Lost in Serialization: Invariance and Generalization of LLM Graph Reasoners

Daniel Herbst [*1] **Lea Karbevska** [*2] **Divyanshu Kumar** [*3]
Akanksha Ahuja [*2] **Fatemeh Gholamzadeh Nasrabadi** [*4]
Fabrizio Frasca [5]

Abstract

While promising, graph reasoners based on Large Language Models (LLMs) lack built-in invariance to symmetries in graph representations. Operating on sequential graph serializations, LLMs can produce different outputs under node reindexing, edge reordering, or formatting changes, raising robustness concerns. We systematically analyze these effects, studying how fine-tuning impacts encoding sensitivity and generalization on unseen tasks. We propose a principled decomposition of graph serializations into node labeling, edge encoding, and syntax, and evaluate LLM robustness to variations of each of these factors on a comprehensive benchmarking suite. We also contribute a novel set of spectral tasks to further assess generalization abilities of fine-tuned reasoners. Results show that larger (non-fine-tuned) models are more robust. Fine-tuning reduces sensitivity to node relabeling but may increase it to variations in structure and format, while it does not consistently improve performance on unseen tasks.

1. Introduction

Large Language Models (LLMs) are emerging as alternatives to Graph Neural Networks (GNNs) for graph reasoning (Gilmer et al., 2017; Xu et al., 2020; Dudzik & Veličković, 2022). While GNNs operate on adjacency and feature matrices, LLMs consume text-based serializations of nodes and edges, which enables reasoning in token space. LLMs can leverage their reasoning skills acquired on large-scale text corpora and support flexible in-context formulations of graph problems. Early out-of-the-box LLM applications have achieved limited success (Fatemi et al., 2023; Tang et al., 2025; Yuan et al., 2025), but subsequent fine-tuning strategies (Sanford et al., 2024; Guo et al., 2025) have substantially improved performance, allowing moderately sized models to perform nontrivial structural reasoning.

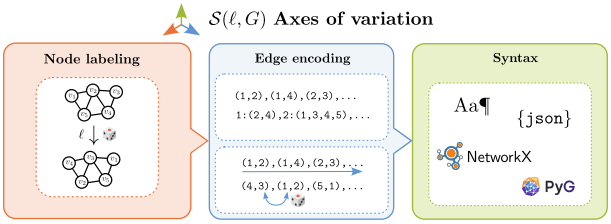


Figure 1. Systematic decomposition of a graph serialization into **node labeling**, **computational structure**, and **syntax**.

This progress raises a fundamental question: *what* is learned when graph reasoners are post-trained on serialized graphs? While GNNs are natively in-/equivariant w.r.t. the central graph symmetries, LLMs operating on textual encodings need not produce consistent outputs for equivalent graphs representations, raising robustness concerns (Fatemi et al., 2023; Yuan et al., 2025; Ge et al., 2025). Importantly, these variations are *not* equivalent from the model's perspective but can alter locality, parsing, and the algorithmic path taken in token space. As post-training shows promise in improving graph reasoning (Guo et al., 2025), it becomes critical to assess whether this mitigates or amplifies sensitivity to such variations, and how it affects generalization to unseen tasks.

Contributions. In this work, we study robustness and generalization of LLMs for graph reasoning. To this end, we make the following individual contributions:

- We propose a principled decomposition of graph serializations into **node labeling**, **computational structure**, and **syntax** (see fig. 1).
- Using the `Erdős` benchmark (Guo et al., 2025), we conduct a large-scale evaluation of how graph serialization affects LLM graph reasoning. We compare post-trained graph reasoners (`G1`, trained with supervised fine-tuning and reinforcement learning) against

[1]Technical University of Munich [2]University of Cambridge [3]Enkrypt AI [4]University of Amsterdam [5]Technion. Correspondence to: Fabrizio Frasca <fabriziof@campus.technion.ac.il>.

Workshop on Graph Foundation Models, held in conjunction with the 43rd International Conference on Machine Learning, Seoul, South Korea. 2026. Copyright 2026 by the author(s).

non-fine-tuned open-weight LLMs (`Qwen` (Bai et al., 2023), `gpt-oss` (OpenAI, 2025)), totaling **over 1.5M individual model inferences**.

- We introduce new spectral tasks to assess if graph reasoners generalize beyond purely combinatorial tasks.

We find that post-training reduces relabeling sensitivity, but increases sensitivity to computational structure and syntax, indicating specialization to the serialization present in the data. Importantly, our analysis suggests that in some cases, post-training can reinforce non-structural shortcuts that graph-native models would not observe. Thus, we argue that progress in LLM graph reasoning should not be judged solely from fixed-format accuracy, which motivates future work on invariance-aware post-training and benchmark design for LLM graph reasoners.

2. Related Work

Graphs, isomorphisms and inductive biases. Since pioneering works in the early 2000s (Gori et al., 2005; Scarselli et al., 2009), Graph Neural Networks (GNNs) have attracted significant attention in the Deep Learning community in recent years (Defferrard et al., 2016; Kipf & Welling, 2017; Veličković et al., 2018; Gilmer et al., 2017). This area has focused on a fundamental question: "*How to best design neural networks that operate on graph-structured data?*". The overarching principle advocated by Geometric Deep Learning is that of constraining these architectures to be in-/equivariant to the main symmetry of graphs, i.e., node relabelings (Bronstein et al., 2021). Permutation invariance, the central inductive bias of GNNs, ensures functions depend solely on the isomorphism class of the input graph, rather than any arbitrary labeling choice in their encodings.

Intriguingly, enforcing this inductive bias on Graph Neural Networks has profound consequences on their expressive power, as these architectures can be universal approximators only when parameterized with an intractable computational complexity (Maron et al., 2019). Importantly, Message-Passing Neural Networks (Gilmer et al., 2017), the most common and computationally efficient GNN variants, can even fail at straightforward graph tasks such as counting the number of substructures such as cycles (Chen et al., 2020).

LLMs on graphs and graph reasoning. Upon the advent of LLMs, researchers have started to investigate their applicability to more structured objects such as graphs. Not only have they started to explore hybrid methods integrating LLMs and GNNs (He et al., 2024; Yang et al., 2024; Liu et al., 2024), but, also, the pure application of LLMs directly operating on graph textual encodings (Zhao et al., 2024; Fatemi et al., 2023). This approach is particularly attractive because, by appropriate prompting in natural language, a single pretrained model can assist in interactive graph explorations, mining and zero-shot property predictions.

Within this trend of applying LLMs on graph-structured data, an emerging line of work specifically focuses on *graph reasoning*, which gathers tasks pertaining to answering queries or computing graph- or node-level parameters, such as shortest paths, connectivity, subgraph counting, and other structural inference problems. These tasks are of interest because they are conceptual representatives of general reasoning problems with dependencies (Besta et al., 2024; Sanford et al., 2024), and because they constitute relevant benchmarks to study LLM capabilities in capturing structural patterns beyond the limits of standard GNNs (Morris et al., 2019; Xu et al., 2019; Zhang et al., 2024; 2023).

Evaluating and improving LLMs for graph reasoning. A first evaluation of LLMs for graph reasoning has been presented in (Wang et al., 2023), followed by a more systematic approach by Fatemi et al. (2023) and later evaluations on newly released LLMs in (Tang et al., 2025; Yuan et al., 2025). The investigated models showed preliminary but still inadequate capabilities. Initial approaches include instruction tuning (Luo et al., 2024; Ye et al., 2024; Tang et al., 2025) preference optimization (Chen et al., 2024), as well as supervised fine-tuning (Sanford et al., 2024; Guo et al., 2025). These techniques can indeed boost the performance of LLMs as graph reasoners, but were shown to be outperformed by the more recent approach of Guo et al. (2025), whereby supervised fine-tuning is followed by a second stage operating the Group Relative Policy Optimization (GRPO) reinforcement learning algorithm. The authors release `G1`, a graph reasoner obtained by fine-tuning small-sized `Qwen` LLMs (Bai et al., 2023) on a curated corpus of reasoning tasks called `Erdős`. `G1` represents a leading purpose-built effort on graph reasoning, and `Erdős` stands as the most diverse post-training and benchmarking suite.

LLM sensitivity to textual graph encodings. Contrary to GNNs, LLMs graph reasoners are not intrinsically in-/equivariant to graph symmetries or representations. A first study on this aspect was conducted in (Fatemi et al., 2023) over PaLM models (Chowdhery et al., 2023) and a small set of simple reasoning tasks. Ge et al. (2025) later performed further analyses on edge-list representations, focusing on the impact of edge orderings for various LLMs on five reasoning tasks. The authors observed a pronounced preference for Breadth- (BFS) and Depth-First-Search (DFS) orderings compared to random ones, which consistently induced performance degradations. BFS ordering was found to yield better results on more local tasks, whereas DFS was found more competitive in tasks requiring a more global graph comprehension. Similar observations were made in Yuan et al. (2025), where lexicographic ordering was found to generally outperform random ones on a small set of tasks. In contrast to these analyses, this present work encompasses

a much larger set of reasoning tasks, considers multiple sources of variations in textual graph encodings and, importantly, also extends to post-trained LLMs. To the best of our knowledge, fine-tuned LLM graph reasoners are examined in their sensitivity only in (Chu et al., 2025), but the analyses are limited to edge reorderings on a small number of tasks.

3. LLMs on Graphs and Graph Serializations

3.1. Processing Graphs with LLMs

A graph is a tuple $G = (V, E)$, consisting of a finite node set V of size n and an edge set $E \subseteq V^2$, which can (optionally) be assigned a real-valued weight. Graphs must be practically stored *sequentially* for any kind of meaningful computation. This requires fixing a node labeling (ordering) $\ell : V \xrightarrow{\sim} [n], [n] := \{1, \dots, n\}$. Clearly, the relational structure encoded by G does not depend on (the arbitrarily chosen) ℓ. This motivates the central inductive bias of GNNs, i.e., in-/equivariance to such node relabelings. GNNs process graphs encoded algebraically as adjacency matrices $\boldsymbol{A}$ and are inherently constrained to be permutation in-/equivariant:

$$\mathrm{GNN}_{\mathrm{graph}}(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top) = \mathrm{GNN}_{\mathrm{graph}}(\boldsymbol{A}), \quad (1)$$

$$\mathrm{GNN}_{\mathrm{node}}(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top) = \boldsymbol{P}\,\mathrm{GNN}_{\mathrm{node}}(\boldsymbol{A}), \quad (2)$$

for any permutation $\boldsymbol{P}$, where $\mathrm{GNN}_{\mathrm{graph}}(\boldsymbol{A}) \in \mathbb{R}^d$ outputs a global graph-level, or $\mathrm{GNN}_{\mathrm{node}}(\boldsymbol{A}) \in \mathbb{R}^{n \times d}$ a node-level prediction.

