Colorful Talks with Graphs: Human-Interpretable Graph Encodings for Large Language Models

Angelo Zangari Peyman Baghershahi Sourav Medya
University of Illinois Chicago, Chicago, IL, USA
{azang,pbaghe2,medya}@uic.edu

Abstract

Graph problems are fundamentally challenging for large language models (LLMs). While LLMs excel at processing unstructured text, graph tasks require reasoning over explicit structure, permutation invariance, and computationally complex relationships, creating a mismatch with the representations of text-based models. Our work investigates how LLMs can be effectively applied to graph problems despite these barriers. We introduce a human-interpretable structural encoding strategy for graph-to-text translation that injects graph structure directly into natural language prompts. Our method involves computing a variant of Weisfeiler–Lehman (WL) similarity classes and maps them to human-like color tokens rather than numeric labels. The key insight is that semantically meaningful and human-interpretable cues may be more effectively processed by LLMs than opaque symbolic encoding. Experimental results across multiple graph algorithms and predictive tasks show significant improvements from our method on both synthetic and real-world datasets. By capturing both local and global-range dependencies, our method enhances LLM performance, especially on graph tasks that require reasoning over global graph structure.

1 Introduction

Large language models (LLMs) have advanced rapidly over the last few years, demonstrating substantial improvements in reasoning, tool use, and multimodal capabilities across successive generations. These advances have enabled LLMs to be deployed in domains traditionally dominated by specialized models, including program synthesis, scientific reasoning, and structured problem solving (DeepSeek-AI, 2025). Recently, the emergence of agentic LLM systems, capable of interacting with tools, APIs, and external environments via standardized protocols, has further blurred the boundary between general-purpose language models and domain-specific reasoning systems. Together, these developments raise a natural question: *to what extent can LLMs serve as general solvers for structured reasoning tasks that have historically required specialized architectures?*

Graph problems provide a particularly revealing testbed for this question. Unlike text or images, graphs are combinatorial objects with no canonical ordering, and many graph tasks require permutation invariance, symmetry awareness, and reasoning over multi-hop relational structure (Wang et al., 2024). Moreover, a large class of canonical graph problems (e.g., subgraph isomorphism) are NP-hard or NP-complete (Manchanda et al., 2020; Ranjan et al., 2022), placing substantial demands on systematic reasoning and abstraction. As a result, graph tasks expose a fundamental mismatch between graph structure and the sequential, text-based representations that LLMs are trained to process (Mao et al., 2024).

Recent work has explored whether LLMs can nevertheless be applied to graph problems by translating graphs into text. Early studies show that LLMs can answer simple queries such as node degree, edge existence, and local neighborhood questions when graphs are serialized into natural language prompts (Guo et al., 2023; Wang et al., 2023). Subsequent work expanded this line of inquiry to include path finding, cycle detection, and limited forms of combinatorial reasoning (Fatemi et al., 2024; Sanford et al., 2024). In parallel, several approaches have proposed augmenting graph-to-text translations with additional structure, such as similarity cues, embeddings, or symbolic annotations, to better guide LLM reasoning (Liu et al., 2024a; Wang et al., 2024; Sun et al., 2023).

Despite this progress, existing works remain limited. Most studies focus on a simple set of tasks, typically emphasizing local or classification-style problems such as node classification or link pre-

Findings of the Association for Computational Linguistics: ACL 2026, pages 41194–41215
July 2-7, 2026

diction. Evaluations are often restricted to small graphs, leaving open questions about scalability and task generality. Furthermore, as closed-source LLMs cannot be retrained on graph data, most methods rely on prompt engineering or representational choices where theoretical grounding is unclear. In particular, many graph-to-text encodings introduce symbols or numeric identifiers that are structurally meaningful but linguistically opaque.

Our Contributions. Our work is motivated by the hypothesis that human-interpretable structural representations can help bridge the gap between graph structure and LLM reasoning. LLMs are pretrained and aligned on vast corpora of human language and are therefore biased toward representations that resemble human explanatory conventions. We posit that graph structure, when expressed through interpretable and semantically grounded cues, may be more effectively exploited by LLMs than arbitrary numeric or symbolic encodings. Our contributions span representation design, theoretical motivation, and rigorous experiments with various tasks and data. They are as follows.
(i) Human-interpretable structural encoding: We introduce a graph-to-text encoding strategy that injects explicit structural information into LLM prompts using human-interpretable tokens derived from Weisfeiler-Leman (WL) refinement. This proposed representation preserves permutation invariance and structural similarity while aligning graph structure with linguistic abstractions familiar to pretrained LLMs.
(ii) Structure-preserving representation: We provide a principled analysis connecting ordered WL refinement to distance-weighted notions of node connectivity. Under mild assumptions, we show that the induced WL ordering is consistent with a broad class of centrality-like measures.
(iii) Experiments: We conduct a comprehensive empirical study across diverse graph families and task types to assess the effectiveness of human-interpretable structural encoding.

2 Related Work

We begin with methods based on the idea that graphs can be represented as text and processed by LLMs. Initial studies (Fatemi et al., 2024; Guo et al., 2023; Wang et al., 2023) explored basic tasks such as node degree, edge existence, and simple structural queries. Some have broadened the scope to include limited forms of combinatorial reasoning. LLMs can often answer small or local queries reliably, but performance tends to degrade as graphs become larger or tasks require multi-step reasoning (Fatemi et al., 2024).

Different Ways of Applying LLMs on Graphs. Another set of approaches provide LLMs an additional structure or guidance, such as augmenting graph descriptions with similarity cues, embeddings, or human-interpretable features. Prior work shows that LLMs can play several distinct roles within graph-related pipelines. Existing methods fall broadly into three categories: *(i) Solvers*: LLMs directly compute answers to tasks such as degree queries, path finding, triangle counting, etc., using only text prompts (Wang et al., 2023; Guo et al., 2023; Fatemi et al., 2024; Zhang et al., 2024; Sun et al., 2023). *(ii) Aligners*: LLMs guide or interface with graph-learning models, enforce constraints, or translate natural-language instructions into graph-processing steps (Jin et al., 2024b; Zhu et al., 2024; Tan et al., 2024; Li et al., 2025). *(iii) Encoders*: LLMs convert graphs into detailed textual representations, embeddings, or rationales that downstream modules or humans can interpret (Zhao et al., 2023; Fatemi et al., 2024; Liu et al., 2024a; Zhu et al., 2025). Please refer to these recent surveys (Jin et al., 2024a; Ren et al., 2024).

Recent representative systems instantiate these directions in different ways. GraphText (Zhao et al., 2023) performs graph reasoning in text space by constructing graph-aware textual contexts for LLM inference. OFA (Liu et al., 2024a) instead trains a unified graph model across classification tasks, and LLaGA (Chen et al., 2024) combines language models with graph-side representations through a graph assistant architecture. These methods show that graph structure can benefit from textual, learned, or hybrid graph-language representations. Our work is complementary: rather than training a graph-language model or injecting continuous embeddings, we study a training-free graph-to-text encoding based on ordered WL descriptors and natural-language color tokens. This lets us isolate how discrete, human-interpretable structural cues affect LLM graph reasoning under controlled prompts.

Methods as Solvers using LLMs. A growing line of work examines LLMs used directly as solvers for graph queries. These methods vary along three orthogonal dimensions such as graph types, graph tasks, and prompt formats. Many solver-oriented studies evaluate LLMs on real-world text-attributed

graphs such as Cora, Citeseer, PubMed, and the OGB datasets (Ye et al., 2024; Qin et al., 2023; Liu et al., 2024a; He et al., 2024). In parallel, several benchmarks for explicit graph-in-text reasoning rely on synthetic graphs with controlled topology, such as NLGraph or G-Recall (Wang et al., 2023, 2024; Sanford et al., 2024). These datasets offer meaningful semantic labels, but their size and irregular structure introduce challenges for text-based models. Beyond these, a substantial number of works also experiment with synthetic graphs because their structure is easier to control and analyze (Wang et al., 2023, 2024; Sanford et al., 2024). When graphs are fully serialized into a single prompt, evaluations almost always restrict to small graphs since LLM performance declines quickly as the number of nodes or edges grows.

Additional details on related work have been provided in Appendix A.

3 The Problem Setup

Unlike GNNs, LLMs are sequence models: they consume and produce text, and have no native interface for graphs. Using an LLM to solve a graph problem therefore requires a *graph-to-text translation* step. The graph, the task description, and any auxiliary information must be encoded as a textual sequence. The central objective of our work is to design a principled graph-to-text representation method that enables LLMs to reason about graphs.

Formulation. Solving a graph problem with an LLM requires transforming a structured, non-sequential object into a textual sequence. We formalize the elements involved in this transformation and highlights the key factors that influence overall effectiveness. Let $G = (V, E)$ be a graph with node set V and edge set $E \subseteq V \times V$, and let $q \in Q$ denote a graph task (e.g., cycle detection, reachability between two nodes). LLMs operate over a set of text sequences $W \subset \Sigma$ where Σ is a finite alphabet, and can be viewed as a function $f_\theta : W \to W'$ where $W' \subset \Sigma$, parameterized by θ.

We translate the graph and task into a textual prompt via a translation function $\tau : X \to W$, where X encompasses all information supplied to the model: graph structure, auxiliary structural information, task description, and prompt scaffolding. This transformation may not be unique. Subsequently, for an input query q (a graph task) the model prediction becomes:

$$y = f_\theta\big(\tau(G, q, x_{\text{struct}})\big). \quad (1)$$

Based on this formulation, the overall effectiveness of a method depends on two major components: **(1) Information content** x_{struct}: the structural information from the graph. This may range from an edge list to richer structural summaries. **(2) Translation strategy** τ: how the information is expressed in text. For example, one can include ordering of edges, natural-language descriptions, few-shot examples, explicit markers for labels or colors, and the level of verbosity. These components interact strongly. For example, adding more structural information improves expressiveness but may increase prompt length and complexity; likewise, stronger models may benefit more from richer encoding.

4 Our Proposed Method

We introduce node-level structural identifiers based on Weisfeiler-Leman (WL) (Shervashidze et al., 2011) refinement and map into a human-interpretable form using colors. Figure 1 shows the overview of our method.

