltd.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, and 1 others. 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Vitaly Ivanovich Voloshin. 2002. *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications: Theory, Algorithms, and Applications*. 17. American Mathematical Soc.

Yanbin Wei, Shuai Fu, Weisen Jiang, Zejian Zhang, Zhixiong Zeng, Qi Wu, James Kwok, and Yu Zhang. 2024. Gita: Graph to visual and textual integration for vision-language graph reasoning. *Advances in Neural Information Processing Systems*, 37:44–72.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, and 1 others. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Yao Xu, Shizhu He, Jiabei Chen, ZengXiangrong ZengXiangrong, Bingning Wang, Guang Liu, Jun Zhao, and Kang Liu. 2025. Llasa: Large language and structured data assistant. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1935–1946.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016a. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016b. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025a. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

Yingjie Zhu, Xuefeng Bai, Kehai Chen, Yang Xiang, Jun Yu, and Min Zhang. 2025b. Benchmarking and improving large vision-language models for fundamental visual graph understanding and reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30678–30701.

A Related Works

Benchmarking LVLMs on Graphs. VisionGraph (Li et al., 2024) and GVLQA (Wei et al., 2024) evaluate the problem-solving capabilities of LVLMs in graph theory problems. VisionGraph assumes that visual graphs are naturally equipped, while GVLQA generates visual graphs from scratch. Both benchmarks include numerous synthetic visual graphs and complex graph theory problems. Ai et al. (2024) introduces a multimodal instruction-following benchmark designed to assess LVLMs' understanding and reasoning capabilities on real-world graph images across various domains. VGCURE (Zhu et al., 2025b) establishes fundamental understanding and reasoning benchmarks tailored to explore how LVLMs understand and reason on visual graphs.

However, these benchmarks are limited to ordinary graphs, leaving LVLMs' capabilities in handling high-order relationships in hypergraphs unexplored. Hypergraphs have substantial practical value in domains such as retrieved-augmented generation (RAG) (Feng et al., 2025a), life sciences (Ravetz et al., 2019), and community analysis (Ruggeri et al., 2023). To address this gap, we propose `HyperGVL`, the first dedicated benchmark for evaluating LVLMs' capabilities in hypergraph understanding and reasoning.

Large Models for Hypergraphs In the realm of benchmarking, the sole existing benchmark for evaluating large models' hypergraph capabilities is `LLM4Hypergraph`, which rigorously assesses LLMs' proficiency in understanding hypergraph structures. Our proposed `HyperGVL` diverges from `LLM4Hypergraph` in two pivotal aspects: 1) `HyperGVL` is tailored for LVLMs, where visual components complement textual information, yielding unique informational gains and expanding capability boundaries. 2) We have augmented task complexity and diversity, demanding models to engage in sophisticated reasoning based on hypergraph comprehension. This approach not only meets the evolving requisites of advanced models but also ensures comprehensive coverage of difficulty levels and evaluation capabilities.

Beyond benchmark creation, recent studies have delved into the role of large models in hypergraph-related learning and reasoning. HEAL (Li et al., 2025a) explores the use of LLM-based agents to enhance representation learning on text-attributed hypergraphs, positioning large language models as tools to improve data quality and structural signals under limited supervision. LLaSA (Xu et al., 2025) designs a hypergraph-aware LLM for knowledge grounding, integrating unified hypergraph representations into LLMs. LLMHG (Chu et al., 2024) employs large language models to generate structured user interest angles, facilitating multi-view hypergraph construction. These advancements collectively underscore the transformative potential of large models in hypergraph processing and their growing importance in the field.

B Hypergraph Generation

In this section, we introduce the hypergraph generation in our HyperGVL with details.

B.1 Synthetic Hypergraph

Synthetic hypergraphs are generated using our custom-developed scripts. For most tasks, we employ a randomized construction approach that creates connected hypergraphs by iteratively adding hyperedges of varying sizes and ensuring connectivity through breadth-first traversal. For NP-hard reasoning tasks such as SHC, HHM, and 3-CL, we utilize specialized constructors designed to ensure the existence of valid solutions. SHC instances are constructed with explicit hypercycle structures where consecutive hyperedges share exactly one vertex. HHM instances are formed using vertex permutations as backbone paths. Meanwhile, 3-CL instances are generated from pre-assigned color classes with constraint-preserving hyperedge selection. These scripts will be released together with our benchmark.