Instead, LLMs operate on graphs encoded as *sequences* of tokens $\mathcal{S}(G, \ell) := (s_1, \dots, s_{Q_{\text{graph}}})$. They take as input both this and a task description $\mathcal{T} := (t_1, \dots, t_{Q_{\text{task}}})$ to generate an answer

$$\mathrm{LLM}(\mathcal{T}, \mathcal{S}(G, \ell)) := (y_1, \dots, y_A). \quad (3)$$

The answer tokens are iteratively sampled or greedily chosen from the autoregressive distribution defined by the parameters θ of the LLM's inner decode-only transformer:

$$y_t \sim p_\theta(\cdot | \mathcal{T}, \mathcal{S}(G, \ell), y_{<t}). \quad (4)$$

The generation proceeds until an end-of-sequence token is produced or the maximum output length is reached. Different from GNNs, a single LLM can tackle (virtually) any reasoning problem by prompt-conditioning via $\mathcal{T}$, a key advantage. However, LLMs are not permutation in-/equivariant by design. Distinct labelings ℓ and serializations $\mathcal{S}$ of the same graph yield distinct input strings for which LLMs may yield distinct outputs.

A sufficiently large network could, in principle, *approximate* any (continuous) permutation invariant function, but limitations stem from computational and sample complexity and, practically, the characteristics of the employed graph dataset. The network may, e.g., learn to exploit *positional* or *formatting regularities* in the available samples rather than capturing structural invariants. Consider, for instance, learning to compute a graph-level (e.g., maximum degree) or node-level (e.g., the maximum-degree node) parameter. If $(\ell, \mathcal{S})$ is confounded with the target, clues about the target can get directly leaked in the input representation. As an degenerate example, if the distinguished node v^* for a node-level task is always assigned the index $\ell(v^*) = n$, the target is trivially recoverable just from positional information, i.e., by reading out the parameters on the "last" node. Biases towards particular positional and formatting regularities may affect pretrained LLMs and, as we will demonstrate through experiments, can be amplified by post-training.

Next, we introduce a principled decomposition of graph serializations to more systematically study and understand the impact of the above aspects.

3.2. Decomposition of Serializations

Besides the labeling, we propose factoring serialization into: a *data structure*, an *internal edge-ordering rule*, and a *surface encoding* (see fig. 1). We detail these components next.

Node labeling determines how the nodes of a given graph are indexed. As mentioned before, any labeling is arbitrary, and a main desideratum for a graph reasoner is to be invariant to these (for graph-level targets). In principle, node labeling invariance would follow automatically once a target algorithm is correctly implemented. However, in practice, deviations may occur due to partial competence. On the other hand, as capacity and data grow, variability across relabelings may already shrink, even before achieving exact algorithmic execution. Either way, we crucially note that, absent potential positional spillage in $(\ell, \mathcal{S})$ and strong positional biases, node relabelings do not change the computational workload presented to the model, in terms of its algorithmic execution. In principle, under these conditions, no systematic trends across ℓ are thus to be expected.

Computational structure gathers the **data structure** and **internal edge ordering** rule. The former determines how the relational structure is described, the latter how this is presented as a sequence. Prominent choices for the data structure would be `edge-list`, `adjacency-list` or `adjacency-matrix` (see § C). Each structure comes with its own symmetries, e.g., any edge ordering in an `edge-list` equivalently represents the same graph, or both node and neighbor indices can be arbitrarily permuted in an `adjacency-list`. An ordering breaks these symmetries by presenting the data structure in an LLM-compatible sequential form. Note that the chosen computational structure may significantly impact the model's algorithmic execution by determining its computational path: e.g., with a *sorted* adjacency list, neighborhoods are pre-

sented contiguously in localized blocks. Many tasks such as BFS-like traversals or degree statistics are solvable by a single linear pass over them. On the contrary, with a *shuffled* edge list, a correct procedure must traverse the entire edge list for each neighborhood lookup or, alternatively, first build a canonical form to supporting linear scans, bringing about additional costs. Hence, unlike node relabelings, one would expect that changes in the computational structural can induce *persistent* performance and compute gaps.

Syntax captures variations in textual style, independently from the graph's structural representation. An ideal reasoner abstracts away from such choices; real models may exhibit biases from pretraining, e.g., bias on code execution may lead to better performance when the input is presented as code. Tokenization length or granularity may also induce side effects. These are *textual* rather than *computational* biases but can still affect accuracy and efficiency. Examples of surface encodings include edge list in plain text, in `json` format, or as Python code in `NetworkX` or `PyG`.

LLM graph reasoners may be sensitive to variations in each of the above encoding components. In the following section, we empirically analyze and discuss this relevant aspect.

4. Experimental Analyses

Models. In our experimental analysis, we evaluated a diverse set of LLMs with varying architectures, sizes, and training paradigms to thoroughly investigate graph reasoning capabilities. Our model selection includes the following:

- `Qwen` **(3B, 7B)** are open-weight models developed by Alibaba Cloud[1], which serve as our non-fine-tuned baseline. These models have been pre-trained on a large corpus of text data but have not been specifically optimized for graph reasoning tasks.
- `G1` **(3B, 7B)** are specialized graph reasoning models obtained by fine-tuning the `Qwen` base models using a two-stage approach (Guo et al., 2025). First, supervised fine-tuning was performed on the `Erdős` dataset, followed by reinforcement learning using Group Relative Policy Optimization. These models represent the current state of the art in LLM-based graph reasoning.
- `gpt-oss` **(20B, 120B)** are open-weight models (OpenAI, 2025) with significantly larger parameter counts. These models provide a contemporary perspective on scaling effects in capabilities without dedicated post-training for graph reasoning, serving as a comparison point to the smaller specialized models.

This selection allows us to analyze the impact of model scale (3B to 120B parameters) and specialization on graph reasoning performance and robustness to representational variations. For a deterministic comparison in our sensitivity experiments, we set the inference temperature of all `Qwen`, `G1` and `gpt-oss` models to 0, ensuring identical outputs for identical inputs. Accuracy differences compared to evaluations at 0.06 (as in (Guo et al., 2025)) are in § D.

Datasets. We run comprehensive experiments on the `Erdős` test set (Guo et al., 2025), encompassing 49 topological tasks over 100 graphs each, totaling 4,900 samples[2]. Across all ablations, shufflings, and models, this amounts to analyzing **over 1.5M individual model inferences**. As in `Erdős`, we group tasks by difficulty (Easy, Medium, Hard, Challenging). We further introduce a novel benchmark of 12 spectral tasks on 100 graphs from the `Erdős` test set (see § 5).

4.1. Sensitivity to Node Labeling

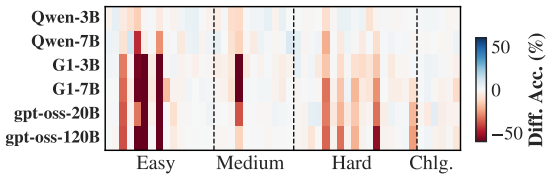


Figure 2. **Acc. diff.** w/ relabeled nodes vs. baseline `Erdős`.

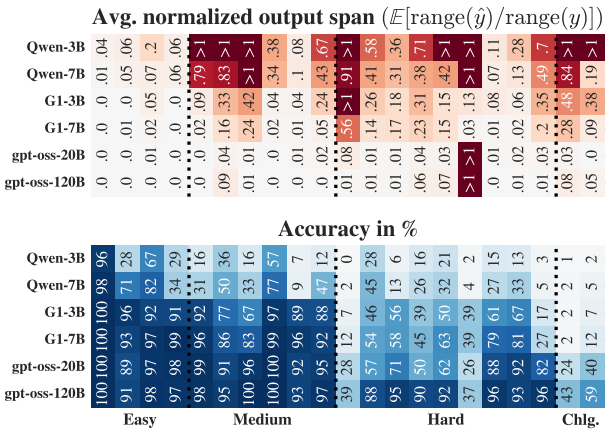


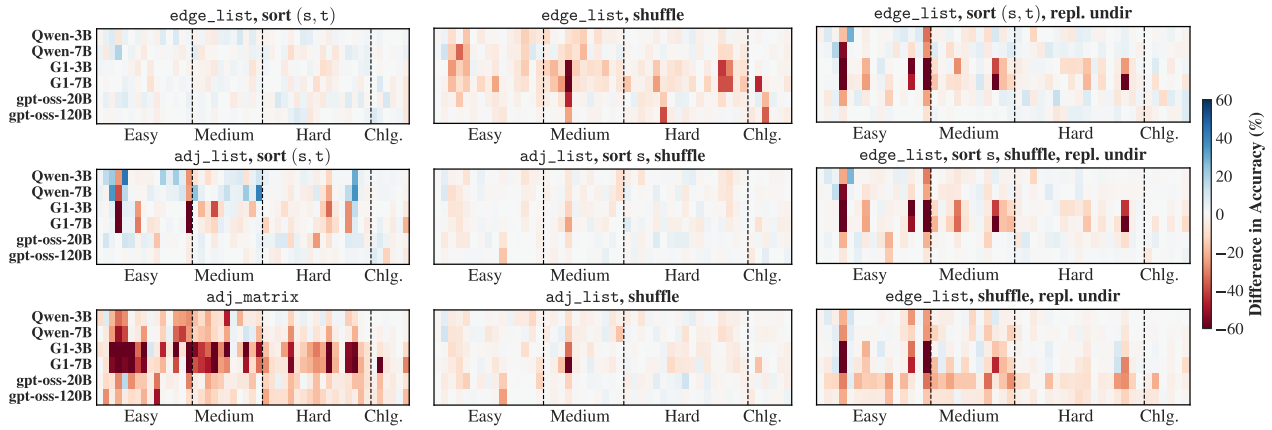
Figure 3. Average **per-example output spreads** normalized by per-task answer range (top), resp. accuracy (bottom).

We quantify the effect of node permutations by evaluating LLMs across multiple random relabelings of the same graph. Specifically, we draw $N = 10$ random node labelings and apply them on the `Erdős` test graphs[3], and then assess: **(a)**

[1] https://huggingface.co/collections/Qwen/qwen25-66e81a666513e518adb90d9e

[2] From the original suite, we only exclude `isomorphic_mapping` due to a different input structure.

[3] After relabeling, we lexicographically sort edges to factor out the impact of shuffled edge-list (Yuan et al., 2025).



(a) Different **sorted** comp. structures vs. baseline `Erdős`.

(b) **Shuffled** comp. structures vs. baseline resp. sorted.

(c) W/ **replicating undir. edges** vs. baseline w/o.

Figure 4. Ablation of *(a)* structure, *(b)* reordering, and *(c)* replicating undirected edges. Each colored cell corresponds to one out of the 49 `Erdős` problems we analyze (columns), evaluated on one out of the 6 models considered (rows), and, thus, aggregates the results of 100 (resp. 1000, as 10 shuffles per individual prompt) individual model inferences for deterministic (resp. shuffled) comp. structures. For full heatmaps with individual task labels, see § E.1.

the **mean accuracy** over these relabelings; **(b)** the **variability of the models' outputs** on *numerical* tasks.

As for **(a)**, fig. 2 reports the accuracy difference between the original `Erdős` dataset and sampled node relabelings across tasks. We observe clear sensitivity to the relabeling, which is more pronounced for `G1` than for `Qwen`. This can often be explained by reliance on positional regularities: certain labelings make the target solution easier to "heuristically guess". For example, in `bipartite_maximum_matching`, the correct solution in `Erdős` always induces a *contiguous* bipartition, i.e., $(\{1, \ldots, x\}, \{x+1, \ldots, n\})$. Fine-tuning can internalize this, but after relabeling the ground-truth bipartition ceases to be contiguous and performance drops.