4.1 Node-level Structural Identifiers (x_{struct})

Many graph problems require reasoning beyond local neighborhood. For example, cycle detection, shortest paths, or flow-related tasks depend on multi-hop neighborhoods, symmetries, and recurring substructures. Therefore, identifiers that are created from global structural information enable better graph comprehension, and thus, becomes beneficial for all downstream tasks. Graph learning models (e.g., GNNs) aggregate a node's neighborhood information through message passing to generate structure-aware embeddings, such that similar nodes attain similar embeddings. However, such latent embeddings have no meaning to a language model and appear as arbitrary numerical values.

To inject structural information, some prior approaches directly use the continuous GNN embeddings into the LLM prompts for graph problems (Baghershahi et al., 2025; Chen et al., 2024). Other methods replace node identifiers with alternative vocabularies or descriptive placeholders (Fatemi et al., 2024). These strategies suffer from a common limitation. The injected symbols do not carry interpretable information for the LLM. Floating-point embeddings are tied to a specific model and training process, while renamed node identifiers remain arbitrary tokens whose relationships are not grounded in the model's linguistic prior knowledge.

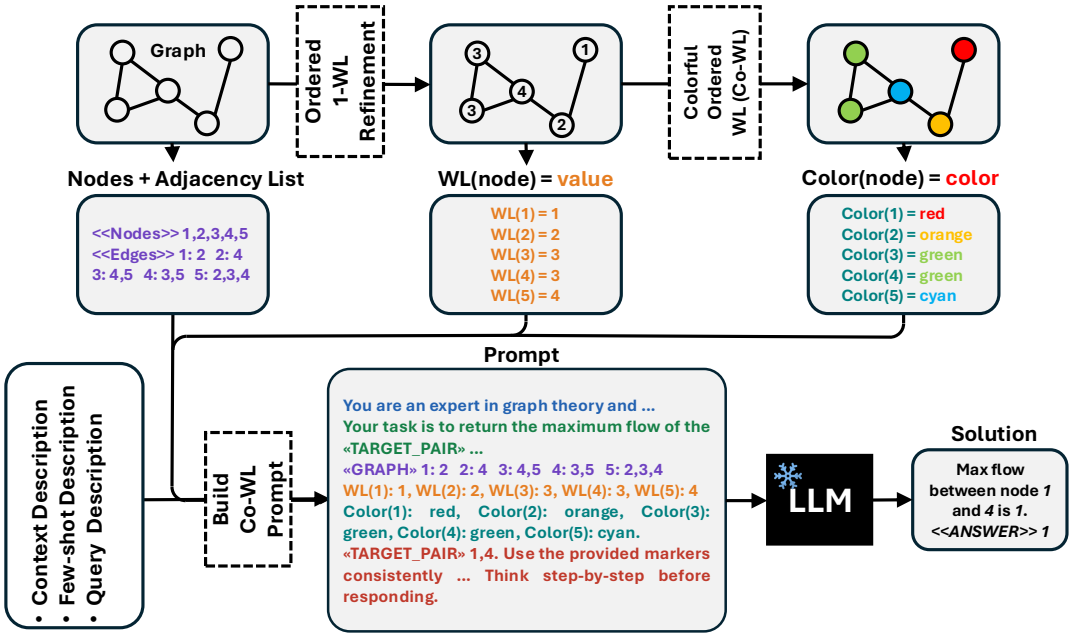

Figure 1: Overview of the proposed Colorful Ordered Weisfeiler-Leman (CL-OWL) pipeline. A graph is first represented as nodes with adjacency lists, then processed using ordered 1-WL refinement to compute node-level structural labels. These labels are mapped to human-tangible color tokens and embedded into a natural-language prompt. The resulting prompt enables an LLM to perform graph reasoning tasks (e.g., maximum-flow).

To address these limitations, we construct node-level structural descriptors that: 1) are *structure-aware* to allow capturing global topological information, 2) preserve *structural similarity* to assign identical descriptors to nodes that are structurally similar, 3) acknowledge an *ordered interpretation* to enable comparison between nodes based on the extent of their connectivity. These descriptors can be reliably used by LLMs.

Constructing a structural descriptor. We adapt or refine the Weisfeiler-Leman (WL) (Shervashidze et al., 2011) coloring algorithm which is a heuristic for graph isomorphism test. The algorithm iteratively refines node labels by aggregating information from neighborhoods. At each iteration, a node updates its label based on its current label and the multiset of labels of its neighbors. After k iterations, the resulting label summarizes the node's k-hop neighborhood structure.

WL labels in their standard form are poorly suited for direct use in LLMs. The labels are arbitrary integers produced by hashing operations; their numeric values have no semantic meaning, no inherent ordering, and no interpretable relation to each other. This misalignment is particularly problematic for LLMs that are pre-trained on natural language patterns and struggle with reasoning over such arbitrary integers or opaque identifiers.

To address the above limitation, we modify the WL method to induce an ordering over node labels that are informative for LLMs. The key idea is to constrain the aggregation step so that label updates reflect not only structural similarity but also a notion of cumulative structural connectivity.

Ordered 1-WL refinement

We compute node-level structural descriptors using a refinement procedure of the canonical 1-dimensional Weisfeiler-Leman (1-WL) (Shervashidze et al., 2011). Let us initialize node labels $\ell_v^{(0)} \in \mathbb{N}$ (e.g., $\ell_v^{(0)} = 1$ for all $v \in V$). Let $\mathcal{N}(v)$ be the self-exclusive neighborhood function s.t. $\mathcal{N}(v) = \{u : (v,u) \in E\}$. For iterations $t = 0, 1, \ldots, T-1$, update each node label by aggregating the multiset of neighbor labels and applying a deterministic, order-preserving canonicalization:

$$\ell_v^{(t+1)} = \mathrm{ID}\Big(\ell_v^{(t)},\ \mathrm{Sort}\Big(\{\!\{\, \ell_u^{(t)} : u \in \mathcal{N}(v) \,\}\!\}\Big)\Big)$$

Here $\{\!\{\cdot\}\!\}$ denotes a multiset and $\mathrm{Sort}(\cdot)$ converts it to a tuple that is sorted lexicographically (thus sorted in ascending numerical order and preserving multiplicities). The function $\mathrm{ID}(\cdot)$ assigns a unique integer identifier to each distinct pair $\big(\ell_v^{(t)}, \mathrm{Sort}(\cdot)\big)$

in a globally deterministic order, by (i) collecting all node messages $m_v^{(t)} = \big(\ell_v^{(t)}, \mathrm{Sort}(\{\{\ell_u^{(t)} : u \in \mathcal{N}(v)\}\})\big)$, (ii) sorting the set of unique messages lexicographically, and (iii) mapping the i-th message in this sorted list to identifier i. These labels form our node-level structural prompt augmentation x_{struct}.

Ordered 1-WL & Centrality

We formalize the relationship between our *ordered* 1-WL refinement and standard notions of node centrality. We define centrality as follows.

Definition 1 (Distance-shell counts and truncated connectivity)**.** *For $k > 0$, define the k-th distance shell of v as $S_k(v) = \{u \in V : d(u, v) = k\}$, where $d(\cdot, \cdot)$ is shortest-path distance. For a weight sequence $\alpha_0 \geq \alpha_1 \geq \cdots \geq \alpha_T > 0$, define the truncated distance-weighted connectivity*

$$C_T(v) \;=\; \sum_{k=0}^{T} \alpha_k \, |S_k(v)|.$$

Theorem 1. *Let $\ell^{(t)}$ be the labels produced by ordered 1-WL on G.*

(1) Degree consistency. *For any nodes $v, w \in V$, $\deg(v) > \deg(w) \implies \ell_v^{(1)} > \ell_w^{(1)}$.*

(2) Shell-dominance implies label dominance. *Fix $T \geq 1$. Suppose there exist nodes $v, w \in V$ such that (i)$|S_k(v)| \geq |S_k(w)|$ for all $k = 1, \ldots, T$, and (ii) $|S_{k^\star}(v)| > |S_{k^\star}(w)|$ for some $k^\star \leq T$, and additionally the rooted T-hop neighborhoods of v and w are* tree-unfoldings *(i.e., have no collisions in the breadth-first expansion up to depth T). Then $\ell_v^{(T)} > \ell_w^{(T)}$ holds.*

(3) Relation with connectivity. *With (2), for any nonincreasing positive weights $\{\alpha_k\}_{k=0}^{T}$, $C_T(v) > C_T(w)$. Hence, ordered 1-WL induces an ordering that is consistent with a class of distance-weighted connectivity functions on locally tree-like neighborhoods.*

The proof is given in Appendix B.1. Theorem 1 provides a connection between ordered 1-WL labels and distance-weighted connectivity. This shows the benefits of using 1-WL refinement as structural identifiers.

4.2 Translating Structural Information into Text (τ) & CL-OWL Prompting

To make x_{struct} usable by an LLM, it must be translated into text via the translation function $\tau(G, q, x_{\text{struct}})$. As discussed before, arbitrary integers or symbolic codes are generally suboptimal for conveying structural meaning to LLMs. Meanwhile, LLMs are effective at leveraging representations that resemble natural explanatory language. An effective translation strategy should therefore express structural information in a form that is both faithful to the underlying graph properties and suitable to exploit the linguistic priors of LLMs.

Our objective is to map these labels into a textual representation that preserves key properties: similarity-preserving, order-preserving, and conciseness. Nodes with similar structural features should have similar representations. For example, nodes with higher structural connectivity should be distinguishable from those with lower connectivity in a consistent and interpretable manner.

Colors as an interpretable similarity space. To achieve the above, we map WL-derived labels into colors. Colors provide a natural similarity space that is widely used in human communication to group and distinguish entities. In natural language, colors carry intuitive relational meaning (e.g., "red is closer to orange than blue"), and LLMs are well-equipped to exploit such associations.

From a representational perspective, colors also admit an ordered interpretation. By restricting attention to a one-dimensional segment of the color spectrum, variations in hue can be associated with a monotonic structural scale derived from the ordered WL labels. This allows us to encode both equivalence and graded similarity in a way that remains human-interpretable. Figure 2 shows an example.