B.2 Real-world Hypergraph

Real-world hypergraphs are derived from the citation co-author network PubMed_CA (Yang et al., 2016a) and protein structure resources M-CSA (Ribeiro et al., 2018). In these hypergraphs, vertices represent anonymized authors or residues, while hyperedges correspond to paper-level co-author relationships or catalytic-site residue sets, respectively. To create scale-controlled instances while preserving real structural patterns, we extract sub-hypergraphs from the full hypergraphs by selecting a seed vertex and traversing incident hyperedges using a random walk, retaining visited vertices until reaching the target size. Finally, all vertex and hyperedge identifiers are re-indexed to canonical IDs. For tasks such as 3-CL, SHC, and

HHM, which require the hypergraph structure to ensure a valid solution exists, we repeat the sampling until the sampled hypergraph meets the requirements.

B.3 Hypergraph Structure Control

We employ the scale partition protocol from Feng et al. (2025b), categorizing hypergraphs by vertex count into three groups: *small* (5–10 vertices), *medium* (10–15 vertices), and *large* (15–20 vertices). The dataset consists of synthetic and real-world hypergraphs in a 1:1 ratio, with an internal distribution of small:medium:large set at 1:2:1. This classification aids in evaluating performance across increasingly complex scenarios.

For synthetic hypergraphs, hyperedge density is regulated by ensuring the number of hyperedges falls within the range of $[0.2|V|, 1.5|V|]$. This constraint prevents the generation of structures that are either trivially sparse or overly dense, enabling more informative capability assessments. Conversely, real-world hypergraphs do not impose a hyperedge-count constraint, allowing their hyperedge statistics to naturally reflect the connectivity patterns inherent in the original data.

C Task Details

In this section, we detail the 12 tasks in HyperGVL, categorized by their primary focus (understanding vs. reasoning) and difficulty level. Each task includes a formal problem definition, evaluation methodology, and implementation details from our codebase.

C.1 Understanding Tasks

Understanding tasks assess three core abilities of hypergraph comprehension for LVLMs: (1) *basic element capture*, recognizing vertices and hyperedges, (2) *adjacency perception*, understanding vertex adjacency relationships, and (3) *heuristic computation*, calculating heuristics like vertex degrees and hyperedge orders.

Level-1 tasks focus on individual basic element capture or adjacency perception, serving as foundational components for other understanding and reasoning tasks.

- **Vertex Counting (VC).** This task tests the model's ability to count vertices in a hypergraph. Given a hypergraph $\mathcal{G} = \{V, E\}$, the goal is to return $|V|$, the vertex set's cardinality. Evaluation is based on exact matches with pre-computed ground truth.

- **Hyperedge Counting (HEC).** This task assesses the model's ability to count hyperedges in a hypergraph. Given a hypergraph $\mathcal{G} = \{V, E\}$, the goal is to return $|E|$, the hyperedge set's cardinality. Unlike ordinary graph edge counting, this requires distinguishing each hyperedge from multiple connected vertices. Evaluation is based on exact matches with pre-computed ground truth.

- **Neighbors (Ne).** This task evaluates the model's ability to identify all direct neighbors of a given vertex. Given $\mathcal{G} = \{V, E\}$ and a vertex $u \in V$, the goal is to output the subset $\mathcal{N}(u) \subseteq V$ such that for each $v \in \mathcal{N}(u)$, there exists a hyperedge $e \in E$ where $u \in e$ and $v \in e$. Evaluation is performed by set-based comparison, allowing any ordering between the LVLMs' response and the pre-computed ground truth set.

Level-2 tasks integrate heuristic computations, such as vertex degrees and hyperedge orders, into Level-1 tasks. This combination increases task complexity by simultaneously requiring the accomplishment of compound atomic capabilities.