As for **(b)**, we capture output variability induced by node relabelings by calculating the normalized output span averaged over all test graphs in *numerical* tasks (0.0 indicates no variability, as desired for invariance). These values are reported in fig. 3 (top). We observe that the `Qwen` models generally exhibit higher sensitivity, and that this typically correlates with task performance (bottom). These results indicate that graph-oriented fine-tuning can indeed make LLM graph reasoners more "invariant" to node permutations, but this is mainly explained by improved reasoning performance rather than the general emergence of functional invariance. Some exceptions are found, however, in the "Challenging" category, where fine-tuning lowers output variability *even without* producing a meaningful accuracy increase.

4.2. Sensitivity to Computational Structure

Here, we study robustness w.r.t. the data structure and its inner ordering. We consider three standard structures: `edge-list`, `adjacency-list`, and `adjacency-matrix`. As pointed out in § 3.2, each of these structures come with their *own* symmetries. Further, for undirected graphs, edges are sets of size 2, adding the symmetry that the endpoint order is irrelevant; we study this effect as well. Next we discuss general trends, for full numerical results see fig. 10, cf. § E.1.

We first evaluate the impact of the comp. structure itself, presented, when applicable, in an canonicalized form obtained by full lexicographic sort. In fig. 4a, we plot accuracy differences for *fully sorted* `edge-list`, `adjacency-list`, and `adjacency-matrix` encodings relative to the default `Erdős`[4]. We observe sorted `edge-lists` stay close to the baseline; `adjacency-lists` introduce shifts on some tasks. The largest drops occur on `edge_number` and `density`, both edge-counting tasks that naturally favor edge lists. The fine-tuned `G1` models are the most brittle to these changes in the data structure, `Qwen` sometimes even improves performance, and `gpt-oss` remains fairly steady. As for `adjacency-matrix` encodings, they often incur sizable losses. We hypothesize this is due to these representations being "misaligned" with counting tasks and inflating the description length to $\mathcal{O}(n^2)$, adding potentially distracting tokens.

[4]The `Erdős` dataset uses an edge-list representation where edges are ordered by a BFS from node 1 over a randomly sampled subgraph.

We now probe robustness w.r.t. the *inherent symmetries* of the `edge-list` and `adjacency-list` encodings. In fig. 4b, we report accuracy differences obtained by a (partly) *shuffled* data structure as opposed to its canonicalized form. The fine-tuned `G1` models are observed, again, to be the most brittle on average, with the largest drops occurring on `edge-list`. In sorted `edge-list`, the presented edges are grouped by source node, so each node's neighbors appear contiguously. We hypothesize this *locality*, i.e., neighbors appearing in "blocks", facilitates retrieval to the model. Random `edge-list` shuffles break this locality: retrieving a node's neighborhood requires a scan of the entire list, or a first internal reordering to allow linear scans, and both these operations can both introduce errors. As for `adjacency-lists`, the aforementioned locality remains preserved under all of its symmetries, and consistently, we instead observe shuffles hit comparatively mildly.

Last, we examine replicating undirected edges in `edge-lists` enforcing *full symmetric locality*: when edges are not replicated in a sorted listing, not all neighborhoods need to be continuous. Accuracy differences are reported in fig. 4c; again `G1`'s are most brittle and edge-counting tasks are hit hardest. This suggests, while replication resolves endpoint symmetries and aligns with edge extraction, it also inflates the sequence and can dilute signal.

4.3. Sensitivity to Syntax

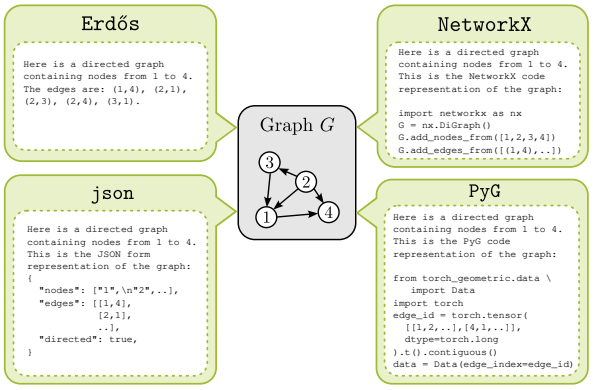


Figure 5. Examples of textual graph encodings.

We now investigate robustness under varying *syntax*: we fix the data structure to `edge-list` and re-encode the `Erdős` test graphs by each of the following encodings: `json`, `NetworkX`, and `PyG`. `json` is a common text-based serialization for storage and data exchange; `NetworkX` and `PyG` follow code-based conventions from graph processing libraries (cf. fig. 5). We choose these formats as widely represented in text and code corpora likely used for LLM pretraining, thereby potentially better activating the models' reasoning abilities.

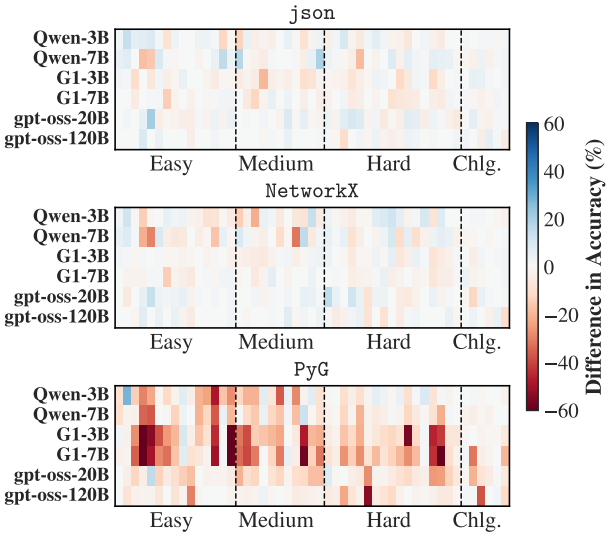


Figure 6. Different **syntax** vs. baseline `Erdős`.

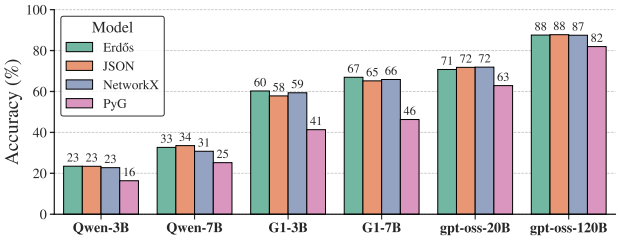


Figure 7. Average model accuracy by **syntax**.

We evaluate `G1`, `Qwen` and `gpt-oss` on the newly encoded graph instances and study the performance impact of these formats, also compared to the "standard" `Erdős` encoding. Results are all reported aggregated in fig. 7 and in full in fig. 11. As expected, `Erdős` remains, on average, the highest-performing encoding across tasks for both `G1`'s (-7B: best in $^{28}/_{49}$ tasks). As for the non-fine-tuned models, `Qwen`-7B shows, instead, a preference towards the `json` format (34% accuracy, best in $^{24}/_{49}$ tasks). Both `json` and `NetworkX` formats give the best performance on `gpt-oss`-20B, while the larger -120B variant exhibits robustness: barring `PyG`, all encodings work equally well (88% accuracy all). We interestingly remark the general underwhelming performance of `PyG`, which we comment on in § E.2.

We also measure the average per-task variability in terms of standard deviation over the performance obtained by the four considered encodings. Consistent with the above, `gpt-oss` models attain the lowest variability (5.41% for -20B, 3.45% for -120B), followed by `Qwen`'s (5.98% for -3B, 5.86% for -7B) and `G1`'s (9.69% for -3B, 10.51% for -7B). However, we note that the largest variability of `G1` is mostly attributed to `PyG`.

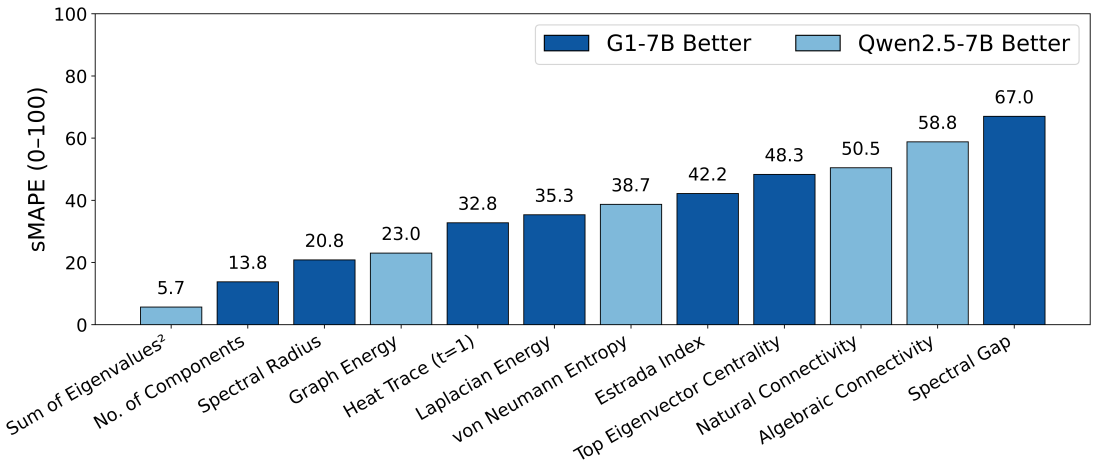


Figure 8. Performance is oscillating between the two models, there is no general consistent winner.

5. Testing Generalization on Spectral Tasks

A Spectral Graph Reasoning benchmark. Spectral quantities are fundamental tools to analyze robustness and diffusion in real-world networks (Jamakovic & Van Mieghem, 2008; Van Mieghem, 2023; Matzner et al., 2024; Ahuja et al., 2025), yet graph reasoning has almost exclusively focused on topological or combinatorial tasks (Tang et al., 2025; Yuan et al., 2025; Guo et al., 2025). Moreover, prior work has primarily emphasized quantitative evaluation; here, we complement it with a qualitative analysis of the actual generated reasoning. For this, we construct a novel suite of 12 spectral tasks.

Table 1. Spectral graph-theoretic tasks by difficulty.

Difficulty	Tasks
Easy	Graph Energy, Number of Connected Components, Sum of Squared Eigenvalues.
Medium	Algebraic Connectivity, Estrada Index, Laplacian Energy, Natural Connectivity, Spectral Gap, Spectral Radius.
Hard	Top Eigenvector Centrality, Heat Trace at $t = 1$, von Neumann Entropy.

We instantiate these tasks on a sample of 100 graphs from the `Erdős` test set and cluster them by difficulty (Easy, Medium, Hard), see table 1. On these real-valued tasks, we consider the well interpretable Symmetric Mean Absolute Percentage Error (sMAPE), as well as Relative Mean Absolute Error (RelMAE). For both, lower is better. sMAPE is bounded in $[0, 100]$; as for RelMAE, robust to near-zero values, a score below 1.0 indicates the model outperforms the mean baseline. Finally, we complement the evaluation with automated and manual reasoning-style classification of answers (*analytical*, *heuristic*, *incomplete*).

Table 2. Average performance on spectral tasks by difficulty ($\text{sMAPE}_{0\text{–}100}$, ↓).