Colorful-WL Prompting (CL-OWL)

Here we formally explain how our **Colorful-ordered-WL (CL-OWL)** translator τ constructs the LLM prompts. The main input to the LLM is a graph task $T \subset U$, where $U \in V \times E \times \mathbb{R}$ is a general set of graph problems/tasks, e.g. shortest path prediction. Each task is a set of tuples of query q and answer a as $T = \{(Q, S(V_s, E_s), a) : S(V_s, E_s) \in V \times E, a \in \mathbb{R}, Q \subset V_s\}$. Note that $S(V_s, E_s)$ is a graph and not necessarily a subgraph of G, but we have $V_s \in V$ and edges $E_s \in E$. Also, Q could be a singleton set, $\{u\} \subset V_s$, (e.g. in node classification) or a subset of nodes, $\{u_i\}_{i=1}^{m} \subset V_s$, (e.g., shortest path where $m = 2$). Given a tuple $T^{(i)} = (Q^{(i)}, S(V_s^{(i)}, E_s^{(i)}), a^{(i)}) \in T$, $\forall i \in [|T|]$, our translator incorporates multiple channels of

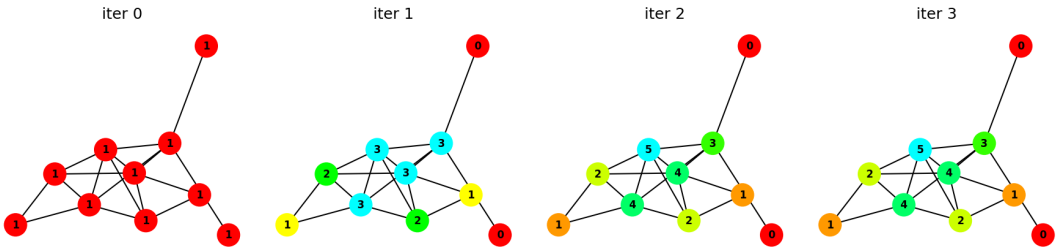

Figure 2: Iterative refinement of WL labels and their mapping to human-interpretable colors, showing how structural equivalence classes evolve as increasingly larger neighborhoods are incorporated.

information in the prompts as follows:

1. Context descriptor. A descriptor $\tau_c : W \times V \times E \to W$ uses the input subgraph $S(V_s^{(i)}, E_s^{(i)})$ along with prespecified graph notation, task definition, and LLM characterization templates to provide a rich description as the task descriptor generates a textual sequence $P_c^{(i)} = \tau_c(S(V_s^{(i)}, E_s^{(i)}))$.

2. Colorful-WL Structure descriptor. A descriptor τ_{cwl} leverages the structural information collected from $S(V_s^{(i)}, E_s^{(i)})$ into the prompt. Specifically, let $WL(.)$ be the WL mechanism which generates labels $L_u = WL(u) \in \mathbb{Z}^+$ for $\forall u \in V_s^{(i)}$, and let $\bar{L}_u \in [0, 1]$ be the normalized labels such that:

$$\bar{L}_u = \frac{L_u - \min_{v \in V} L_v}{\max_{v \in V} L_v - \min_{v \in V} L_v},$$

We use a color mapping function $\Psi : [0, 1] \to W_c$ to obtain natural language color labels $C_v = \Psi(\bar{L}_v)$, we use the, where $W_c \subset W$ is a subset of natural language color words like {red, pink, yellow, ...}. To exemplify, choosing the RGB color spectrum, we can have $\Psi = \Psi_{\text{NLC}} \circ \Psi_{\text{RGB}}$, where $\Psi_{\text{RGB}} : [0, 1] \to [0, 1]^3$ which gives RGB channels color as $\Psi_{\text{RGB}}(\bar{L}_u) \to (R_u, G_u, B_u)$, and $\Psi_{\text{NLC}} : [0, 1]^3 \to W$ which maps the RGB channels to natural language color tokens. The structure descriptor generates a textual sequence $P_{\text{cwl}}^{(i)} = \tau_{\text{cwl}}(S(V_s^{(i)}, E_s^{(i)}))$.

Notably, we express colors using natural language descriptors rather than alphanumeric encodings such as hexadecimal color codes. Unlike numeric labels or hexadecimal color codes, natural-language color tokens correspond to semantically grounded embeddings learned during pretraining, enabling the model to exploit similarity relationships already encoded in its representation space.

3. Few-shot Guidance descriptor. A few-shot builder τ_e is utilized to sample a set of queries and answers from the task such that $I^{(i)} \subset T$. Specifically, we use a wrapper $r(.)$ to put the few-shot examples in a fixed template. Next, the builder puts all the wrapped examples with some description to generate $P_e^{(i)} = \tau_e(r(I^{(i)}))$.

4. Query descriptor. The final component of the prompt is the query descriptor τ_q. It states the problem along with the wrapped query as $P_q^{(i)} = \tau_q(r(\{Q\}))$.

Our CL-OWL translator mixes all the above components and makes the suitable structure-aware prompts as $P^{(i)} = \tau(Q^{(i)}, S(V_s^{(i)}, E_s^{(i)}))$, which could be a simple concatenation as $P^{(i)} = [P_c^{(i)} \| P_{\text{cwl}}^{(i)} \| P_e^{(i)} \| P_q^{(i)}]$.

Compressed prompting for localized queries. The same translator supports a localized prompt construction. For a query node set $Q^{(i)}$ and a hop budget k_n, we define the retained node set

$$R_{k_n}(Q^{(i)}) = \{v \in V_s^{(i)} : \min_{u \in Q^{(i)}} d(v, u) \leq k_n\}.$$

When $k_n = -1$, no node filtering is applied. Otherwise, the prompt serializes only the induced subgraph on $R_{k_n}(Q^{(i)})$ and emits WL labels or color descriptors only for retained nodes. This node-level filtering is independent of the choice of textual encoding: it can be applied to the TLG-style baseline, to L-OWL, to C-OWL, or to CL-OWL. It therefore separates two effects that are otherwise confounded in long prompts: the amount of graph context given to the LLM and the type of structural descriptor used for the retained context.

Properties of the translation. This color-based translation is structure-, similarity-, and order-preserving, while remaining human-interpretable and aligned with LLM inductive biases. The computed labels and colors are the output of the translation function τ and are injected into the prompt. Examples of the prompt structure are in Appendix B.3.

Task	Time complexity	Objective
Triangle Counting	$O(m^{3/2})$	Local Pattern Counting
Cycle Check	$O(n+m)$	Global Structural Reasoning
Reachability	$O(n+m)$	Global Connectivity Reasoning
Shortest Path	$O(n+m)$	Combinatorial Reasoning
Maximum Flow	$O(n\,m^2)$	Optimization

Table 1: Task time complexity and characteristics (with $n = |V|$, $m = |E|$) for algorithmic tasks. We also use node classification as a predictive task.

5 Experimental Results

We evaluate whether ordered WL structural descriptors and their color-based translation improve LLM-based graph reasoning and prediction. Code is available at: https://github.com/angelozangari/CL-OWL

5.1 Setup

The following contains major details on experimental setups. Additional details are in Appendix D.

Graph Tasks. We evaluate our methods on a variety of local and global algorithmic tasks requiring different types of reasoning. Table 1 shows the time complexity of the optimal algorithmic solver for each task. We also evaluate **node classification** as the most common graph predictive task.

Datasets. For algorithmic tasks (Table 1), we use synthetic graphs of multiple types, namely: Erdős–Rényi, Barabási–Albert, and Path. For each type, we generate graphs with sizes from $n = 5$ to $n = 100$—beyond this, we observed that even strong LLMs (e.g., GPT-4-class) maintain zero accuracy on our hardest tasks. For node classification, we sample subgraphs from real-world datasets, specifically Cora, Citeseer, PubMed, (Yang et al., 2016) and OGBN-ArXiv (Hu et al., 2020). Additional details are in Appendix D.

Baselines. The recent work, Talk-Like-a-Graph (TLG) (Fatemi et al., 2024), introduces various types of methods for graph-to-text translation by graph traversal. Each method employs a distinct vocabulary of words for textual node encoding, along with a special edge encoding. We compare our method against two variants. 1) **TLG-A**: node indices for node encoding and tuples representation for edges, and 2) **TLG-F**: Friends series' characters for node encoding and tuples for edge representation. For node classification, we further compare against three recent graph-language baselines: GraphText (Zhao et al., 2023), LLaGA (Chen et al., 2024), and OFA (Liu et al., 2024a). We use the official GraphText inference pipeline, the official released LLaGA checkpoint, and the official OFA codebase with the paper-reported supervised training setting for node classification (100 epochs, learning rate 10^{-4}, batch size 128). *We use GPT-3.5 as the base LLM for the experiments unless specified otherwise.*

Variations of our Method. We evaluate three variants of our method against the baselines as follows: (i) **L-OWL:** Pure (Only) WL-based Labels; (ii) **C-OWL:** Pure (Only) WL-based Colors; (iii) **CL-OWL:** WL-based Labels + Colors.

5.2 Graph Reasoning Tasks

We evaluate our methods on graph algorithmic tasks. We sample 200 graphs for each task of sizes 10-30 nodes from Barabasi-Albert (BA) and Erdos-Renyi (ER) graph types following the setup in (Fatemi et al., 2024; Wang et al., 2024). To avoid non-trivial tasks, we make graphs sufficiently connected by setting an edge probability of $p = 0.2$ for ER graphs and a minimum number of four edges per node for BA graphs. These specific graph types are closer to real-world graphs compared to simpler structures, such as complete or path graphs.

Results are shown in Table 2. For challenging graph tasks, our proposed WL-enriched prompting (with/without colors) achieves significant improvements over the baselines. This empirically verifies our claim, as including the WL-labels provides information from long-range dependencies that are accessible and allow reasoning on these tasks that require global structural knowledge. The coloring variants (C-OWL and CL-OWL) show superior performance over the pure-WL methods. Regarding graph-level triangle counting, the WL-enriched prompts underperform the baselines. We conjecture that exact global triangle counting is closer to exhaustive local pattern matching over all triples, where extra descriptors can lengthen the prompt without directly identifying the counted motifs. In Section E.4, we therefore separately evaluate a localized triangle-membership task and find that WL-based cues improve local motif recognition.