- **Degree-specified Vertex Counting (DVC).** This task requires counting vertices that meet a specific degree constraint. Given a hypergraph $\mathcal{G} = \{V, E\}$ and a target degree d, the goal is to return the number of vertices $v \in V$ such that $\deg(v) = |\{e \in E \mid v \in e\}| = d$. The target degree is randomly selected from the hypergraph's existing degree distribution, ensuring task validity. Evaluation is based on exact matches with pre-computed ground truth.

- **Order-specified Edge Counting (OEC).** This task evaluates the model's understanding of hyperedge order (cardinality). Given a hypergraph $\mathcal{G} = \{V, E\}$ and a target order k, the goal is to count hyperedges $e \in E$ where $|e| = k$. The hyperedge order represents the number of vertices it contains, a unique characteristic of hypergraphs. The target order is randomly sampled from the existing order distribution in the hypergraph. Evaluation is

Table 5: Dataset Statistics for Zero-shot Hypergraph Node Classification.

	DBLP-CA (co-authorship)	**Pubmed-CC** (co-citation)	**Cora-CA** (co-authorship)	**Cora-CC** (co-citation)	**Citeseer-CC** (co-citation)
# nodes(vertices), $\|V\|$	43413	19717	2708	2708	3312
# hyperedges, $\|E\|$	22535	7963	1072	1579	1079
avg.hyperedge order	4.7 ± 6.1	4.3 ± 5.7	4.2 ± 4.1	3.0 ± 1.1	3.2 ± 2.0
# node classes	6	3	7	7	6

based on exact matches with pre-computed ground truth.

- **Order-filtered Neighbors (ONe).** This task extends basic neighbor identification by introducing order-based filtering. Given a hypergraph $\mathcal{G} = \{V, E\}$, a vertex u, and a minimum order threshold k, the goal is to identify neighbors of u connected only through hyperedges with order $\geq k$. Formally, return $\{v \in V \mid \exists e \in E : u \in e \wedge v \in e \wedge |e| \geq k\}$. Evaluation is performed by set-based comparison, allowing any ordering between the LVLMs' response and the pre-computed ground truth set.

C.2 Reasoning Tasks

Reasoning tasks represent an advanced stage beyond understanding tasks, requiring coherent, multi-step reasoning to analyze and manipulate the information acquired from structural understanding.

Level-3 tasks involve reasoning on problems with known polynomial-time or quasi-polynomial-time algorithms. Due to the necessity of following strict algorithmic steps, where each step examines individual or compound comprehension abilities and requires organization according to the algorithm's sequence, these tasks present a greater challenge.

- **Order-weighted Shortest Path (OSP).** This task evaluates the model's ability to compute shortest paths in weighted hypergraphs. Given a hypergraph $\mathcal{G} = \{V, E\}$ and vertices $s, t \in V$, where each hyperedge e has weight $w(e) = |e|$ (its order), the goal is to find the minimum total weight W of a path from s to t. A path is a sequence of hyperedges where consecutive hyperedges share at least one vertex. Implementation uses a Dijkstra-like algorithm adapted for hypergraphs (Dijkstra, 1959). Evaluation is based on exact matches with pre-computed ground truth.

- **Order-weighted Maximum Flow (OMF).** This task assesses the model's capability to estimate maximum flow in hypergraph networks. Given a hypergraph $\mathcal{G} = \{V, E\}$ with source s and sink t, where each hyperedge e has capacity $c(e) = |e|$, the goal is to compute the maximum flow from s to t. The hypergraph is converted to a flow network by constructing its bipartite incidence graph (Berge, 1985): we introduce a dedicated intermediate node for each hyperedge, and connect each intermediate node to all vertices contained in the corresponding hyperedge. To adapt to maximum flow computation (Ahuja et al., 1994), edges between intermediate nodes and their associated vertices are assigned capacities equal to the hyperedge's capacity $c(e) = |e|$. The Edmonds-Karp algorithm (Edmonds and Karp, 1972) is then applied to this transformed flow network. Evaluation is based on exact matches with pre-computed ground truth.