Model	Easy	Medium	Hard
Qwen2.5-3B	35.01	53.89	56.60
Qwen2.5-7B	**19.73**	50.05	51.31
G1-3B	30.33	55.74	59.14
G1-7B	20.80	**49.58**	**39.95**

Results are reported aggregated in table 2, in full in § F.3. `G1`-7B yields modest gains compared to `Qwen`-7B on harder spectral problems (e.g., `eigenvector centrality`), yet it outperforms `Qwen` on only $7/12$ tasks, indicating no universal winner (fig. 8). While fine-tuning with `G1`-7B improves sMAPE performance on harder problems (e.g., `eigenvector_centrality`, `heat_trace`, fig. 13, cf. § F.3), on these tasks it remains not better than the mean baseline, as RelMAE is larger than 1.0 (cf. table 10, cf. § F.3). Overall, we observe `G1`-7B outperforms the mean baseline in 50% of the tasks, but the gains are still limited (RelMAE in $[0.6, 0.9]$). Manual inspection on 10% of the generated answers from `G1`-7B show an interesting pattern where it first approaches the task by analytical derivations (e.g., computation of degrees, normalized adj. matrices), but reverts to heuristics as complexity increases (e.g., known properties, assumptions of intermediate values, approximations based on structure).

6. Conclusion

We presented an evaluation of LLMs graph reasoners under symmetry transformations. Our framework decomposes serializations into *(i)* **node labeling**, *(ii)* **computational structure**, and *(iii)* **syntax**. We also test *(iv)* generalization on novel spectral tasks. Our experiments reveal the following key takeaways:

- In our analysis, **post-training can reduce sensitivity to node labeling**, although mostly reflecting performance enhancement rather than true invariance.
- The **serialization dictates computation paths** taken by the model, **altering algorithmic difficulty** and impacting performance.
- LLMs are not format-invariant, with specialized reasoners **excelling mostly on formats seen in post-training**.
- On spectral tasks, neither base nor specialized models generalize reliably.
- Overall, larger models appear more robust to the above variations, but are much costlier (cf. § B), and fine-tuning of smaller models does not consistently improve generalization or invariance, as sensitivity tends to be exacerbated by focus on datasets idiosyncrasies.

Based on these findings, we argue that progress in LLM graph reasoning should not be judged from fixed-format accuracy alone, and both post-training and evaluation should account for robustness across equivalent graph representations. An interesting future direction could be to design invariance-aware post-training and benchmarking pipelines for LLM graph reasoners.

Limitations

Our study focuses on LLM-based graph reasoners using textual serializations. They are comprehensive across node labeling, edge encoding, and syntax, and cover small and large scale models, as well an example of a (leading) fine-tuned reasoner. We could, however, cover a broader set of LLM models and other previously proposed tuning strategies. Additionally, we could perform similar analyses on real-world node- and graph-property prediction tasks.

Societal Impact

LLM graph reasoners have potential applications in knowledge discovery, network analysis, and scientific computation. Misinterpretation of outputs due to sensitivity to serializations could propagate errors in critical domain. In this sense, our research can educate the broader community of researchers and practitioners on the limits and brittleness of this general methodological approach, fostering the development of more reliable graph reasoners. On the other hand, we note that shedding light on this sensitivity could cue malicious actors into developing targeted exploits for adversarial attacks.

Acknowledgments

We would like to thank all people involved in the 2025 London Geometry and Machine Learning Summer School (LOGML), where this research project started. In particular, we would like to express our gratitude to the members of the organizing committee: Vincenzo Marco De Luca, Massimiliano Esposito, Simone Foti, Valentina Giunchiglia, Daniel Platt, Pragya Singh, Arne Wolf, and Zhengang Zhong. DH would like to thank Yifei Wang and Stefanie Jegelka for helpful discussions. FF is very grateful to Haggai Maron and Giorgos Bouritsas for general support and constructive early exchanges in project scoping. FF is also extremely thankful to the members of the "Eva Project", whose support he immensely appreciates. DH acknowledges funding by the Alexander von Humboldt Foundation, the Munich Center for Machine Learning (MCML), and the Munich Data Science Institute (MDSI) via the MDSI Doctoral Fellowship. FF conducted this work supported by an Aly Kaufman Post-Doctoral Fellowship.

References

Ahuja, A., Nallaperuma-Herzberg, S., Rafel, A., Wright, P., Lord, A., and Savory, S. J. Topology architect: Graph generative intelligence for scalable optical network topology design. In *2025 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 1–3. IEEE, 2025.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., and Hoefler, T. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24, pp. 17682–17690. AAAI Press, 2024.

Botchkarev, A. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*, 2021.

Chen, N., Li, Y., Tang, J., and Li, J. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2024)*, pp. 19, Barcelona, Spain, 2024. ACM. doi: 10.1145/3637528.3672010. URL https://doi.org/10.1145/3637528.3672010.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(1):11324–11436, 2023.

Chu, X., Xue, H., Tan, Z., Wang, B., Mo, T., and Li, W. Graphsos: Graph sampling and order selection to help llms understand graphs better, 2025. URL https://arxiv.org/abs/2501.14427.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016.

Dudzik, A. J. and Veličković, P. Graph neural networks are dynamic programmers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20635–20647, 2022.

Fatemi, B., Halcrow, J., and Perozzi, B. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.

Ge, Y., Liu, S., Bi, B., Wang, Y., Mei, L., Feng, W., Chen, L., and Cheng, X. Can graph descriptive order affect solving graph problems with llms? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, 2025.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, volume 2, pp. 729–734, 2005.

Guo, X., Li, A., Wang, Y., Jegelka, S., and Wang, Y. G1: Teaching llms to reason on graphs with reinforcement learning. *arXiv preprint arXiv:2505.18499*, 2025.

He, X., Tian, Y., Sun, Y., Chawla, N. V., Laurent, T., LeCun, Y., Bresson, X., and Hooi, B. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Hyndman, R. J. and Koehler, A. B. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

Jamakovic, A. and Van Mieghem, P. On the robustness of complex networks by using the algebraic connectivity. In *International conference on research in networking*, pp. 183–194. Springer, 2008.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations (ICLR) 2024*, 2024. URL https://arxiv.org/abs/2310.00149.

Luo, Z., Song, X., Huang, H., Lian, J., Zhang, C., Jiang, J., and Xie, X. Graphinstruct: Empowering large language models with graph understanding and reasoning capability. *arXiv preprint arXiv:2403.04483*, 2024.

Makridakis, S. Accuracy measures: theoretical and practical concerns. *International journal of forecasting*, 9(4):527–529, 1993.

Makridakis, S. and Hibon, M. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In *International Conference on Machine Learning*, pp. 4363–4371. PMLR, 2019.

Matzner, R., Ahuja, A., Sadeghi, R., Doherty, M., Beghelli, A., Savory, S. J., and Bayvel, P. Topology bench: systematic graph-based benchmarking for core optical networks.

Journal of Optical Communications and Networking, 17 (1):7–27, 2024.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI conference on artificial intelligence*, 33: 4602–4609, 2019.

OpenAI. Gpt-oss: Open-weight language models. https://openai.com/research/gpt-oss, 2025. Accessed: 2025-09-06.

Sanford, C., Fatemi, B., Hall, E., Tsitsulin, A., Kazemi, M., Halcrow, J., Perozzi, B., and Mirrokni, V. Understanding transformer reasoning capabilities via graph algorithms. *Advances in Neural Information Processing Systems*, 37: 78320–78370, 2024.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 1 2009. ISSN 1045-9227. doi: 10.1109/TNN.2008.2005141.

Tang, J., Zhang, Q., Li, Y., Chen, N., and Li, J. Grapharena: Evaluating and exploring large language models on graph computation. In *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*, 2025. URL https://arxiv.org/abs/2407.00379.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Van Mieghem, P. *Graph spectra for complex networks*. Cambridge university press, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, volume 37, 2023. doi: 10.48550/arXiv.2305.10037. URL https://arxiv.org/abs/2305.10037.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *The Seventh International Conference on Learning Representations*, 2019.

Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K.-i., and Jegelka, S. What can neural networks reason about? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rJxbJeHFPS.

Yang, H., Wang, X., Tao, Q., Hu, S., Lin, Z., and Zhang, M. Gl-fusion: Rethinking the combination of graph neural network and large language model. *arXiv preprint arXiv:2412.06849*, 2024. URL https://doi.org/10.48550/arXiv.2412.06849.

Ye, R., Zhang, C., Wang, R., Xu, S., and Zhang, Y. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1955–1973. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-eacl.132/.

Yuan, Z., Liu, M., Wang, H., and Qin, B. Gracore: Benchmarking graph comprehension and complex reasoning in large language models. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025)*, pp. 7925–7948. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.coling-main.531. URL https://aclanthology.org/2025.coling-main.531/.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the expressive power of GNNs via graph biconnectivity. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhang, B., Gai, J., Du, Y., Ye, Q., He, D., and Wang, L. Beyond weisfeilerlehman: A quantitative framework for gnn expressiveness. *The Twelfth International Conference on Learning Representations*, 2024.

Zhao, J., Zhuo, L., Shen, Y., Qu, M., Liu, K., Bronstein, M., Zhu, Z., and Tang, J. Graphtext: Graph reasoning in text space. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning (Workshop at NeurIPS 2024)*, 2024. URL https://arxiv.org/abs/2310.01089.

A. Compute Resources & Serving Stack

All models are served using `vLLM`, a high-throughput LLM serving system based on *Paged Attention* for efficient KV-cache management[5]. Unless otherwise specified, the evaluation was run on NVIDIA H100 80GB accelators with identical decoding and batching settings: temperature 0 (deterministic), fixed maximum output lengths and matched prompt budgets across models and tasks. For `gpt-oss`, we use the default `medium` reasoning effort. For `gpt-oss`-120, we additionally use a multi-GPU configuration (2×H100 80GB) with tensor parallelism.

B. Inference Runtime vs Accuracy Trade-offs

Across all graph reasoning tasks, `gpt-oss` models attain the highest accuracy and robustness, consistently outperforming `Qwen` (3B, 7B) and `G1` (3B, 7B). The gains come with increase in the latency per request when served via `vLLM` under matched scheduling and decoding policies. In table 3 report wall-clock latency relative to the 3B-7B class on a single H100 GPU.

Table 3. Inference performance comparison across model sizes and configurations. Results averaged over 100 prompts using single or dual H100 80GB GPUs.

Model	GPUs (H100 80GB)	Latency (s)	Throughput (tokens/s)
`Qwen`-3B	1	0.531	1,976.13
`Qwen`-7B	1	0.919	1,377.60
`G1`-3B	1	0.765	5,349.55
`G1`-7B	1	1.207	3,392.40
`gpt-oss`-20B	1	5.457	2,013.14
`gpt-oss`-120B	1	31.740	361.14
`gpt-oss`-120B	2	7.103	1,459.86

These factors isolate the impact of model size and computational reasoning on response time by keeping the serving infrastructure (`vLLM` scheduler, KV-cache paging) and decoding settings constant. In practice, the decision involves balancing accuracy against latency: `gpt-oss` delivers state-of-the-art accuracy but requires longer inference time. While multi-GPU deployment helps offset some of the slowdown, it introduces additional costs.