5.3 Structural Information Compression

The ordered WL labels and their color-based translation expose graph-level structural information in a compact node-level form. To verify this claim we first evaluate the compressed prompting variant introduced in Section 4.2. The idea is to retain only the induced subgraph within a fixed hop budget around the query node while preserving WL-derived descriptors for the retained nodes. If

Variant	Cycle Check		Maximum Flow		Shortest Path		Triangle Counting	
	Acc. (%)	MAE ↓	Acc. (%)	MAE ↓	Acc. (%)	MAE ↓	Acc. (%)	MAE ↓
TLG-A	89.50	0.105	33.33	0.494	87.80	0.041	**16.00**	0.456
TLG-F	91.50	0.085	**36.59**	0.445	83.74	0.050	11.50	0.523
L-OWL	92.00	0.080	**36.59**	0.475	**91.87**	**0.025**	14.00	0.508
C-OWL	92.50	0.075	**36.59**	0.443	88.62	0.039	14.00	**0.454**
CL-OWL	**93.00**	**0.070**	**36.59**	**0.438**	86.99	0.042	14.50	0.469

Table 2: Results on graph algorithmic tasks averaged over BA and ER datasets using `gpt-4o` (additional results are in Appendix E). CL-OWL generally achieves superior performance over the baselines. Overall, the color-based variants (CL-OWL and C-OWL) demonstrate more improvements compared to others.

Variant	Acc. (%)	MAE ↓	RMSE ↓
TLG-A (1-hop)	30.0	0.700	0.837
TLG-A (2-hop)	40.0	0.600	0.775
TLG-A (3-hop)	45.0	0.550	0.742
C-OWL (1-hop)	**75.0**	0.211	0.459
L-OWL (1-hop)	50.0	0.500	0.707
CL-OWL (1-hop)	70.0	**0.176**	**0.420**

Table 3: Compressed prompting on Cora node classification. The hop count specifies the retained neighborhood around the target node before serialization.

Graph	Method	50	100	500	1000
BA	TLG-A	4442	15122	390949	1705906
	C-OWL	2595	6148	188525	364108
	L-OWL	**2518**	**5943**	**187435**	**362409**
	CL-OWL	3172	6957	191561	368648
ER	TLG-A	**1382**	1842	9045	29020
	C-OWL	1393	**1399**	**1454**	1530
	L-OWL	1409	1414	1458	**1521**
	CL-OWL	1813	1822	1898	2002

Table 4: Mean prompt length (characters) for compressed node-classification prompts across graph types and sizes. The TLG-A baseline serializes the full retained graph context, while WL-based variants use 1-hop node filtering around the query node.

the descriptors indeed summarize non-local structural information, then they can compensate for the loss of explicit graph context. We test this on Cora for node classification task using 50-node subgraphs. For the TLG-A baseline, we vary the retained neighborhood from 1-hop to 3-hop around the target node; for WL-based variants, we use only the 1-hop neighborhood. Table 3 shows that the baseline degrades sharply under this restriction (accuracy of 30-45%). In contrast, C-OWL reaches 75.0% accuracy with only 1-hop context, and CL-OWL reaches 70.0% while achieving the best MAE and RMSE. Thus, the WL-based variants outperform the baseline while observing strictly less serialized graph context, indicating that the descriptors encode information that the baseline must otherwise recover from a larger neighborhood.

We observe similar phenomenon in prompt length. Table 4 reports the mean number of characters for compressed node-classification prompts on Cora across graph sizes from 50 to 1000 nodes and for two graph types. On BA graphs, the baseline prompt length at $n = 1000$ is about $4\times$ that of the WL-based variants. On ER graphs, the reduction is even larger (nearly $15\times$). The effect is strongest on sparse graphs, where a 1-hop retained subgraph remains small while the full serialization continues to scale with graph size. This shows that the benefit of WL-based prompting is not merely predictive but also representational.

5.4 Importance of Colors

Our method translates ordered WL labels into natural-language color words rather than into arbitrary identifiers. To test whether the observed gains come from the structural information or from expressing that information in a semantically grounded vocabulary due to compatibility with LLM's linguistic priors.

First, we compare color words against alternative vocabularies that preserve the same WL equivalence classes but remove their natural-language grounding. We evaluate node classification on Cora using over-sampled subgraphs of sizes 10, 20, 30, 40, and 50, and replace color words with either synthetic hue identifiers (e.g., "Hue i") or human names. Table 5 shows that natural-language colors substantially outperform both alternatives.

Since these variants preserve the same underlying WL partition, the large performance gap cannot be explained solely by structural information, meaning the LLM better exploits WL-derived structure when it is expressed using semantically meaningful color terms rather than opaque tokens.

We next study how many distinct color descriptors are needed, since large graphs may yield more WL labels that cannot be uniquely represented with a reasonably small palette. We vary the number of

Variant	Accuracy (%)	MAE ↓	RMSE ↓
TLG-A	49.0	0.510	0.714
C-OWL	**68.9**	**0.311**	**0.558**
L-OWL	63.6	0.364	0.604
CL-OWL	66.4	0.336	0.580
WL Colors (hue)	15.2	0.848	0.921
WL Colors (names)	6.3	0.937	0.968

Table 5: Effect of the textual realization on node classification. Hue identifiers and names preserve the same WL class assignment as C-OWL but replace semantically grounded color words with opaque tokens.

Variant	Acc. (%)	MAE ↓	# Colors	CF
TLG-A	60.0	0.400	–	–
C3	65.0	0.350	3	10.6
C6	**75.0**	**0.250**	6	5.3
C9-0	**75.0**	**0.250**	9	3.5
C9-3	**75.0**	**0.250**	27	1.2
C9-5	70.0	0.300	45	0.7

Table 6: Color-cardinality ablation. C3, C6, and C9-0 use 3, 6, and 9 base color words, respectively. C9-3 and C9-5 augment the 9-color palette with 3 and 5 lightness levels (i.e., 27 and 45 distinct color descriptors). Collision denotes the ratio between the number of WL labels and available color descriptors. CF is collision factor.

available color descriptors in C-OWL on 50-node Cora subgraphs. We use three palettes with only base colors (C3, C6, C9-0), and two larger palettes formed by combining nine base colors with lightness modifiers, yielding 27 colors (C9-3) and 45 colors (C9-5). In this dataset, WL refinement produces about 32 distinct labels per graph, so limited palettes induce collisions, where multiple WL labels are mapped to the same color descriptor.

Table 6 shows two main patterns. First, all color-based variants outperform the baseline, even when the collision factor is high, showing the usefulness of approximate grouping of structurally similar nodes. Second, performance improves with more color descriptors as collisions decrease. The strongest results are obtained for collision factors between roughly 1 and 5, while performance drops again when the palette becomes overly fine-grained. This suggests that too few colors merge distinct structural roles, while too many descriptors weaken the regularizing effect of shared human-interpretable categories.

5.5 Performance on Prediction Tasks

Besides the algorithmic tasks, we also show the effectiveness of CL-OWL for node classification. Table 7 presents the results. We first sample 30 graphs of five different sizes (10-50 nodes). Then, we sample 150 subgraphs from all the graphs. Because of this, the dataset contains 600 graph-task instances in total. Overall, incorporating WL-based structural information improves performance over the baseline, indicating that explicit structural cues can benefit predictive tasks beyond purely algorithmic reasoning. The color-only variant (C-OWL) achieves the strongest results on Cora, Citeseer, and PubMed. In contrast, the color-label variant (CL-OWL) performs best on the larger and structurally more complex OGBN-ArXiv dataset, where human-tangible similarity cues appear to better support long-range structural dependencies. These results highlight that the relative benefit of different encodings depends on dataset scale and complexity, with color-based representations becoming more advantageous as graphs grow larger.

Method	Cora	Citeseer	PubMed	OGBN-ArXiv
TLG-A	11.35	9.01	5.70	22.76
L-OWL	18.78	13.83	13.23	22.61
C-OWL	**22.30**	**20.08**	**14.33**	26.87
CL-OWL	20.93	13.63	11.94	**40.94**

Table 7: Node-classification performance measured by F1-Macro (%). The color-only variant performs best on Cora, Citeseer and PubMed, with the Colors and Full method trailing behind. On ArXiv the full method performs best, followed by the labels.

Additional experiments including the effects on long range dependencies, and scalability are available in Appendix E.

6 Conclusion

We have studied how LLMs can be better adapted to graph-structured reasoning through human-tangible auxiliary structural information. By introducing CL-OWL, an ordered WL–based graph-to-text representation and mapping structural equivalence classes to natural-language color tokens, we provide a principled way to expose graph structure while preserving permutation invariance and linguistic alignment. We have provided a theoretical analysis connecting ordered WL refinement to distance-weighted connectivity, and a systematic empirical evaluation across diverse graph tasks and graph families. Our results show that human-tangible structural cues can substantially improve LLM performance on graph reasoning, particularly for tasks requiring global structural understanding.

7 Ethical Considerations

This work focuses on improving the representation of graph-structured data for large language models (LLMs) through human-interpretable auxiliary structural information. Our method operates solely on abstract graph topology and synthetic or standard benchmark datasets, and does not involve human subjects, personal data, or sensitive attributes. We do not introduce any deployment mechanisms that could amplify social bias, privacy risks, or misuse beyond those already associated with existing LLMs. As such, we do not foresee any ethical issues from our study.

Potential Risks. While our method operates on abstract graph structures and does not involve sensitive data, it introduces a representation bias by encoding structural information through human-interpretable tokens (e.g., colors). Such encodings may implicitly prioritize certain structural patterns over others, potentially leading to misinterpretation or over-reliance by LLMs in downstream tasks. Additionally, as our approach is prompt-based and leverages pretrained LLMs, it inherits known limitations of these models, including sensitivity to prompt design, lack of robustness to distribution shifts, and potential hallucinations in reasoning.

8 Limitations

We present some limitations of our work:

- *Generalization to NP-hard Problems.* Although we evaluate a diverse set of synthetic and real-world graphs, the task set does not exhaustively cover all graph reasoning problems, especially the NP-hard ones.
- *Human-interpretable Encodings are Heuristic.* While color-based encoding aligns well with human and linguistic priors, it is ultimately a heuristic design choice. Alternative interpretable representations may yield different or superior performance.