- **Isomorphism Recognition (ISM).** This task evaluates the model's ability to recognize structural equivalence between hypergraphs. Given two hypergraphs $G = \{V, E\}$ and $H = \{V', E'\}$, the goal is to determine whether a bijective mapping $f : V \to V'$ exists such that $(v_1, \ldots, v_k) \in E \iff (f(v_1), \ldots, f(v_k)) \in E'$ for all hyperedges (Babai, 2016). The benchmark generates both positive (isomorphic via vertex relabeling) and negative (structurally different) examples with equal probability. Evaluation is based on binary classification accuracy.

Level-4 Tasks require reasoning on NP-hard (NP-complete) (Garey et al., 1990) problems, which are considered classic and extremely challenging. These tasks lack definitive polynomial-time algorithms as guides, often necessitating demanding models that proactively search and independently plan for feasible solutions.

- **Hypergraph 3-Coloring (3-CL).** This task assesses the model's planning capability on the

NP-complete hypergraph coloring problem (Voloshin, 2002), a generalization of the classic graph 3-coloring problem to high-order hypergraph structures. Given a hypergraph $\mathcal{G} = \{V, E\}$, determine if there exists a coloring $c : V \to \{c_0, c_1, c_2\}$ such that every hyperedge $e \in E$ satisfies $|\{c(v) \mid v \in e\}| \geq 2$ (contains at least 2 different colors). Each case includes at least one feasible coloring strategy. Solutions are verified using automated scripts.

- **Strict Hypercycle (SHC).** This task evaluates cycle detection and validation in hypergraphs with strict intersection constraints. A strict hypercycle is a sequence of hyperedges $e_1, e_2, \ldots, e_k$ ($k \geq 2$) where $|e_i \cap e_{i+1}| = 1$ for $i = 1, \ldots, k-1$ and $|e_k \cap e_1| = 1$ (consecutive hyperedges share exactly one vertex, forming a closed loop) (Bretto, 2013; Dacar, 1998). Given $\mathcal{G} = \{V, E\}$, determine if such a cycle exists and provide it. Solutions are verified using automated scripts.

- **Hypergraph Hamiltonian Path (HHM).** This task tests the model's ability to solve the NP-complete Hamiltonian path problem on hypergraphs (Bermond, 1978), an extension of the classic graph Hamiltonian path problem to hypergraphs. Given $\mathcal{G} = \{V, E\}$ and vertices $s, t \in V$, determine if there exists a sequence of hyperedges $e_1, \ldots, e_m$ that forms a path from s to t visiting every vertex in V exactly once. The path is represented as edge indices rather than vertex sequences. Solutions are verified using automated scripts.

D Visual Representation Generation

Visual hypergraph representations in HyperGVL are generated using custom visualization algorithms implemented with NetworkX (Hagberg et al., 2008) and Matplotlib (Tosi, 2009). Unlike traditional graph visualization methods, hypergraph visualization must explicitly represent hyperedges that connect arbitrary numbers of vertices, requiring specialized encoding strategies. We implement five distinct visual representation families, each designed to highlight different structural aspects of hypergraphs. The examples of each type have been provided in Fig. 2.

D.1 Implementation Details

All visualizations transform hypergraph structures into images with resolution 1400×1100 pixels at 150 DPI. The implementation uses the following core transformations:

Bipartite Transformation. For V-E pairwise representations (Bi-Inc, Sh-Inc, St-Inc), hypergraphs are first converted to bipartite graphs $G_{\text{bi}} = (V \cup E, E_{\text{bi}})$, where vertices and hyperedges form two disjoint node sets, and edges E_{bi} represent membership relations. Formally, for each hyperedge $e_i \in E$ and vertex $v_j \in e_i$, we create an edge $(v_j, e_i) \in E_{\text{bi}}$. This bipartite structure is then visualized using different layout algorithms:

- **Bipartite Incidence (Bi-Inc)**: Employs a horizontal two-layer layout where vertices are positioned in the top row and hyperedge nodes in the bottom row, with straight lines depicting membership. Vertex nodes are rendered as black circles (○, radius=20pt) with white labels, while hyperedge nodes are colored squares (□, size=18pt) with distinct colors assigned to each hyperedge.