C. Example Prompts

This section illustrates, at the prompt level, our encodings under node relabelings, the computational structures, and the surface encodings. As a running example throughout this section, consider the following `shortest_path` prompt. For all list-style encodings, the right side includes a heatmap of node indices: 1 appears lightest and 19 darkest.

Example prompt `shortest_path`, `Erdős` encoding

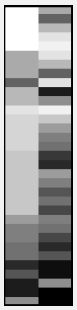
```
The task is to determine the shortest path between two nodes.

The input nodes are guaranteed to be connected.

Here is an undirected graph containing nodes from 1 to 19. The edges are: (1, 7), (1, 12), (1,
6), (1, 3), (1, 2), (7, 3), (7, 6), (7, 12), (12, 3), (6, 17), (6, 9), (3, 2), (4, 5), (4, 8),
(4, 10), (4, 11), (5, 15), (5, 16), (5, 8), (5, 10), (5, 13), (5, 11), (5, 14), (8, 10), (10, 11),
(10, 14), (11, 16), (11, 13), (16, 18), (13, 18), (17, 9), (17, 19), (9, 19).

Question: What is the shortest path between node 12 and node 19?

You need to format your answer as a list of nodes, e.g., [node-1, node-2, ..., node-n].
```



[5] https://github.com/vllm-project/vllm

C.1. Node Relabeling (ℓ)

Random node relabeling

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are: (1, 6), (1, 10), (1, 16), (2, 7), (2, 18), (3, 14), (3, 19), (4, 18), (5, 7), (5, 13), (5, 18), (6, 8), (6, 10), (6, 16), (7, 13), (7, 15), (7, 18), (8, 10), (9, 11), (9, 12), (9, 17), (10, 16), (10, 17), (11, 12), (11, 17), (13, 15), (13, 18), (14, 15), (14, 18), (15, 18), (15, 19), (16, 17), (18, 19).**

...

C.2. Computational Structure

`edge_list`, sorting key = `(source,target)`

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are: (1, 2), (1, 3), (1, 6), (1, 7), (1, 12), (2, 3), (3, 7), (3, 12), (4, 5), (4, 8), (4, 10), (4, 11), (5, 8), (5, 10), (5, 11), (5, 13), (5, 14), (5, 15), (5, 16), (6, 7), (6, 9), (6, 17), (7, 12), (8, 10), (9, 17), (9, 19), (10, 11), (10, 14), (11, 13), (11, 16), (13, 18), (16, 18), (17, 19).**

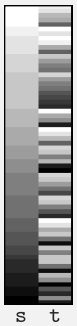
...

`edge_list`, sorting key = `(source,target)`, replicate undirected edges

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are (each undirected edge is listed in both directions): (1, 2), (1, 3), (1, 6), (1, 7), (1, 12), (2, 1), (2, 3), (3, 1), (3, 2), (3, 7), (3, 12), (4, 5), (4, 8), (4, 10), (4, 11), (5, 4), (5, 8), (5, 10), (5, 11), (5, 13), (5, 14), (5, 15), (5, 16), (6, 1), (6, 7), (6, 9), (6, 17), (7, 1), (7, 3), (7, 6), (7, 12), (8, 4), (8, 5), (8, 10), (9, 6), (9, 17), (9, 19), (10, 4), (10, 5), (10, 8), (10, 11), (10, 14), (11, 4), (11, 5), (11, 10), (11, 13), (11, 16), (12, 1), (12, 3), (12, 7), (13, 5), (13, 11), (13, 18), (14, 5), (14, 10), (15, 5), (16, 5), (16, 11), (16, 18), (17, 6), (17, 9), (17, 19), (18, 13), (18, 16), (19, 9), (19, 17).**

...

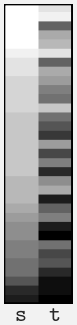


`edge_list`, sorting key = `source`, shuffle `target`

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are: (1, 3), (1, 2), (1, 12), (1, 7), (1, 6), (2, 3), (3, 12), (3, 7), (4, 8), (4, 10), (4, 11), (4, 5), (5, 11), (5, 13), (5, 15), (5, 8), (5, 14), (5, 10), (5, 16), (6, 9), (6, 7), (6, 17), (7, 12), (8, 10), (9, 17), (9, 19), (10, 11), (10, 14), (11, 13), (11, 16), (13, 18), (16, 18), (17, 19).**

...



`edge_list`, sorting key `source`, shuffle `target`, replicate undirected edges

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are (each undirected edge is listed in both directions): (1, 3), (1, 2), (1, 12), (1, 7), (1, 6), (2, 3), (2, 1), (3, 1), (3, 7), (3, 12), (3, 2), (4, 10), (4, 5), (4, 11), (4, 8), (5, 14), (5, 4), (5, 11), (5, 16), (5, 8), (5, 10), (5, 15), (5, 13), (6, 7), (6, 1), (6, 17), (6, 9), (7, 1), (7, 12), (7, 6), (7, 3), (8, 10), (8, 5), (8, 4), (9, 6), (9, 19), (9, 17), (10, 5), (10, 4), (10, 11), (10, 8), (10, 14), (11, 10), (11, 4), (11, 5), (11, 16), (11, 13), (12, 3), (12, 1), (12, 7), (13, 5), (13, 11), (13, 18), (14, 5), (14, 10), (15, 5), (16, 18), (16, 5), (16, 11), (17, 9), (17, 6), (17, 19), (18, 13), (18, 16), (19, 17), (19, 9).**

...

`edge_list`, shuffle entire list

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are: (4, 11), (1, 2), (9, 6), (11, 10), (14, 5), (11, 5), (11, 16), (15, 5), (4, 10), (5, 10), (5, 16), (14, 10), (17, 19), (17, 6), (8, 5), (5, 13), (9, 19), (16, 18), (8, 4), (1, 12), (9, 17), (11, 13), (4, 5), (7, 12), (7, 1), (1, 3), (6, 1), (6, 7), (3, 2), (12, 3), (13, 18), (3, 7), (8, 10).**

...

`edge_list`, shuffle entire list, replicate undirected edges

...

Here is an undirected graph containing nodes from 1 to 19. **The edges are (each undirected edge is listed in both directions): (8, 5), (7, 1), (6, 17), (16, 11), (16, 18), (11, 5), (3, 7), (6, 17), (3, 7), (9, 19), (6, 1), (10, 5), (10, 4), (11, 16), (10, 14), (13, 18), (13, 18), (17, 9), (2, 3), (3, 1), (4, 10), (7, 6), (19, 17), (5, 10), (11, 4), (12, 7), (5, 4), (1, 7), (6, 1), (5, 15), (7, 12), (16, 5), (12, 3), (1, 12), (17, 19), (8, 10), (3, 2), (5, 8), (11, 5), (9, 17), (3, 12), (10, 14), (6, 9), (5, 4), (5, 13), (1, 2), (16, 5), (11, 10), (14, 5), (13, 11), (1, 3), (5, 15), (18, 16), (11, 10), (13, 5), (11, 4), (12, 1), (14, 5), (6, 7), (1, 2), (11, 13), (4, 8), (8, 10), (19, 9), (6, 9), (4, 8).**

...

`adj_list`, sorting key = `(source,target)`

...

Here is an undirected graph containing nodes from 1 to 19. **The adjacency list is:**

- **node 1 is connected to (2, 3, 6, 7, 12),**
- **node 2 is connected to (1, 3),**
- **node 3 is connected to (1, 2, 7, 12),**
- **node 4 is connected to (5, 8, 10, 11),**
- **node 5 is connected to (4, 8, 10, 11, 13, 14, 15, 16), ...**

...

`adj_list`, sorting key = `source`, shuffle `target`

...

Here is an undirected graph containing nodes from 1 to 19. **The adjacency list is:**

- **node 1 is connected to (12, 6, 7, 3, 2),**
- **node 2 is connected to (1, 3),**
- **node 3 is connected to (7, 1, 2, 12),**
- **node 4 is connected to (5, 11, 8, 10),**
- **node 5 is connected to (8, 15, 13, 10, 14, 16, 11, 4), ...**

...

adj_list, sorting key = target, shuffle source

```
...
Here is an undirected graph containing nodes from 1 to 19.  The adjacency list is:

• node 4 is connected to (5, 8, 10, 11),
• node 13 is connected to (5, 11, 18),
• node 18 is connected to (13, 16),
• node 10 is connected to (4, 5, 8, 11, 14),
• node 19 is connected to (9, 17), ...

...
```

adj_list, shuffle all

```
...
Here is an undirected graph containing nodes from 1 to 19.  The adjacency list is:

• node 9 is connected to (17, 19, 6),
• node 16 is connected to (18, 11, 5),
• node 18 is connected to (16, 13),
• node 8 is connected to (4, 5, 10),
• node 7 is connected to (6, 1, 3, 12), ...

...
```

adj_matrix

```
...
Here is an undirected graph containing nodes from 1 to 19.  This is the binary adjacency matrix
representation of the graph where 1 denotes an edge between nodes:

[[0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
 [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0],
...]

...
```

C.3. Surface Encoding

json

```
...
Here is a undirected graph containing nodes from 1 to 19.  This is the JSON form representation of the
graph:
{
"nodes":  [ "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17",
"18", "19" ],
"edges":  [ [ 1, 7 ], [ 1, 12 ], [ 1, 6 ], [ 1, 3 ], [ 1, 2 ], [ 7, 3 ], [ 7, 6 ], [ 7, 12 ], [ 12, 3 ], [ 6,
17 ], [ 6, 9 ], [ 3, 2 ], [ 4, 5 ], [ 4, 8 ], [ 4, 10 ], [ 4, 11 ], [ 5, 15 ], [ 5, 16 ], [ 5, 8 ], [ 5, 10
], [ 5, 13 ], [ 5, 11 ], [ 5, 14 ], [ 8, 10 ], [ 10, 11 ], [ 10, 14 ], [ 11, 16 ], [ 11, 13 ], [ 16, 18 ], [
13, 18 ], [ 17, 9 ], [ 17, 19 ], [ 9, 19 ] ],
"directed":  false
}

...
```

NetworkX

```
...

Here is an undirected graph containing nodes from 1 to 19.  This is the NetworkX code representation of the graph:
import networkx as nx
G = nx.Graph()
G.add_nodes_from([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
G.add_edges_from([(1, 7), (1, 12), (1, 6), (1, 3), (1, 2), (7, 3), (7, 6), (7, 12), (12, 3), (6, 17), (6, 9),
(3, 2), (4, 5), (4, 8), (4, 10), (4, 11), (5, 15), (5, 16), (5, 8), (5, 10), (5, 13), (5, 11), (5, 14), (8,
10), (10, 11), (10, 14), (11, 16), (11, 13), (16, 18), (13, 18), (17, 9), (17, 19), (9, 19)])

...
```

PyG

```
...

Here is a undirected graph containing nodes from 1 to 19.  This is the PyG code representation of the graph:
from torch_geometric.data import Data
import torch
edge_index = torch.tensor([[1, 7, 1, 12, 1, 6, 1, 3, 1, 2, 7, 3, 7, 6, 7, 12, 12, 3, 6, 17, 6, 9, 3, 2, 4, 5,
4, 8, 4, 10, 4, 11, 5, 15, 5, 16, 5, 8, 5, 10, 5, 13, 5, 11, 5, 14, 8, 10, 10, 11, 10, 14, 11, 16, 11, 13,
16, 18, 13, 18, 17, 9, 17, 19, 9, 19], [7, 1, 12, 1, 6, 1, 3, 1, 2, 1, 3, 7, 6, 7, 12, 7, 3, 12, 17, 6, 9, 6,
2, 3, 5, 4, 8, 4, 10, 4, 11, 4, 15, 5, 16, 5, 8, 5, 10, 5, 13, 5, 11, 5, 14, 5, 10, 8, 11, 10, 14, 10, 16,
11, 13, 11, 18, 16, 18, 13, 9, 17, 19, 17, 19, 9]], dtype=torch.long).t().contiguous()
data = Data(edge_index=edge_index)

...
```

D. Temperature Setting

To ensure a fair and deterministic comparison across our experiments, we set the inference temperature of both the `G1` and `Qwen` models to 0. A temperature of 0 eliminates randomness in the models' outputs, guaranteeing that identical inputs always yield identical responses.