References

Peyman Baghershahi, Gregoire Fournier, Pranav Nyati, and Sourav Medya. 2025. From nodes to narratives: Explaining graph neural networks with llms and graph context. *Preprint*, arXiv:2508.07117.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024. Llaga: Large language and graph assistant. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Dongqi Fu, Liri Fang, Zihao Li, Zhe Xu, Hanghang Tong, Vetle I. Torvik, and Jingrui He. 2025. Bring Complex Geometric Information to LLMs: A Positional Survey of Graph Parametric Representation. In *The Fourth Learning on Graphs Conference*.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data ? An Empirical Evaluation and Benchmarking.

Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. 2024. UniGraph: Learning a Unified Cross-Domain Foundation Model for Text-Attributed Graphs.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024a. Large Language Models on Graphs: A Comprehensive Survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8622–8642.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024b. Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning on Graphs.

Jintang Li, Ruofan Wu, Yuchang Zhu, Huizhe Zhang, Liang Chen, and Zibin Zheng. 2025. Are Large Language Models In-Context Graph Learners?

Rui Li, Jiwei Li, Jiawei Han, and Guoyin Wang. 2024. Similarity-based Neighbor Selection for Graph LLMs.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024a. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V. Chawla. 2024b. Can we Soft Prompt LLMs for Graph Learning Tasks? In *Companion Proceedings of the ACM Web Conference 2024*, pages 481–484.

Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2020. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33:20000–20011.

Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. 2024. Position: graph foundation models are already here. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled Representation Learning with Large Language Models for Text-Attributed Graphs.

Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. 2022. Greed: A neural framework for learning graph distance functions. *Advances in Neural Information Processing Systems*, 35:22518–22530.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh V. Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 6616–6626. ACM.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. 2024. Understanding transformer reasoning capabilities via graph algorithms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561.

Shengyin Sun, Yuxiang Ren, Jiehao Chen, and Chen Ma. 2023. Large Language Models as Topological Structure Enhancers for Text-Attributed Graphs.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. 2025. GraphICL: Unlocking Graph Learning Potential in LLMs through Structured Prompt Design.

Yanchao Tan, Hang Lv, Pengxiang Zhan, Shiping Wang, and Carl Yang. 2024. Graph-oriented Instruction Tuning of Large Language Models for Generic Graph Mining.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 491–500. ACM.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yanbang Wang, Hejie Cui, and Jon M. Kleinberg. 2024. Microstructures and accuracy of graph recall by large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Zhe Xu, Kaveh Hassani, Si Zhang, Hanqing Zeng, Michihiro Yasunaga, Limei Wang, Dongqi Fu, Ning Yao, Bo Long, and Hanghang Tong. 2024. How to Make LLMs Strong Node Classifiers?

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 40–48.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1955–1973, St. Julian's, Malta. Association for Computational Linguistics.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024. Llm4dyg: Can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 4350–4361. ACM.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. GraphText: Graph Reasoning in Text Space.

Qi Zhu, Da Zheng, Xiang Song, Shichang Zhang, Bowen Jin, Yizhou Sun, and George Karypis. 2024. Parameter-Efficient Tuning Large Language Models for Graph Representation Learning.

Xi Zhu, Haochen Xue, Ziwei Zhao, Wujiang Xu, Jingyuan Huang, Minghao Guo, Qifan Wang, Kaixiong Zhou, Imran Razzak, and Yongfeng Zhang. 2025. LLM as GNN: Graph Vocabulary Learning for Text-Attributed Graph Foundation Models.

A Additional Details on Related Work

Locality-driven interfaces. The most common strategy restricts the textual description of a graph to local neighborhoods, such as k-hop ego-nets, short paths, or small induced subgraphs. This approach reduces prompt length and stabilizes generation, but obscures long-range dependencies required for global reasoning. Many early and recent solver-oriented works adopt locality-driven serialization, including NLGraph-style benchmarks and graph-in-text prompting methods (Wang et al., 2023; Guo et al., 2023; Sun et al., 2023; Zhao et al., 2023). Similar locality biases appear in text-attributed graph pipelines and hybrid graph–LLM systems that operate on bounded-depth neighborhoods (Qin et al., 2023; Ye et al., 2024; Chen et al., 2024). Empirical analyses show that such locality constraints correlate with failures on tasks requiring global connectivity or cycle reasoning (Wang et al., 2024; Fu et al., 2025).

Similarity-driven interfaces. A smaller body of work selects which nodes or substructures to expose to the LLM based on similarity under a learned encoder, often using cosine similarity in embedding space. The goal is to group semantically related nodes rather than strictly adjacent ones. However, similarity in text or embedding space does not necessarily correspond to topological similarity, and these methods introduce additional design complexity. Representative examples include similarity-based neighbor selection and retrieval strategies for graph LLMs (Li et al., 2024; Xu et al., 2024; Sun et al., 2025; Li et al., 2025).

Embedding-based interfaces. Another line of work injects continuous graph representations into LLMs by first encoding structure with a GNN and then aligning the resulting embeddings with the LLM input space via projectors or soft prompts. These methods can convey rich structural information but reduce transparency and often require training or tuning. Examples include LLaGA and related graph–LLM architectures that integrate GNN embeddings into language models (Chen et al., 2024; Liu et al., 2024b; Zhu et al., 2024; He et al., 2024; Tang et al., 2024).

Textual mapping to discrete symbols. A complementary strategy converts structural or embedding-derived information into discrete textual tokens that can be directly consumed by LLMs. Talk Like a Graph systematically studies how different graph encodings affect LLM reasoning (Fatemi et al., 2024), while other works map graph structure or learned signals into labels or symbolic descriptions (Zhao et al., 2023; Tang et al., 2024). AuGLM is among the few approaches that explicitly translate embedding-derived information into textual labels for LLM consumption (Xu et al., 2024). These methods are closest in spirit to our work.

Our approach belongs to the textual mapping category. Unlike prior methods that rely on arbitrary numeric labels or post hoc embedding translation, we derive a task-agnostic structural signal using Weisfeiler-Leman refinement and encode it using human-interpretable, similarity-preserving textual attributes. This design maintains a fully text-based interface while exposing non-local structural information aligned with the inductive biases of LLMs.

B Additional Details on Our Method

B.1 Proof of Theorem 1

Here we provide the proof for Theorem 1 described in Section 4.1.

Proof sketch. **(1)** At $t = 0$ all nodes have $\ell_v^{(0)} = 1$, hence $m_v^{(0)} = (1, (1, 1, \ldots, 1))$ where the tuple length equals $\deg(v)$. Lexicographic order compares tuples first by prefix and then by length; since all entries are 1, larger degree yields a lexicographically larger message, so $\mathrm{ID}^{(0)}$ assigns a larger integer: $\deg(v) > \deg(w) \Rightarrow \ell_v^{(1)} > \ell_w^{(1)}$.

(2) Under the tree-unfolding assumption up to depth T, the T-hop rooted neighborhood of a node can be represented by a rooted tree in which the number of nodes at depth k equals $|S_k(\cdot)|$. Ordered 1-WL on such a tree corresponds to bottom-up aggregation of sorted multisets of child labels. If $|S_k(v)| \geq |S_k(w)|$ for all $k \leq T$ with strict inequality at some $k^\star$, then at the earliest depth where the rooted trees differ, the multiset (hence the sorted tuple) of child labels for v strictly dominates that of w in lexicographic order. Because $\mathrm{ID}^{(t)}$ is order-preserving over messages, this dominance propagates through subsequent iterations, yielding $\ell_v^{(T)} > \ell_w^{(T)}$.

(3) If α_k are nonincreasing and positive, shell dominance with a strict inequality implies $C_T(v) > C_T(w)$ immediately by summation. Combining with (2) yields consistency between the induced WL order and $C_T(\cdot)$ for this class of local neighborhoods. □

Algorithm 1 Ordered 1-WL Refinement (Canonical Relabeling)

Require: Graph $G = (V, E)$
Ensure: Final node labels $\ell : V \to \mathbb{N}$ (and optionally the full history)

Initialize $\ell_v \leftarrow 1$ for all $v \in V$
repeat
  for all $v \in V$ **do**
    $M_v \leftarrow \mathrm{sort}\big(\{\{\ell_u : u \in \mathcal{N}(v)\}\}\big)$
    $m_v \leftarrow (\ell_v, M_v)$ ▷ node message
  end for
  $U \leftarrow \mathrm{sort}\big(\{m_v : v \in V\}\big)$ ▷ unique messages, lexicographically sorted
  Define $\mathrm{ID}(m)$ as the index of m in U (0-based or 1-based)
  for all $v \in V$ **do**
    $\ell'_v \leftarrow \mathrm{ID}(m_v)$
  end for
  $\ell \leftarrow \ell'$
until ℓ does not change
return ℓ

Algorithm 2 Map WL Labels to Color Names

Require: Labels ℓ_v for all $v \in V$
Ensure: Color annotation $\mathrm{Color}(v)$ for all $v \in V$

$m_{\min} \leftarrow \min_{v \in V} \ell_v, \quad m_{\max} \leftarrow \max_{v \in V} \ell_v$
$r \leftarrow \max(1, m_{\max} - m_{\min})$
for all $v \in V$ **do**
  $x_v \leftarrow (\ell_v - m_{\min})/r$ ▷ $x_v \in [0, 1]$
  $h_v \leftarrow h_{\min} + (h_{\max} - h_{\min}) \cdot x_v$ ▷ e.g., $[h_{\min}, h_{\max}] = [0°, 180°]$
  $\mathrm{Color}(v) \leftarrow \mathrm{HueToName}(h_v)$
end for
return Color

B.2 The Pseudocodes

The pseudocode of ordered 1-WL refinement is shown in Algorithm 1. The same for color mapping is shown in Algorithm 2.