- **Shell Incidence (Sh-Inc)**: Uses NetworkX's `shell_layout` algorithm to arrange vertices in an inner concentric circle and hyperedge nodes in an outer circle. This radial arrangement facilitates the identification of vertex-hyperedge connectivity patterns through visual clustering.

- **Star Incidence (St-Inc)**: Similar to Sh-Inc but with vertices positioned on an outer circle and hyperedge nodes placed at the center, with radial connections emanating from hyperedge centers to member vertices. This star-like pattern emphasizes the grouping structure of each hyperedge.

Clique Expansion. For Cli-Exp representation, each hyperedge $e = \{v_1, v_2, \ldots, v_k\}$ is decomposed into a complete subgraph over its vertices, generating $\binom{k}{2}$ pairwise edges. Edge labels annotate each pairwise connection with the hyperedge IDs it belongs to (e.g., "e0,e2" indicates the edge participates in both hyperedge 0 and 2). Layout is computed using NetworkX's `spring_layout` with parameters $k = 2.0$, iterations=100, scale=3.0 to minimize edge crossings while maintaining readability.

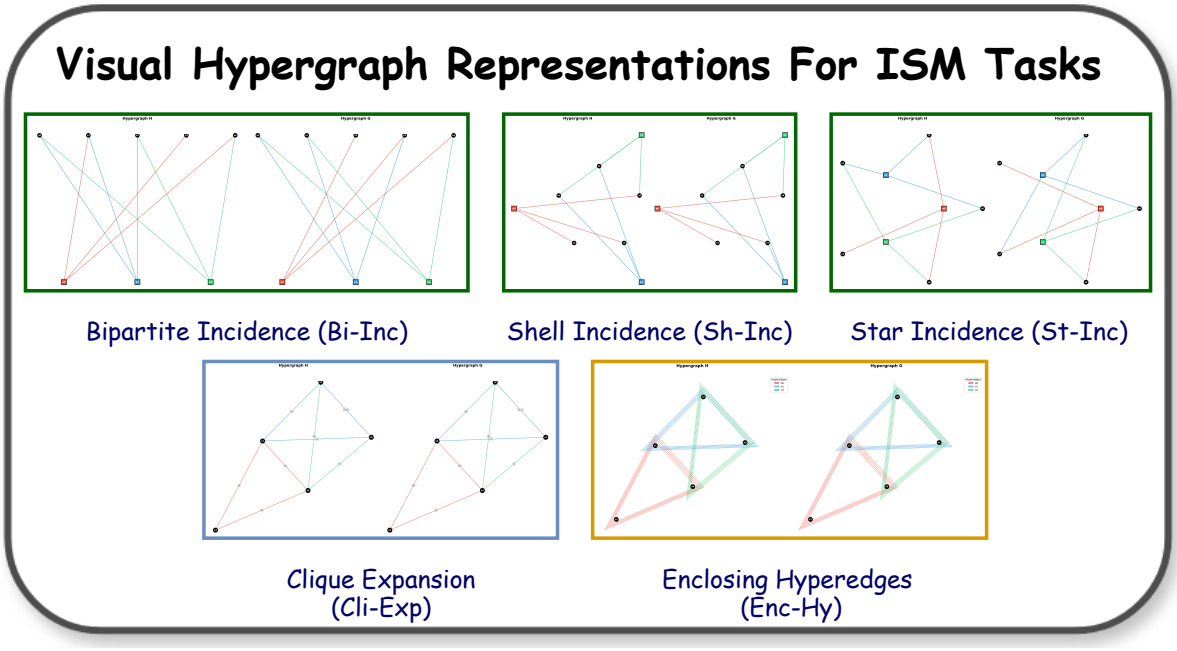

Figure 7: The visual hypergraph representations for the ISM task. The similar layouts between the two hypergraph visualizations provide straightforward cues for distinguishing isomorphism.