Fig. 9 illustrates the accuracy differences between evaluations performed at temperature 0.06, as reported by Guo et al. (2025), and our deterministic evaluations at temperature 0. To maintain consistency, we exclude the task `isomorphic_mapping`, following our main analysis, and the tasks `weighted_minimum_spanning_tree` and `weighted_shortest_path`, as results for them are not reported in Guo et al. (2025). Overall, we observe only marginal differences in accuracy between the two temperature settings. Specifically, `G1`-3B achieves 62.53 % at temperature 0 compared to 63.21 % at 0.06, while `G1`-7B reaches 69.13 % and 69.36 %, respectively. For `Qwen`, the 3B model scores 24.34 % at temperature 0 and 24.09 % at 0.06, whereas the 7B models both achieve 33.96 %. Although these overall differences are small, we note that some individual tasks exhibit larger variations, for example `dominating_set` and `common_neighbor` in `Qwen`-3B, and `connected_component_number` in both `Qwen`-3B and `Qwen`-7B, where discrepancies of up to 15 % occur.

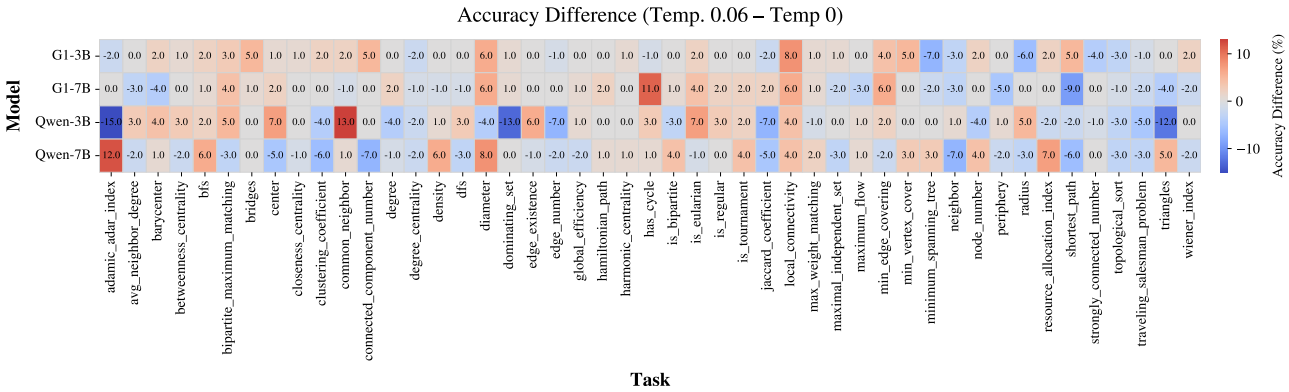


Figure 9. Accuracy difference between `G1` and `Qwen` evaluated at temperature 0.06 (as reported by (Guo et al., 2025)) and the deterministic version at temperature 0 used in our analysis.

E. Additional Results

E.1. Edge Encoding

We report per-task accuracies for the 3B models in table 4, for the 7B models in table 5, and for the `gpt-oss` models in table 6. Further, we visualize the per-task results in fig. 10.

Table 4. Edge reordering ablation results for the 3B models. We report the accuracy in %. `Erdős` baseline (w/ zero temperature) is shaded. For the computational structure, `edge` = edge list, `adjl` = adjacency list, `adjm` = adjacency matrix. For sorting, `s` = source and `t` = target, and the entry represents the sorting key. Shuffling is applied to all remaining ambiguities; in this case we report the mean accuracy ± std over 10 seeds.