B.3 Prompt Structure

The prompt follows a clearly defined structure. First of all it is split in two parts: system and user prompt. To better visualize the prompt structure we can refer to figures 3, 4, 5. More specifically, Figure 3 solves max-flow with the CL-OWL method, Figure 4 solves cycle check with the L-OWL method and Figure 5 solves shortest path with the C-OWL method. The system prompt contains information that is useful to explain to the LLM its role in the conversation. It contains information such as "you are an expert in ...", explanation of the syntax that will be used and general guidelines (e.g. "think carefully before responding"). The user prompt contains the task definition, the graph and the explanation for both labels and colors. Blocks are marked explicitly using tokens such as «GRAPH», «WL_LABELS», or «COLORS» to simplify parsing. Answer template: forced answer pattern such as «ANSWER» Yes/No or «ANSWER» [number] to try to make evaluation deterministic. We additionally include a short instruction such as "Think step-by-step and reason to solve the task." This prompt pattern is widely used in LLM literature and helps stabilize reasoning without introducing model-specific biases.

C Reproducibility

Code. The code is available at: https://github.com/angelozangari/CL-OWL

The entire pipeline is designed to be easily executable and reproducible. Easiness of **executability** is guaranteed by one main *./run.sh* script that automates all the pipeline stages. This includes the initial setup, which ensures that all the required packages are installed, and sets up the python virtual environment with the required packages. Then one can proceed with the dataset generation, followed by the LLM testing, the automatic parsing and the computation of the results. Each of these steps is executable singularly with a dedicated script command. Each configuration was evaluated with a single run using a different random seed. We do not report standard deviation as it is not highly relevant in our evaluation context. We generate different random graphs and the answers of the graph queries are averaged on these graphs.

Regarding **reproducibility**, all the parameters in the pipeline are in a dedicated config/ folder, which contains exclusively *.yaml* files, in which all the needed parameters for dataset generation, LLM testing and reporting reside. Moreover, all the parts of the prompts (from the pre-task definition to the task instruction) have their content entirely confined to the *config/* directory too. This way, if anyone wants to change the content of the prompt they can simply edit the configuration file of the prompt and avoid modifying the code.

Graph Type	Size	Avg Edges	Avg Degree	Avg Diameter	Density
BA	10	20.4 ± 5.6	4.08 ± 1.12	2.1 ± 0.3	0.454
BA	15	43.7 ± 12.2	5.83 ± 1.63	2.4 ± 0.5	0.417
BA	20	76.4 ± 17.1	7.64 ± 1.71	2.4 ± 0.5	0.402
BA	25	114.2 ± 31.6	9.13 ± 2.53	2.4 ± 0.5	0.381
BA	30	171.3 ± 40.6	11.42 ± 2.71	2.3 ± 0.5	0.394
ER	10	9.7 ± 2.6	1.95 ± 0.51	4.8 ± 0.9	0.216
ER	15	20.1 ± 3.6	2.68 ± 0.48	5.3 ± 1.5	0.192
ER	20	38.2 ± 5.8	3.82 ± 0.58	4.5 ± 0.8	0.201
ER	25	59.9 ± 7.2	4.79 ± 0.58	4.0 ± 0.3	0.200
ER	30	87.0 ± 8.8	5.80 ± 0.59	4.0 ± 0.7	0.200

Table 8: Synthetic graph properties by type and size for the dataset used in algorithmic tasks 5.2. BA denotes Barabasi Albert and ER denotes Erdos Renyi.

Graph Type	Size	Avg Edges	Avg Degree	Avg Diameter	Density
Erdos Renyi	75	41.0 ± 0.0	1.09 ± 0.00	N/A	0.015
Erdos Renyi	100	72.4 ± 8.4	1.45 ± 0.17	N/A	0.015
Path	50	49.0 ± 0.0	1.96 ± 0.00	49.0 ± 0.0	0.040
Path	75	74.0 ± 0.0	1.97 ± 0.00	N/A	0.027
Path	100	99.0 ± 0.0	1.98 ± 0.00	N/A	0.020

Table 9: Synthetic graph properties by type and size for the dataset used in the experiments about long range dependencies E.1 and scalability 5.3 in the main paper.

D Detailed Experimental Setup

Synthetic Data Generation. When generating synthetic data we can choose the specific number of nodes we want our graphs to have, the graph type, the task to be solved and how many of this graph-type-task combination we want to generate. The specific statistics of the datasets used in section 5.

Real-World Datasets. We consider four widely used real-world citation and academic graphs: Cora, Citeseer, PubMed, and OGBN-ArXiv. For each dataset, we sample induced subgraphs of varying sizes as described in Section 5.5, and compute structural statistics over the resulting instances.

Table 10 reports the average number of edges, average degree, diameter, and density as a function of subgraph size. Across all datasets, we observe a consistent increase in the number of edges and average degree with graph size, while graph density decreases, reflecting the sparsity typical of real-world graphs. Diameters grow sublinearly with size, indicating that sampled subgraphs preserve small-world characteristics. OGBN-ArXiv subgraphs tend to exhibit higher average degree and variance compared to the other datasets, consistent with its denser and more heterogeneous global structure.

E Additional Results

This section reports additional experiments. They cover external graph-language baselines, lexical robustness of the color vocabulary, color-cardinality ablations, compressed prompting, local motif recognition, and prompt-length compression.

E.1 Effects on Long Range Dependencies

As we discussed in Section 3, one of the major motivations of our design is to capture long-range dependencies (e.g., global structure). To demonstrate this, we evaluate performance on the maximum flow task where the source and target nodes are significantly farther apart. To achieve this, we construct 120 graphs with 50, 75, 100 nodes of path and ER graphs with $p = 0.015$ (sparse, for high diameter), such that all the graphs' diameters g_d are above a threshold, i.e., $g_d > 10$.

Table 11 shows the results over different source-to-target distance ranges. Clearly, enriching the prompt with WL-labels not only enhances performance across all ranges, but also the improvement is larger as the distance increases. Indeed, encoding graphs from their non-Euclidean space to the textual sequence where position matters might make two nodes appear farther from each other in the sequence compared to their original shortest (path) distance in the graph. Therefore, attending to the WL-labels allows the LLM to compensate for the noisy signals from the positional-encoding module as its information is affected by the appearance position of the encoded node/edge tokens.

E.2 Scalability

We next evaluate whether the same structural advantage persists when graph size increases in hard reasoning tasks. Although modern LLMs support larger context windows, performance still deteriorates as prompts become longer and graph structure becomes harder to recover from a linearized serialization. We therefore study the same sparse high-diameter setting as in Section E.1, but now stratify results by graph size rather than by source–target distance. Table 12 reports accuracy for Maximum Flow and Shortest Path on graphs with 50, 75, and 100 nodes. Performance degrades substantially with graph size for all methods, confirming that larger graphs remain challenging even when they fit within context. However, WL-based variants mitigate this degradation. For Maximum Flow, the baseline is near failure across sizes, with aggregate

Dataset	Size	Avg Edges	Avg Degree	Avg Diameter	Density
Cora	10	12.4 ± 2.4	2.49 ± 0.48	3.0 ± 0.7	0.276
	20	29.7 ± 6.7	2.97 ± 0.67	3.9 ± 0.9	0.156
	30	46.3 ± 9.7	3.09 ± 0.64	4.5 ± 1.1	0.106
	40	61.8 ± 14.1	3.09 ± 0.71	4.6 ± 1.1	0.079
	50	78.7 ± 10.1	3.15 ± 0.40	5.1 ± 1.2	0.064
Citeseer	10	12.8 ± 3.3	2.56 ± 0.67	3.3 ± 0.8	0.285
	20	28.1 ± 6.7	2.81 ± 0.67	4.5 ± 1.6	0.148
	30	43.3 ± 6.7	2.89 ± 0.45	5.7 ± 1.8	0.100
	40	57.7 ± 10.3	2.89 ± 0.51	6.1 ± 1.8	0.074
	50	74.7 ± 12.3	2.99 ± 0.49	6.2 ± 1.8	0.061
PubMed	10	11.1 ± 1.8	2.21 ± 0.36	2.4 ± 0.5	0.246
	20	26.2 ± 6.6	2.62 ± 0.66	3.3 ± 0.8	0.138
	30	41.5 ± 11.0	2.77 ± 0.74	3.9 ± 0.7	0.095
	40	59.8 ± 14.1	2.99 ± 0.70	4.2 ± 0.7	0.077
	50	81.5 ± 24.3	3.26 ± 0.97	4.2 ± 0.7	0.067
OGBN-ArXiv	10	12.7 ± 2.5	2.54 ± 0.51	2.6 ± 0.6	0.282
	20	34.6 ± 13.0	3.46 ± 1.30	3.4 ± 1.1	0.182
	30	63.0 ± 30.3	4.20 ± 2.02	3.9 ± 1.2	0.145
	40	86.9 ± 31.0	4.35 ± 1.55	4.1 ± 1.1	0.111
	50	131.6 ± 44.5	5.26 ± 1.78	3.8 ± 0.8	0.107

Table 10: Graph statistics for sampled subgraphs used in node-classification experiments (Section 5.5). Values are reported as mean $\pm$ standard deviation over sampled subgraphs.

	Source-to-Target Distance Range							
	Maximum Flow				**Shortest Path**			
Method	10–15	16–25	26–40	41+	10–15	16–25	26–40	41+
TLG-A	3.1	0.0	5.0	0.0	6.5	**6.2**	**20.0**	0.0
C-OWL	12.5	9.1	15.0	**31.2**	0.0	0.0	**20.0**	**11.8**
L-OWL	15.6	**27.3**	**25.0**	6.2	6.5	**6.2**	0.0	5.9
CL-OWL	**25.0**	9.1	5.0	18.8	**12.9**	**6.2**	6.7	5.9

Table 11: Accuracy (%) by source–target distance range for Maximum Flow and Shortest Path. Best values per column are in bold; second-best are underlined. Our C-OWL variant generally outperforms the baselines. CL-OWL achieves superior performance on close range. L-OWL performs best on medium to long ranges, and color-based variants achieve superior performance on the longest range.

accuracy only 2.53%, whereas L-OWL and the color-based variants raise aggregate performance to 16.46–17.72%. For Shortest Path, the gains are smaller and less uniform, but CL-OWL attains the strongest aggregate performance at 8.86%. These results mirror the long-range analysis in Section E.1: once direct structural recovery from the serialized graph becomes unreliable, WL-derived descriptors provide a more robust signal.