Enclosing Regions. For Enc-Hy representation, we implement a labeled convex hull visualization using NetworkX's `kamada_kawai_layout` for vertex positioning, which minimizes graph energy to reduce overlaps. For each hyperedge e_i with $|e_i| \geq 3$, we compute the convex hull of its member vertices using SciPy's `ConvexHull` (Virtanen et al., 2020), rendering it as a colored polygonal region with 15% opacity. Crucially, a white rectangular label box displaying the hyperedge ID (e.g., "e2") is placed at the centroid of the region to provide explicit identification, addressing the perceptual ambiguity of color-only encoding.

D.2 Design Rationale

The choice of these five representations is motivated by three design principles validated through preliminary LVLM experiments: (1) *Explicitness*: Hyperedges must have visual identifiers beyond color coding, as LVLMs struggle with pure color-based semantic extraction. All representations except Enc-Hy use explicit hyperedge nodes or edge labels. (2) *Structural diversity*: Different representations emphasize complementary structural properties: Bipartite views highlight membership relations, clique expansion reveals pairwise connectivity, and convex hulls show geometric groupings. (3) *Scalability*: Layouts are optimized for small-to-medium hypergraphs (5–20 vertices) to maintain visual clarity and avoid information overload.

D.3 Visual Representations for ISM task

Isomorphism Recognition (ISM) is special among the tasks, as it contains two hypergraphs in one query. For this task, we observe that isomorphic hypergraphs often exhibit similar visual patterns under deterministic layouts (with fixed random seeds), as layout algorithms preserve structural symmetries. This phenomenon contributes to the higher performance on ISM than our expectation, as illustrated in Tab.3. Figure 7 illustrates various visual hypergraph representations used in ISM tasks. In these representations, most layouts can directly reveal isomorphism relationships through their striking similarities.

E Prompt Examples

In this section, we exemplified the prompts used in `HyperGVL` evaluation.

E.1 Textual Representation Prompts

We provide representative examples of textual hypergraph representations used in `HyperGVL`. The following prompts demonstrate different encoding approaches for a simple hypergraph with 5 vertices (v0–v4) and 3 hyperedges (e0–e2), where e0={v0, v1, v2}, e1={v1, v2, v3}, and e2={v2, v3, v4}.

LO-Inc

Prompt: *G describes a hypergraph among vertices v0, v1, v2, v3, and v4 and hyperedges e0, e1, and e2.*
In this hypergraph:
Vertex v0 is connected to vertices v1, v2.
Vertex v1 is connected to vertices v0, v2, v3.
Vertex v2 is connected to vertices v0, v1, v3, v4.
Vertex v3 is connected to vertices v1, v2, v4.
Vertex v4 is connected to vertices v2, v3.

HO-Neigh

Prompt: *G describes a hypergraph among vertices v0, v1, v2, v3, and v4 and hyperedges e0, e1, and e2.*
In this hypergraph:
Vertex v0 is connected to hyperedge e0.
Vertex v1 is connected to hyperedges e0, e1.
Vertex v2 is connected to hyperedges e0, e1, e2.
Vertex v3 is connected to hyperedges e1, e2.
Vertex v4 is connected to hyperedge e2.
Hyperedge e0 is connected to vertices v0, v1, v2.
Hyperedge e1 is connected to vertices v1, v2, v3.
Hyperedge e2 is connected to vertices v2, v3, v4.

N-Pair

Prompt: *In an undirected hypergraph, (i,j) means that vertex i and vertex j are connected with an undirected hyperedge. G describes a hypergraph among vertices v0, v1, v2, v3, and v4 and hyperedges e0, e1, and e2.*
The connection relation between vertices in G are: (v0, v1) (v0, v2) (v1, v2) (v1, v3) (v2, v3) (v2, v4) (v3, v4).