		Qwen-3B												G1-3B											
	Comp. structure	edge	edge	edge	edge	edge	edge	edge	adjl	adjl	adjl	adjl	adjm	edge	edge	edge	edge	edge	edge	edge	adjl	adjl	adjl	adjl	adjm
	Sort (rest shuffled)	Erdős	(s,t)	(s,t)	s	s	✗	✗	(s,t)	s	t	✗	NA	Erdős	(s,t)	(s,t)	s	s	✗	✗	(s,t)	s	t	✗	NA
	Repl. undir	✗	✗	✓	✗	✓	✗	✓	(✓)	(✓)	(✓)	(✓)	(✓)	✗	✗	✓	✗	✓	✗	✓	(✓)	(✓)	(✓)	(✓)	(✓)
Easy	node_number	98.0	97.0	94.0	96.8 ± 2.3	97.9 ± 2.0	96.2 ± 1.3	94.7 ± 1.7	100.0	99.6 ± 0.5	99.1 ± 0.6	99.3 ± 0.5	87.0	98.0	100.0	100.0	99.9 ± 0.3	99.7 ± 0.5	100.0 ± 0.0	99.8 ± 0.4	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	90.0
	dominating_set	36.0	41.0	50.0	41.3 ± 4.2	50.0 ± 6.1	44.3 ± 4.0	52.2 ± 5.1	51.0	56.9 ± 5.8	52.4 ± 4.3	57.3 ± 5.1	44.0	98.0	99.0	99.0	99.2 ± 0.8	98.0 ± 1.1	99.3 ± 0.7	98.8 ± 1.1	97.0	98.3 ± 0.9	98.3 ± 0.8	96.4 ± 1.3	92.0
	common_neighbor	31.0	39.0	37.0	33.7 ± 2.3	37.2 ± 4.4	28.9 ± 2.5	24.8 ± 3.4	53.0	50.5 ± 4.6	61.4 ± 3.0	48.2 ± 4.6	11.0	89.0	90.0	94.0	87.1 ± 1.9	85.5 ± 2.7	64.0 ± 2.0	67.8 ± 3.4	95.0	87.2 ± 1.8	91.8 ± 3.0	84.7 ± 2.2	16.0
	edge_number	38.0	31.0	13.0	32.0 ± 2.9	8.8 ± 2.4	30.3 ± 2.8	9.0 ± 2.4	12.0	10.9 ± 1.8	8.7 ± 1.7	8.6 ± 3.1	4.0	97.0	97.0	13.0	96.9 ± 1.3	12.9 ± 0.3	82.7 ± 1.8	12.1 ± 1.2	11.0	3.9 ± 2.7	4.6 ± 1.5	4.0 ± 1.3	7.0
	neighbor	35.0	39.0	68.0	35.3 ± 2.1	62.8 ± 3.9	22.5 ± 3.9	33.3 ± 4.1	79.0	70.9 ± 3.1	87.1 ± 3.8	70.3 ± 4.8	7.0	94.0	95.0	94.0	92.2 ± 0.9	92.4 ± 2.5	74.8 ± 3.0	78.4 ± 2.5	100.0	96.1 ± 1.3	98.8 ± 1.2	95.6 ± 2.0	4.0
	bfs	1.0	1.0	1.0	3.5 ± 1.8	2.3 ± 0.8	1.2 ± 0.6	1.2 ± 1.2	0.0	0.6 ± 0.7	0.5 ± 0.7	0.7 ± 0.7	0.0	93.0	91.0	90.0	93.7 ± 1.2	90.6 ± 1.7	94.3 ± 1.8	93.1 ± 2.6	98.0	97.2 ± 1.5	97.6 ± 1.2	97.8 ± 1.5	7.0
	has_cycle	48.0	51.0	47.0	46.6 ± 4.3	46.1 ± 3.9	45.6 ± 3.5	45.6 ± 4.5	45.0	49.1 ± 3.9	41.6 ± 2.4	45.7 ± 4.1	39.0	90.0	85.0	64.0	86.2 ± 1.5	63.6 ± 0.7	82.8 ± 2.7	61.8 ± 1.3	64.0	63.8 ± 1.8	64.1 ± 1.5	62.9 ± 1.9	56.0
	dfs	6.0	3.0	3.0	5.4 ± 1.6	2.5 ± 1.4	2.1 ± 1.4	2.2 ± 1.8	4.0	3.9 ± 2.0	2.9 ± 1.5	2.5 ± 1.3	1.0	100.0	100.0	100.0	99.1 ± 0.6	98.9 ± 1.4	98.3 ± 0.8	98.0 ± 1.1	99.0	98.8 ± 0.9	97.6 ± 1.3	97.4 ± 1.7	26.0
	minimum_spanning_tree	8.0	9.0	8.0	7.9 ± 2.8	4.2 ± 1.8	7.7 ± 3.3	5.3 ± 2.3	6.0	5.8 ± 2.1	6.3 ± 1.9	5.1 ± 2.0	10.0	88.0	83.0	91.0	87.7 ± 2.3	90.9 ± 2.5	76.7 ± 4.1	84.7 ± 2.7	88.0	89.0 ± 2.3	90.1 ± 3.0	85.9 ± 2.7	98.0
	weighted_minimum_spanning_tree	4.0	1.0	0.0	1.4 ± 1.1	0.7 ± 0.8	1.7 ± 1.1	0.9 ± 1.0	4.0	1.5 ± 1.0	1.6 ± 1.1	1.3 ± 0.8	0.0	5.0	5.0	4.0	5.9 ± 2.2	3.0 ± 1.2	6.4 ± 1.6	5.3 ± 1.8	2.0	2.1 ± 1.2	3.8 ± 1.8	3.1 ± 1.6	0.0
	edge_existence	74.0	82.0	87.0	81.1 ± 2.9	85.4 ± 2.8	78.6 ± 3.9	78.8 ± 3.4	94.0	95.7 ± 0.9	97.8 ± 1.5	97.7 ± 0.9	57.0	100.0	100.0	100.0	99.7 ± 0.7	98.2 ± 0.8	97.4 ± 1.4	98.7 ± 0.8	98.0	98.2 ± 0.8	97.8 ± 0.9	97.9 ± 2.0	54.0
	is_regular	92.0	92.0	100.0	90.4 ± 1.2	94.3 ± 1.9	85.7 ± 3.2	79.3 ± 3.1	98.0	96.6 ± 1.8	93.7 ± 1.6	92.5 ± 2.5	91.0	100.0	100.0	100.0	99.1 ± 1.0	97.2 ± 1.1	92.7 ± 1.1	89.6 ± 3.0	100.0	99.9 ± 0.3	99.6 ± 0.7	99.8 ± 0.6	93.0
	degree	62.0	71.0	73.0	62.2 ± 4.3	58.4 ± 5.3	56.5 ± 3.6	30.1 ± 3.6	83.0	75.4 ± 2.9	85.6 ± 2.7	77.8 ± 3.6	38.0	95.0	91.0	32.0	91.7 ± 1.3	25.3 ± 2.9	75.9 ± 2.3	43.4 ± 4.8	95.0	88.0 ± 1.9	87.9 ± 2.1	85.1 ± 3.3	21.0
	is_tournament	73.0	72.0	80.0	73.6 ± 3.6	72.4 ± 2.5	63.5 ± 3.2	65.6 ± 3.3	80.0	76.8 ± 3.5	70.3 ± 6.1	67.8 ± 5.2	48.0	99.0	99.0	99.0	98.9 ± 0.3	98.7 ± 0.5	98.7 ± 0.7	98.3 ± 0.8	99.0	99.0 ± 0.0	98.0 ± 0.9	98.8 ± 0.4	91.0
	density	32.0	34.0	1.0	31.9 ± 1.3	1.3 ± 0.5	28.1 ± 2.3	2.5 ± 1.3	5.0	2.5 ± 1.8	4.3 ± 1.9	2.6 ± 1.3	4.0	92.0	90.0	12.0	90.8 ± 0.8	11.9 ± 0.6	80.4 ± 2.4	10.6 ± 1.8	8.0	6.6 ± 1.9	8.2 ± 2.0	7.4 ± 2.9	5.0
Medium	adamic_adar_index	21.0	18.0	15.0	13.6 ± 2.3	13.9 ± 2.3	6.0 ± 2.7	3.9 ± 1.4	12.0	9.7 ± 2.2	12.7 ± 3.1	10.2 ± 2.8	3.0	96.0	95.0	85.0	93.9 ± 2.0	81.9 ± 1.9	81.7 ± 1.6	80.7 ± 1.1	97.0	96.5 ± 1.3	97.8 ± 1.3	96.0 ± 1.2	40.0
	clustering_coefficient	35.0	37.0	35.0	36.7 ± 2.7	35.8 ± 3.2	35.2 ± 3.9	31.0 ± 4.0	33.0	33.4 ± 2.5	31.0 ± 2.7	29.4 ± 3.4	24.0	80.0	78.0	77.0	72.3 ± 1.8	74.2 ± 3.2	58.1 ± 3.4	63.9 ± 1.7	60.0	61.1 ± 3.0	65.8 ± 3.8	63.8 ± 1.7	37.0
	connected_component_number	27.0	23.0	26.0	21.8 ± 5.6	18.5 ± 2.3	22.9 ± 3.8	18.8 ± 2.5	17.0	16.7 ± 4.9	20.6 ± 3.1	17.6 ± 4.6	5.0	74.0	71.0	73.0	70.4 ± 2.8	64.1 ± 2.6	60.1 ± 2.0	57.0 ± 2.0	66.0	64.0 ± 2.1	65.8 ± 2.1	64.3 ± 2.8	39.0
	bipartite_maximum_matching	14.0	9.0	9.0	9.4 ± 2.5	2.6 ± 1.3	1.0 ± 1.1	0.9 ± 1.0	11.0	3.8 ± 1.9	3.0 ± 1.3	1.0 ± 0.9	3.0	79.0	73.0	45.0	69.9 ± 2.4	38.0 ± 2.6	15.2 ± 3.6	14.8 ± 3.3	40.0	26.2 ± 3.7	12.2 ± 1.5	6.4 ± 2.6	4.0
	local_connectivity	58.0	62.0	56.0	56.6 ± 2.2	52.9 ± 1.7	50.1 ± 2.8	52.0 ± 2.7	58.0	60.9 ± 2.4	60.9 ± 2.1	57.4 ± 1.8	49.0	82.0	85.0	80.0	79.1 ± 3.0	71.9 ± 3.2	70.0 ± 3.5	66.0 ± 3.4	71.0	73.7 ± 3.5	71.0 ± 3.3	67.7 ± 2.6	53.0
	jaccard_coefficient	55.0	60.0	52.0	52.2 ± 3.9	49.2 ± 6.4	45.9 ± 5.6	33.2 ± 4.6	61.0	60.5 ± 3.4	66.5 ± 3.2	56.5 ± 5.9	8.0	97.0	96.0	96.0	94.5 ± 1.8	96.0 ± 1.2	88.8 ± 1.6	89.3 ± 1.8	100.0	96.3 ± 1.6	98.4 ± 1.1	98.5 ± 0.8	36.0
	min_edge_covering	0.0	2.0	4.0	2.0 ± 0.7	2.9 ± 1.4	1.3 ± 0.9	2.4 ± 2.6	7.0	3.1 ± 1.2	5.1 ± 1.7	4.1 ± 1.5	4.0	47.0	50.0	53.0	47.4 ± 4.0	50.8 ± 2.6	39.2 ± 3.5	47.9 ± 3.0	51.0	47.9 ± 2.3	49.2 ± 3.4	43.5 ± 2.9	49.0
	is_eularian	74.0	84.0	74.0	77.4 ± 2.3	72.3 ± 3.7	73.8 ± 4.0	63.3 ± 4.0	83.0	83.9 ± 2.0	79.8 ± 2.1	82.0 ± 2.1	53.0	96.0	97.0	96.0	95.6 ± 1.3	90.1 ± 2.3	88.5 ± 2.0	84.5 ± 3.3	91.0	90.4 ± 0.8	89.5 ± 1.5	90.0 ± 1.5	78.0
	degree_centrality	9.0	7.0	17.0	7.8 ± 2.6	12.8 ± 2.0	3.7 ± 1.8	11.3 ± 2.7	0.0	0.0 ± 0.0	2.3 ± 0.7	3.6 ± 1.9	4.0	91.0	93.0	53.0	89.3 ± 1.8	47.7 ± 4.0	76.5 ± 3.6	59.2 ± 3.6	87.0	86.4 ± 1.6	91.3 ± 1.3	87.6 ± 2.5	21.0
	is_bipartite	42.0	49.0	44.0	46.8 ± 2.6	44.9 ± 4.3	44.8 ± 3.8	41.6 ± 3.3	46.0	45.5 ± 4.0	38.7 ± 2.5	39.3 ± 3.0	37.0	79.0	71.0	53.0	71.1 ± 2.1	53.4 ± 1.1	67.7 ± 3.8	53.9 ± 1.3	61.0	54.4 ± 2.0	60.4 ± 2.3	59.5 ± 4.1	59.0
	resource_allocation_index	12.0	10.0	12.0	11.5 ± 2.5	9.0 ± 1.7	7.2 ± 2.8	4.0 ± 1.8	24.0	13.9 ± 2.0	23.8 ± 3.0	21.1 ± 3.6	4.0	90.0	88.0	84.0	89.4 ± 1.0	77.4 ± 1.8	80.9 ± 1.4	79.9 ± 2.6	91.0	91.7 ± 1.8	92.6 ± 1.1	91.2 ± 1.0	38.0
Hard	max_weight_matching	4.0	2.0	6.0	3.4 ± 1.9	1.1 ± 1.0	2.6 ± 1.3	1.3 ± 1.1	8.0	3.6 ± 1.4	3.5 ± 1.5	3.6 ± 1.7	3.0	23.0	25.0	20.0	26.8 ± 4.0	22.8 ± 3.4	23.7 ± 2.8	20.9 ± 2.8	23.0	21.4 ± 3.2	18.9 ± 2.8	22.4 ± 2.7	5.0
	closeness_centrality	1.0	0.0	0.0	0.7 ± 0.7	0.2 ± 0.6	0.3 ± 0.7	0.4 ± 0.5	2.0	1.3 ± 1.2	1.5 ± 1.0	1.2 ± 1.2	1.0	8.0	8.0	6.0	6.7 ± 0.9	6.5 ± 1.8	6.0 ± 1.4	5.9 ± 1.7	7.0	6.9 ± 1.6	6.7 ± 1.4	5.4 ± 1.6	3.0
	traveling_salesman_problem	29.0	29.0	16.0	29.1 ± 2.5	16.1 ± 2.7	28.0 ± 2.7	22.9 ± 2.6	30.0	22.0 ± 5.2	26.6 ± 2.8	22.2 ± 2.7	32.0	44.0	44.0	46.0	44.7 ± 3.7	43.8 ± 2.4	37.6 ± 4.6	38.7 ± 3.6	42.0	41.8 ± 2.3	43.3 ± 1.9	42.1 ± 2.9	41.0
	strongly_connected_number	6.0	7.0	7.0	7.7 ± 2.2	8.9 ± 3.0	6.4 ± 2.6	4.8 ± 2.9	4.0	4.5 ± 1.6	6.4 ± 2.8	6.2 ± 1.4	8.0	62.0	57.0	59.0	57.0 ± 1.2	56.0 ± 2.5	55.1 ± 1.3	54.8 ± 1.8	52.0	53.0 ± 1.2	53.2 ± 1.0	52.9 ± 1.4	37.0
	shortest_path	21.0	18.0	18.0	18.3 ± 3.0	19.9 ± 2.6	14.2 ± 3.2	16.5 ± 3.2	18.0	17.6 ± 3.0	17.8 ± 3.0	16.9 ± 4.0	7.0	57.0	56.0	57.0	54.9 ± 2.9	55.8 ± 4.1	39.9 ± 2.5	45.5 ± 4.3	54.0	50.1 ± 2.9	56.1 ± 4.6	53.6 ± 2.8	8.0
	weighted_shortest_path	2.0	2.0	4.0	1.3 ± 0.8	2.5 ± 1.1	1.6 ± 0.7	2.0 ± 1.2	1.0	2.3 ± 0.8	2.2 ± 1.1	2.1 ± 1.1	0.0	8.0	9.0	9.0	6.6 ± 1.2	9.0 ± 1.6	5.9 ± 1.2	6.7 ± 1.4	5.0	7.4 ± 1.4	7.5 ± 1.5	7.6 ± 1.8	1.0
	center	1.0	8.0	4.0	4.7 ± 0.9	6.3 ± 2.1	4.2 ± 1.4	4.8 ± 2.0	5.0	6.1 ± 2.1	5.2 ± 2.3	4.6 ± 1.7	5.0	24.0	24.0	14.0	25.1 ± 1.5	15.5 ± 1.8	16.2 ± 2.1	15.1 ± 3.1	23.0	22.0 ± 2.8	19.5 ± 2.8	16.8 ± 2.2	9.0
	diameter	12.0	18.0	12.0	14.0 ± 2.6	13.2 ± 3.6	13.1 ± 3.7	13.1 ± 3.7	14.0	13.0 ± 2.5	11.9 ± 3.7	12.5 ± 3.8	11.0	40.0	43.0	32.0	40.4 ± 3.2	31.1 ± 4.0	32.5 ± 3.9	32.0 ± 4.3	37.0	37.5 ± 3.1	35.7 ± 2.8	33.0 ± 3.3	16.0
	barycenter	11.0	12.0	18.0	11.2 ± 1.3	15.9 ± 4.8	14.1 ± 3.0	13.0 ± 2.5	7.0	11.0 ± 2.6	10.2 ± 2.8	10.1 ± 1.8	7.0	37.0	36.0	26.0	33.5 ± 1.5	27.6 ± 3.4	30.8 ± 2.5	29.4 ± 3.4	38.0	33.6 ± 4.2	31.5 ± 5.8	32.5 ± 4.1	6.0
	radius	18.0	17.0	22.0	19.4 ± 5.1	19.8 ± 3.8	22.2 ± 3.3	19.4 ± 4.0	17.0	23.0 ± 3.7	21.2 ± 2.5	22.7 ± 3.3	24.0	62.0	52.0	35.0	53.1 ± 2.9	40.0 ± 3.0	49.0 ± 4.1	43.8 ± 2.6	47.0	47.2 ± 4.1	47.0 ± 2.4	46.9 ± 3.1	39.0
	topological_sort	17.0	13.0	11.0	12.0 ± 2.4	12.6 ± 2.5	10.1 ± 2.1	9.5 ± 1.8	4.0	3.0 ± 1.6	2.1 ± 0.9	1.5 ± 1.2	6.0	70.0	62.0	66.0	63.9 ± 3.7	64.6 ± 2.8	62.4 ± 2.8	63.5 ± 2.5	36.0	40.6 ± 3.1	32.0 ± 5.4	30.2 ± 4.2	9.0
	periphery	2.0	4.0	2.0	1.8 ± 0.9	3.3 ± 1.6	3.0 ± 1.2	3.1 ± 1.4	3.0	2.9 ± 1.7	2.6 ± 1.5	3.8 ± 1.8	4.0	22.0	18.0	10.0	18.4 ± 1.9	13.9 ± 2.1	13.3 ± 2.8	12.7 ± 1.7	16.0	15.9 ± 1.7	11.9 ± 1.8	10.9 ± 2.3	3.0
	betweenness_centrality	1.0	2.0	2.0	1.8 ± 1.1	1.3 ± 0.8	1.2 ± 1.0	0.9 ± 0.9	2.0	2.4 ± 1.3	2.1 ± 1.7	1.7 ± 1.1	3.0	38.0	39.0	39.0	38.8 ± 0.6	38.0 ± 0.7	38.7 ± 0.5	37.7 ± 1.1	39.0	38.3 ± 0.7	38.2 ± 0.6	38.4 ± 0.5	37.0
	triangles	16.0	16.0	17.0	11.0 ± 2.2	10.7 ± 2.9	9.2 ± 2.4	8.9 ± 2.9	17.0	11.1 ± 3.3	13.0 ± 2.5	10.5 ± 2.5	1.0	67.0	68.0	64.0	50.8 ± 3.6	51.1 ± 3.7	25.7 ± 3.6	35.0 ± 2.4	39.0	38.1 ± 3.8	41.9 ± 4.0	38.7 ± 3.9	13.0
	avg_neighbor_degree	14.0	13.0	17.0	12.6 ± 2.7	12.2 ± 3.9	14.8 ± 3.0	10.7 ± 1.6	44.0	34.7 ± 4.9	48.6 ± 4.0	40.5 ± 2.6	5.0	68.0	68.0	42.0	65.9 ± 2.6	22.6 ± 2.5	38.4 ± 3.3	27.9 ± 3.4	83.0	74.7 ± 3.1	80.8 ± 2.5	78.5 ± 3.4	10.0
	harmonic_centrality	3.0	4.0	3.0	3.0 ± 0.9	2.3 ± 1.6	1.7 ± 1.3	1.8 ± 0.6	7.0	3.7 ± 1.3	3.2 ± 1.8	3.2 ± 1.1	2.0	18.0	21.0	15.0	17.3 ± 1.3	14.1 ± 1.4	14.0 ± 1.9	13.2 ± 2.5	17.0	14.0 ± 2.6	13.6 ± 3.0	11.5 ± 3.0	2.0
	bridges	0.0	2.0	2.0	0.9 ± 0.7	0.4 ± 0.5	0.1 ± 0.3	0.2 ± 0.4	1.0	0.5 ± 0.7	0.3 ± 0.5	0.2 ± 0.4	0.0	11.0	15.0	16.0	13.0 ± 1.5	14.6 ± 0.7	5.9 ± 1.7	4.4 ± 1.3	10.0	11.1 ± 2.0	8.8 ± 1.9	9.5 ± 1.4	2.0
Challenging	global_efficiency	0.0	1.0	1.0	0.7 ± 0.5	1.2 ± 0.6	0.7 ± 0.7	0.4 ± 0.5	1.0	1.0 ± 0.0	0.9 ± 0.3	1.2 ± 0.6	0.0	1.0	1.0	1.0	1.1 ± 0.3	1.0 ± 0.0	1.0 ± 0.5	1.0 ± 0.0	1.0	1.0 ± 0.0	1.1 ± 0.3	1.0 ± 0.0	0.0
	maximal_independent_set	1.0	3.0	2.0	2.6 ± 1.1	1.7 ± 1.4	1.5 ± 1.0	1.1 ± 1.0	3.0	2.8 ± 1.7	2.1 ± 1.1	2.0 ± 1.2	1.0	12.0	11.0	19.0	12.9 ± 2.5	12.8 ± 2.0	7.8 ± 2.7	10.0 ± 1.6	12.0	11.5 ± 1.8	11.8 ± 2.1	10.9 ± 2.5	4.0
	maximum_flow	1.0	1.0	2.0	2.2 ± 1.4	2.0 ± 1.3	1.8 ± 1.1	2.0 ± 1.4	2.0	2.4 ± 0.8	1.4 ± 1.6	1.5 ± 1.1	2.0	7.0	7.0	7.0	11.1 ± 1.9	11.4 ± 3.0	8.0 ± 1.1	8.3 ± 0.8	7.0	8.5 ± 1.3	8.6 ± 1.7	8.2 ± 1.3	3.0
	wiener_index	1.0	1.0	4.0	0.9 ± 1.0	1.4 ± 1.2	1.1 ± 0.9	1.3 ± 1.3	3.0	1.3 ± 0.7	2.4 ± 1.2	2.2 ± 1.9	1.0	6.0	6.0	1.0	5.0 ± 0.8	3.0 ± 0.8	6.2 ± 1.8	4.1 ± 1.0	3.0	3.6 ± 0.8	3.9 ± 1.4	2.9 ± 1.0	4.0
	hamiltonian_path	1.0	1.0	1.0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.3	0.0 ± 0.0	2.0	0.4 ± 0.5	0.9 ± 0.6	0.2 ± 0.4	1.0	2.0	3.0	3.0	1.6 ± 1.2	1.3 ± 1.2	1.1 ± 0.9	1.1 ± 1.1	2.0	0.5 ± 0.5	1.0 ± 1.2	0.4 ± 0.5	2.0
	min_vertex_cover	3.0	5.0	3.0	4.6 ± 1.3	3.0 ± 1.5	2.5 ± 1.6	2.9 ± 1.3	2.0	1.8 ± 1.0	2.0 ± 0.9	2.3 ± 0.9	1.0	16.0	13.0	15.0	14.4 ± 1.8	8.8 ± 2.9	11.0 ± 1.7	8.3 ± 2.6	10.0	7.3 ± 1.8	7.2 ± 1.6	6.6 ± 1.5	4.0