To isolate this large-graph effect more directly, Table 13 compares Maximum Flow accuracy between 50-node and 100-node Erdős–Rényi and path graphs. All WL-based variants remain substantially stronger than the baseline at both sizes. Moreover, the degradation from 50 to 100 nodes is much smaller for the color-based variants: C-OWL drops by only 6.8 points, compared to 45.0 points for the baseline, 40.9 for L-OWL, and 33.3 for CL-OWL. While CL-OWL achieves the highest accuracy at 50 nodes, C-OWL is strongest at 100 nodes and exhibits the most stable scaling behavior. This suggests that the color-based realization of ordered WL structure is particularly helpful once graphs become large enough that the raw serialized edge list no longer reliably exposes the relevant global organization.

E.3 Additional Baseline Comparisons

We further compare our method against recent graph-language baselines on node classification over Cora subgraphs. We follow the same evaluation setting as in Section 5.5: subgraphs of sizes 10, 20, 30, 40, and 50 nodes, with 30 sampled graphs per size, yielding 150 graph-task instances in total. For the external baselines, we follow the protocols provided by the original authors. GraphText is eval-

	Maximum Flow				Shortest Path			
Method	50	75	100	All	50	75	100	All
TLG-A	5.00	4.76	0.00	2.53	5.00	**14.29**	5.26	7.59
C-OWL	45.00	4.76	7.89	16.46	0.00	9.52	**7.89**	6.33
L-OWL	**50.00**	**9.52**	5.26	**17.72**	10.00	4.76	2.63	5.06
CL-OWL	30.00	4.76	**15.79**	16.46	**20.00**	**14.29**	0.00	**8.86**

Table 12: Scalability of Maximum Flow and Shortest Path for different graph sizes. We report accuracy (%). The best is in bold, and the second-best is underlined. Accuracy degrades rapidly with graph size for both tasks, with WL-based methods mitigating but not eliminating scaling failures. Note that the highest graph size used in (Fatemi et al., 2024) for TLG is 20.

Variant	Acc. (50)	Acc. (100)	Δ
TLG-A	45.0	0.0	-45.0
L-OWL	90.9	50.0	-40.9
CL-OWL	**100.0**	66.7	-33.3
C-OWL	81.8	**75.0**	**-6.8**

Table 13: Maximum-flow accuracy (%) on Erdős–Rényi and path graphs with 50 and 100 nodes. Δ denotes the change in accuracy from 50 to 100 nodes; smaller absolute degradation is better. Best values per column are in bold; second-best are underlined.

Method	Type	Acc. (%)
LLaGA	Graph-LLM	15.0
GraphText	Graph-text	49.7
OFA	Graph model	16.3
TLG-A	Prompt baseline	49.0
L-OWL	Ours	63.6
C-OWL	Ours	**68.9**
CL-OWL	Ours	66.4

Table 14: Comparison with external baselines on Cora node classification. Accuracy is reported over all sampled instances. GraphText uses its official inference pipeline, LLaGA the released checkpoint, and OFA the official codebase with the paper-reported supervised node-classification setting.

Variant	Acc. (%)	MAE ↓	RMSE ↓
TLG-A	63.7	0.363	0.603
C-OWL	65.8	0.342	0.585
L-OWL	66.0	0.340	0.583
CL-OWL	**68.3**	**0.317**	**0.563**

Table 15: Triangle-membership results on a localized three-node query task. Unlike graph-level triangle counting, this formulation isolates local motif recognition by asking whether a queried triple forms a triangle.

uated in inference mode using its official pipeline; LLaGA is evaluated with the released pretrained checkpoint; and OFA is trained and evaluated using the official codebase with the paper-reported supervised node-classification setting (100 epochs, learning rate 10^{-4}, batch size 128). All methods are evaluated on the same sampled instances.

Table 14 shows that all variants of our method outperform the external baselines as well as the TLG-A prompt baseline. In particular, C-OWL achieves the best overall accuracy at 68.9%, followed by CL-OWL at 66.4% and L-OWL at 63.6%. GraphText performs similarly to TLG-A, while LLaGA and OFA are substantially weaker in this setting. These results indicate that the gains observed in Section 5.5 are not merely due to applying an LLM to citation-graph subproblems, but are specifically associated with exposing ordered WL structure through the proposed textual encoding.

E.4 Local Motif Recognition

The graph-level triangle-counting task in Table 2 requires counting all triangles in the graph, which combines local motif detection with a global aggregation burden over the full serialized graph. Since our method enriches node-level representations through WL-derived structural descriptors, a weaker result on graph-level triangle counting does not necessarily imply a weakness in recognizing local triangular structure itself. To isolate this distinction, we introduce a three-node triangle-membership task that asks whether a queried node triple forms a triangle.

Table 15 shows that all WL-based variants improve over the TLG-A baseline on this localized formulation, with CL-OWL achieving the strongest performance at 68.3% accuracy compared to 63.7% for the baseline. The same trend is reflected in MAE and RMSE, where CL-OWL also performs best. This indicates that WL-based labels and colors do help the LLM recognize local motif structure when the task is aligned with the node-level information injected by the prompt.

Taken together with the results of Section 5.2, this experiment suggests that the weaker perfor-

mance on graph-level triangle counting is primarily a task-formulation issue. For local motif recognition, where the relevant information is concentrated around a small queried set of nodes, the proposed node-level structural augmentation is beneficial.

E.5 WL and Color Descriptor Statistics

To contextualize the design choices behind the WL-based descriptors, we report summary statistics of the refinement process on the main synthetic reasoning datasets. Table 16 shows the number of WL iterations required for stabilization and the number of distinct structural labels and color descriptors produced at each graph size. Since cycle check, maximum flow, reachability, shortest path, and triangle counting are instantiated on the same generated graph pool, these statistics are identical across tasks for a fixed graph size and are therefore reported only once per size.

Two patterns are clear. First, the number of WL iterations needed for stabilization remains small across the main reasoning regime, typically around three to four iterations for graphs with 15–30 nodes. This supports the use of a small fixed WL refinement budget in these experiments. Second, the number of distinct WL labels grows steadily with graph size, which implies that larger graphs require either a larger descriptor vocabulary or a controlled level of collisions when labels are mapped to natural-language colors.

This behavior is also consistent with the color-cardinality ablation in Section 5.4. In the 50-node Cora node-classification setting used there, WL refinement requires 4.00 ± 0.63 iterations (range 3–5) and produces 31.80 ± 8.89 distinct WL labels per graph (range 18–46). With a small base color vocabulary, multiple WL labels must therefore map to the same color descriptor. Expanding the vocabulary with lightness modifiers reduces these collisions, but the results in Table 6 show that performance saturates once the effective collision factor is roughly between one and three. Overall, these statistics provide a descriptive account of the structural resolution induced by ordered WL refinement and help explain the empirical tradeoffs observed in the color-vocabulary ablations.

E.6 Full results on Graph Reasoning Tasks

Tables 17 and 18 report per-task reasoning performance stratified by graph type (BA, ER) and aggregated across graph types. These tables provide a finer-grained view of model behavior beyond the aggregated results discussed in Section 5.2.

Across tasks, we observe substantial variability between BA and ER graphs, particularly for tasks requiring global reasoning such as Maximum Flow and Shortest Path. Performance on BA graphs is generally higher for cycle-related tasks, while ER graphs often exhibit increased difficulty for flow- and path-based reasoning, reflecting differences in graph structure and connectivity patterns. Aggregated results therefore mask non-trivial graph-type effects that are visible only at this granularity. While WL-enriched prompting improves performance for several tasks under `gpt-4o`, the same trends are less consistent for `gpt-3.5-turbo`, suggesting that the effectiveness of structural encodings depends on the underlying model's reasoning capacity.

Overall, these detailed results support the conclusions drawn in the main text while highlighting the sensitivity of graph reasoning performance to both graph type and model generation.

Size	WL iters	WL labels	RGB colors	Color names
10	2.78±0.79 (2–4)	6.67±3.13 (2–10)	6.67±3.13 (2–10)	4.67±1.63 (2–6)
15	3.78±0.79 (2–5)	12.56±3.89 (4–15)	12.56±3.89 (4–15)	5.67±0.67 (4–6)
20	3.78±0.92 (3–6)	18.22±2.53 (12–20)	18.22±2.53 (12–20)	6.00±0.00 (6–6)
25	3.56±0.68 (3–5)	23.89±1.66 (21–25)	23.89±1.66 (21–25)	6.00±0.00 (6–6)
30	3.44±0.68 (3–5)	29.67±0.67 (28–30)	29.67±0.67 (28–30)	6.00±0.00 (6–6)

Table 16: WL refinement statistics for the main synthetic reasoning datasets, aggregated by graph size. Values are reported as mean ± standard deviation, with min–max ranges in parentheses.

Task	Graph	TLG-A		TLG-F		C-OWL		L-OWL		CL-OWL	
		Acc. (%)	MAE	Acc. (%)	MAE	Acc. (%)	MAE	Acc. (%)	MAE	Acc. (%)	MAE
Cycle Check	BA	100.00	0.0000	98.00	0.0200	100.00	0.0000	99.00	0.0100	100.00	0.0000
	ER	79.00	0.2100	85.00	0.1500	85.00	0.1500	85.00	0.1500	86.00	0.1400
	All	89.50	0.1050	91.50	0.0850	92.50	0.0750	92.00	0.0800	**93.00**	**0.0700**
Maximum Flow	BA	26.09	0.5233	34.78	0.4954	30.43	0.4863	17.39	0.6408	30.43	0.3564
	ER	35.00	0.4870	37.00	0.4337	38.00	0.4330	41.00	0.4370	38.00	0.4573
	All	33.33	0.4938	36.59	0.4452	36.59	0.4430	36.59	0.4751	**36.59**	**0.4385**
Shortest Path	BA	73.91	0.0870	78.26	0.0725	82.61	0.0580	78.26	0.0725	73.91	0.0870
	ER	91.00	0.0308	85.00	0.0447	90.00	0.0349	95.00	0.0137	90.00	0.0317
	All	87.80	0.0413	83.74	0.0499	88.62	0.0392	**91.87**	**0.0247**	86.99	0.0420
Triangle Count	BA	11.00	0.4438	11.00	0.4701	12.00	0.4495	13.00	0.4660	13.00	0.3996
	ER	21.00	0.4674	12.00	0.5764	16.00	0.4579	15.00	0.5509	16.00	0.5377
	All	**16.00**	**0.4556**	11.50	0.5233	14.00	0.4537	14.00	0.5085	14.50	0.4686

Table 17: Per-task reasoning performance for `gpt-4o`, stratified by graph type (BA, ER) and aggregated (*All*). Accuracy is higher-is-better; MAE is lower-is-better. Bold and underline indicate best and second-best methods, respectively, computed only on aggregated (*All*) rows.