Inc-Mat

Prompt: *G describes a hypergraph among vertices v0, v1, v2, v3, and v4 and hyperedges e0, e1, and e2.*
The incidence matrix of the hypergraph is
[[1,0,0,],
[1,1,0,],
[1,1,1,],
[0,1,1,],
[0,0,1,]]

Adj-Mat

Prompt: *G describes a hypergraph among vertices v0, v1, v2, v3, and v4 and among hyperedges e0, e1, and e2.*
The adjacency matrix between the vertices of the hypergraph is
[[0,1,1,0,0,]
, [1,0,1,1,0,]
, [1,1,0,1,1,]
, [0,1,1,0,1,]
, [0,0,1,1,0,]]

N-Set

Prompt: *In an undirected hypergraph, (i, j, k) means that vertex i, vertex j, and vertex k are connected with an undirected hyperedge. G describes a hypergraph among vertices v0, v1, v2, v3, and v4, and among hyperedges e0, e1, and e2.*
The hyperedges in G are: (v0, v1, v2), (v1, v2, v3), (v2, v3, v4).

HO-Inc

Prompt: *G describes a hypergraph among vertices v0, v1, v2, v3, and v4 and among hyperedges e0, e1, and e2.*
In this hypergraph:
Vertex v0 is connected to vertices v1, v2 with hyperedge e0.
Vertex v1 is connected to vertices v0, v2 with hyperedge e0, to vertices v2, v3 with hyperedge e1.
Vertex v2 is connected to vertices v0, v1 with hyperedge e0, to vertices v1, v3 with hyperedge e1, to vertices v3, v4 with hyperedge e2.
Vertex v3 is connected to vertices v1, v2 with hyperedge e1, to vertices v2, v4 with hyperedge e2.
Vertex v4 is connected to vertices v2, v3 with hyperedge e2.

E.2 Examples of Questions for Different Tasks

We provide representative question examples for each of the 12 tasks in `HyperGVL`, organized by difficulty levels. These examples are taken directly from the actual benchmark dataset.

E.2.1 Level-1 Tasks

Vertex Counting (VC)

Q: How many vertices are in the hypergraph $\mathcal{G}$? List the answer after "Ans:".

Hyperedge Counting (HEC)

Q: How many hyperedges are in the hypergraph $\mathcal{G}$? List the answer after "Ans:".

Neighbor (Ne)

Q: What are the direct neighbors of vertex v_4 in hypergraph $\mathcal{G}$? (Neighbors = vertices sharing at least one hyperedge with v_4). List the answer after "Ans:" in the format `{v1,v2,...}` or "No neighbors".

E.2.2 Level-2 Tasks

Degree-specified Vertex Counting (DVC)

Q: How many vertices have degree 3 in hypergraph $\mathcal{G}$? (Degree = number of hyperedges the vertex belongs to). List the answer after "Ans:".

Order-specified HyperEdge Counting (OEC)

Q: How many hyperedges have order 4 in hypergraph $\mathcal{G}$? (Order = number of vertices in the hyperedge). List the answer after "Ans:".

Order-filtered Neighbors (ONe)

Q: What are the neighbors of vertex v_5 when only considering hyperedges with order $Eq2$ in hypergraph $\mathcal{G}$? List the answer after "Ans:" in the format `{v1,v2,...}` or "No n-neighbors".

E.2.3 Level-3 Tasks

Order-weighted Shortest Path (OSP)

Q: What is the shortest path length from vertex v_4 to vertex v_8 in hypergraph $\mathcal{G}$, where each hyperedge's weight equals its order (number of vertices)? If no path exists, answer "No path". List the answer after "Ans:".

Order-weighted Maximum Flow (OMF)

Q: What is the estimated maximum flow from vertex v_5 to vertex v_0 in hypergraph $\mathcal{G}$, where each hyperedge's capacity equals its order? If no flow exists, answer "0". List the answer after "Ans:".

Isomorphism Recognition (ISM)

There are two hypergraphs: H and $\mathcal{G}$.
The description of H is:
The description of $\mathcal{G}$ is:
Q: Are these two hypergraphs isomorphic? (Two hypergraphs are isomorphic if there exists a vertex relabeling that transforms one into the other). List the answer after "Ans:" in the format [Yes/No].