Task	Graph	TLG-A		TLG-F		C-OWL		L-OWL		CL-OWL	
		Acc. (%)	MAE	Acc. (%)	MAE	Acc. (%)	MAE	Acc. (%)	MAE	Acc. (%)	MAE
Cycle Check	BA	66.00	0.3400	51.00	0.4900	66.00	0.3400	75.00	0.2500	72.00	0.2800
	ER	55.00	0.4500	52.00	0.4800	60.00	0.4000	54.00	0.4600	57.00	0.4300
	All	60.50	0.3950	51.50	0.4850	63.00	0.3700	**64.50**	**0.3550**	**64.50**	**0.3550**
Maximum Flow	BA	26.09	0.5746	21.74	0.6348	26.09	0.5971	30.43	0.5387	21.74	0.5522
	ER	15.00	0.7248	29.00	0.5377	10.00	0.7832	16.00	0.7032	11.00	0.7632
	All	17.07	0.6967	**27.64**	**0.5558**	13.01	0.7484	18.70	0.6724	13.01	0.7237
Shortest Path	BA	13.04	0.6232	8.70	0.6087	8.70	0.6377	8.70	0.6377	8.70	0.6377
	ER	14.00	0.6586	27.00	0.4347	9.00	0.7071	14.00	0.6653	10.00	0.6944
	All	13.82	0.6520	**23.58**	**0.4672**	8.94	0.6941	13.01	0.6602	9.76	0.6838
Triangle Count	BA	4.00	0.8557	4.00	0.8860	1.00	0.8616	6.00	0.8471	1.00	0.8818
	ER	10.00	0.7703	2.00	0.8686	5.00	0.7024	8.00	0.7930	7.00	0.6888
	All	**7.00**	0.8130	3.00	0.8773	3.00	**0.7820**	**7.00**	0.8201	4.00	0.7853

Table 18: Per-task reasoning performance for `gpt-3.5-turbo`, stratified by graph type (BA, ER) and aggregated (*All*). Accuracy is higher-is-better; MAE is lower-is-better. Bold and underline indicate best and second-best methods, respectively, computed only on aggregated (*All*) rows.

Prompt template for max-flow using the CL-OWL method

SYSTEM

```
You are an expert in graph theory and graph machine learning. You have deep knowledge of the
Weisfeiler-Leman (WL) algorithm, structural graph analysis, and graph visualization techniques.
Graph structure is provided as adjacency lists: each line is ‘node: neighbor1, neighbor2,
...‘ and isolated nodes use ‘(none)‘. Always follow the example format and present the final
answer after the ‘«ANSWER»‘ marker. Reason carefully about the graph before responding.
```

USER (task and format)

```
Your task is to return the maximum flow value between the «TARGET_PAIR» with unit capacities.
The following example demonstrates the format. A graph is represented as adjacency lists
where each line is ‘node: neighbor1, neighbor2, ...‘ (use ‘(none)‘ for isolated nodes).
‘«WL_LABELS»‘ shows Weisfeiler-Leman structural labels for each node. WL labels capture the
structural role of each node based on its neighborhood. WL labels are given the template:
WL(i)=k, meaning that node i has WL label k. Nodes with closer labels are more similar; for
instance if WL(node_1)=1, WL(node_2)=4, WL(node_3)=9, then node_1 is more similar to node_2
than to node_3. ‘«COLORS»‘ shows colors assigned based on WL label similarity. Colors capture
the structural role of each node based on its neighborhood. Colors are given the template:
Color(i)=c, meaning that node i has color c. Colors can be used as a similarity metric among
nodes (nodes with similar colors are more similar). ‘«TARGET_NODE»‘ marks the queried node;
‘«TARGET_PAIR» a,b‘ marks the queried pair for edge/pairwise tasks.
Always use the provided markers consistently and place the final answer after the ‘«ANSWER»‘
marker. Think step-by-step about the graph before responding.
```

«GRAPH»

```
«GRAPH» 0: 1, 2, 3, 4, 5, 6, 8. 1: 0, 5, 6, 7, 8, 9. 2: 0, 5. 3: 0, 5, 7, 8, 9. 4: 0, 6, 7.
5: 0, 1, 2, 3, 6, 7, 8. 6: 0, 1, 4, 5, 9. 7: 1, 3, 4, 5. 8: 0, 1, 3, 5, 9. 9: 1, 3, 6, 8.
```

«WL_LABELS»

```
WL(0):8, WL(1):7, WL(2):0, WL(3):5, WL(4):1, WL(5):9, WL(6):4, WL(7):2, WL(8):6, WL(9):3
```

«COLORS»

```
Color(0):teal, Color(1):lime, Color(2):red, Color(3):green, Color(4):crimson, Color(5):teal,
Color(6):lime, Color(7):orange, Color(8):green, Color(9):orange
```

«TARGET_PAIR»

```
«TARGET_PAIR» 2,9
Use the provided markers consistently and place the final answer after the «ANSWER» marker.
Think step-by-step about the graph before responding.
```

Figure 3: Prompt template for max-flow using the CL-OWL method

Prompt template for cycle check using L-OWL

SYSTEM

```
You are an expert in graph theory and graph machine learning. You have deep knowledge of the
Weisfeiler-Leman (WL) algorithm and structural graph analysis.
Graph structure is provided as adjacency lists: each line is ‘node: neighbor1, neighbor2, ...‘
and isolated nodes use ‘(none)‘.
Always follow the example format and present the final answer after the ‘«ANSWER»‘ marker.
Reason carefully about the graph before responding.
```

USER (task and format)

```
Your task is to answer Yes/No whether the «TARGET_NODE» is part of any cycle. The following
example demonstrates the format. A graph is represented as adjacency lists where each line
is ‘node: neighbor1, neighbor2, ...‘ (use ‘(none)‘ for isolated nodes). ‘«WL_LABELS»‘ shows
Weisfeiler-Leman structural labels for each node. WL labels capture the structural role of
each node based on its neighborhood. WL labels are given the template: WL(i)=k, meaning
that node i has WL label k. Nodes with closer labels are more similar; for instance if
WL(node_1)=1, WL(node_2)=4, WL(node_3)=9, then node_1 is more similar to node_2 than to
node_3. ‘«TARGET_NODE»‘ marks the queried node; ‘«TARGET_PAIR» a,b‘ marks the queried pair
for edge/pairwise tasks.
Always use the provided markers consistently and place the final answer after the ‘«ANSWER»‘
marker. Think step-by-step about the graph before responding.
```

«GRAPH»

```
«GRAPH» 0: 1, 2, 3, 4, 5, 6, 7, 8, 9 1: 0, 7, 8, 9 2: 0, 7 3: 0, 8, 9 4: 0, 7 5: 0, 7, 8, 9
6: 0, 7, 8 7: 0, 1, 2, 4, 5, 6, 8, 9 8: 0, 1, 3, 5, 6, 7, 9 9: 0, 1, 3, 5, 7, 8
```

«WL_LABELS»

```
«WL_LABELS» WL(0):7, WL(1):3, WL(2):0, WL(3):1, WL(4):0, WL(5):3, WL(6):2, WL(7):6, WL(8):5,
WL(9):4
```

«TARGET_NODE»

```
«TARGET_NODE» 7
Use the provided markers consistently and place the final answer after the ‘«ANSWER»‘ marker.
Think step-by-step about the graph before responding.
```

Figure 4: Prompt template for cycle check using L-OWL

Prompt template for shortest path using C-OWL

SYSTEM

You are an expert in graph theory and graph machine learning. You have deep knowledge of the Weisfeiler-Leman (WL) algorithm, structural graph analysis, and graph visualization techniques.
Graph structure is provided as adjacency lists: each line is ‘node: neighbor1, neighbor2, ...‘ and isolated nodes use ‘(none)‘.
Always follow the example format and present the final answer after the ‘«ANSWER»‘ marker.
Reason carefully about the graph before responding.

USER (task and format)

Your task is to return the shortest path length between the «TARGET_PAIR» (inf if no path). The following example demonstrates the format. A graph is represented as adjacency lists where each line is ‘node: neighbor1, neighbor2, ...‘ (use ‘(none)‘ for isolated nodes). ‘«COLORS»‘ shows colors assigned based on WL label similarity. Colors capture the structural role of each node based on its neighborhood. Colors are given the template: Color(i)=c, meaning that node i has color c. Colors can be used as a similarity metric among nodes (nodes with similar colors are more similar). ‘«TARGET_NODE»‘ marks the queried node; ‘«TARGET_PAIR» a,b‘ marks the queried pair for edge/pairwise tasks.
Always use the provided markers consistently and place the final answer after the ‘«ANSWER»‘ marker. Think step-by-step about the graph before responding.

«GRAPH»

«GRAPH» 0: 1, 2, 3, 4, 5, 6, 8 1: 0, 5, 6, 7, 8, 9 2: 0, 5 3: 0, 5, 7, 8, 9 4: 0, 6, 7 5: 0, 1, 2, 3, 6, 7, 8 6: 0, 1, 4, 5, 9 7: 1, 3, 4, 5 8: 0, 1, 3, 5, 9 9: 1, 3, 6, 8

«COLORS»

«COLORS» Color(0):teal, Color(1):lime, Color(2):red, Color(3):green, Color(4):crimson, Color(5):teal, Color(6):lime, Color(7):orange, Color(8):green, Color(9):orange

«TARGET_PAIR»

«TARGET_PAIR» 2,9
Use the provided markers consistently and place the final answer after the ‘«ANSWER»‘ marker. Think step-by-step about the graph before responding.

Figure 5: Prompt template for shortest path using C-OWL