E.2.4 Level-4 Tasks

Hypergraph 3-Coloring (3-CL)

Q: Please provide a 3-coloring strategy such that each hyperedge contains nodes with at least 2 different colors (assign each vertex a color from {c0, c1, c2}). List the answer after "Ans:" as "Coloring:[v0:c0,v1:c1,...]".

Strict Hypercycle (SHC)

Q: Please identify a strict hypercycle in the hypergraph $\mathcal{G}$ (A strict hypercycle is a sequence of hyperedges $e_1, e_2, ..., e_k$ where adjacent hyperedges share exactly one vertex, i.e., $|e_i \cap e_{i+1}| = 1$, and $|e_k \cap e_1| = 1$, forming a closed loop). List the hypercycle after "Ans:" as "Cycle:[e0,e1,...]".

Hypergraph Hamiltonian Path (HHM)

Q: Please provide a valid Hamiltonian path from v1 to v0. (Hamiltonian path = path visiting all vertices exactly once). List the answer after "Ans:" as "Path:[e0,e1,...]".

Despite there is format guidance in the prompt, we perform manual review to accept factually correct answers with minor formatting deviations, ensuring evaluation fairness.

F HyperGVL Evaluation Details

For all evaluated LVLMs and LLMs, we set the temperature parameter to 0.8 to introduce a controlled level of randomness during generation, which allows us to assess both the stability and consistency of model outputs across different runs. We adopt nucleus sampling with a top-p value of 0.95 and a maximum generation length sufficient to cover all task-specific output requirements. All experiments were conducted on a compute cluster equipped with 64 NVIDIA RTX 5090 GPUs. Model inference was executed using mixed-precision computation to balance efficiency and numerical stability. The evaluation pipeline was parallelized across GPUs to ensure consistent runtime conditions for all models.

G Hypergraph Node Classification Details

G.1 Dataset Statistics

In Sec. 4, we demonstrate the out-of-domain generalizability of `WiseHyGR` on real-world hypergraph node classification under the zero-shot setting. The tested hypergraphs come from multiple sources, including co-authorship hypergraphs and co-citation hypergraphs. The dataset statistics are provided in Tab.5.

- Co-authorship hypergraphs: Cora-CA [1] and DBLP-CA[2] (Yadati et al., 2019), where vertices represent articles and hyperedges denote the co-author relationship of articles that share a same author.
- Co-citation hypergraphs: Cora, Citeseer, and Pubmed[3] (Yang et al., 2016b), where vertices are articles and hyperedges connect all articles that are cited by the same article.

G.2 Experimental Setups

From each dataset, we sample 1K hypergraph nodes for classification. To ensure these nodes (i.e., vertices) form a connected hypergraph, we initiate a random walk from a target node, collecting up to n nodes. We set $n = 40$ in our experiments to provide sufficient structural information to the LVLMs while minimizing noise and information congestion from excessively long-range dependencies and overly broad scopes. The performances shown in Fig.6 are averaged over three trials to ensure robust comparisons. Zero-shot hypergraph node classification is conducted using prompts exemplified below, with categorization descriptions generated by GPT-4o (OpenAI, 2024). The node attributes are the manually retrieved article abstracts.

[1] https://people.cs.umass.edu/mccallum/data.html
[2] https://aminer.org/lab-datasets/citation/DBLP-citation-Jan8.tar.bz
[3] https://linqs.soe.ucsc.edu/data

Hypergraph Node Classification Prompts

Prompt: *Classify node v0 in the hypergraph.*
Hypergraph Structure: {Textual Hypergraph Representation}
Known labels in the subgraph:

- *v1: Probabilistic Methods*
- *v2: Reinforcement Learning*
- *...*

Target Node Content: {Paper Abstract}
Categories:

- *Genetic Algorithms: ...*
- *Probabilistic Methods: ...*
- *...*

Based on the hypergraph structure, known labels of other nodes, and the target node's abstract content, which category does node v0 most likely belong to?
Answer with the category name only.