Table 5. Edge reordering ablation results for the 7B models. We report the accuracy in %. `Erdős` baseline (w/ zero temperature) is shaded. For the computational structure, `edge` = edge list, `adjl` = adjacency list, `adjm` = adjacency matrix. For sorting, `s` = source and `t` = target, and the entry represents the sorting key. Shuffling is applied to all remaining ambiguities; in this case we report the mean accuracy ± std over 10 seeds.

		Qwen-7B												G1-7B											
	Comp. structure	edge	edge	edge	edge	edge	edge	edge	adjl	adjl	adjl	adjl	adjm	edge	edge	edge	edge	edge	edge	edge	adjl	adjl	adjl	adjl	adjm
	Sort (rest shuffled)	Erdős	(s,t)	(s,t)	s	s	✗	✗	(s,t)	s	t	✗	NA	Erdős	(s,t)	(s,t)	s	s	✗	✗	(s,t)	s	t	✗	NA
	Repl. undir	✗	✗	✓	✗	✓	✗	✓	(✓)	(✓)	(✓)	(✓)	(✓)	✗	✗	✓	✗	✓	✗	✓	(✓)	(✓)	(✓)	(✓)	(✓)
Easy	node_number	95.0	98.0	99.0	97.6 ± 1.6	96.0 ± 1.8	99.2 ± 1.1	97.7 ± 1.2	99.0	99.4 ± 0.7	99.0 ± 1.2	97.9 ± 1.9	95.0	100.0	100.0	100.0	100.0 ± 0.0	100.0 ± 0.0	99.9 ± 0.3	98.0 ± 0.8	100.0	99.6 ± 0.5	99.5 ± 0.7	99.4 ± 0.7	100.0
	dominating_set	27.0	21.0	23.0	26.4 ± 2.7	27.6 ± 4.4	33.4 ± 4.6	35.2 ± 3.6	25.0	34.0 ± 5.1	37.5 ± 4.1	37.1 ± 6.4	21.0	98.0	97.0	98.0	98.4 ± 1.1	97.2 ± 1.3	94.2 ± 2.0	89.5 ± 2.1	99.0	98.1 ± 1.1	96.9 ± 1.7	97.0 ± 1.5	89.0
	common_neighbor	51.0	58.0	76.0	53.3 ± 3.0	68.1 ± 3.0	42.7 ± 2.7	48.9 ± 3.8	86.0	76.1 ± 3.8	89.2 ± 2.5	78.2 ± 2.3	25.0	94.0	94.0	95.0	92.0 ± 1.8	91.2 ± 2.0	75.5 ± 1.8	84.4 ± 3.6	97.0	93.3 ± 2.4	97.7 ± 0.9	91.6 ± 2.4	38.0
	edge_number	60.0	78.0	23.0	74.4 ± 3.4	18.1 ± 2.5	43.1 ± 4.6	15.1 ± 4.2	21.0	15.1 ± 1.7	15.1 ± 3.2	14.0 ± 1.9	8.0	97.0	97.0	19.0	96.4 ± 1.1	16.7 ± 1.6	87.5 ± 2.0	13.2 ± 1.9	25.0	17.4 ± 2.0	15.0 ± 2.1	14.3 ± 3.0	10.0
	neighbor	72.0	72.0	73.0	72.2 ± 2.0	75.0 ± 2.1	52.1 ± 4.1	59.0 ± 3.2	88.0	86.3 ± 3.2	90.3 ± 2.3	86.3 ± 3.1	32.0	96.0	100.0	95.0	98.5 ± 1.0	95.8 ± 1.5	84.7 ± 2.5	88.9 ± 2.1	99.0	96.5 ± 1.2	99.4 ± 0.7	96.2 ± 1.2	39.0
	bfs	6.0	9.0	9.0	11.4 ± 2.7	12.3 ± 2.9	6.8 ± 1.2	6.1 ± 1.7	7.0	7.5 ± 1.7	11.4 ± 2.9	9.4 ± 2.5	4.0	97.0	99.0	95.0	98.2 ± 1.0	96.4 ± 1.8	97.3 ± 1.4	91.1 ± 2.4	100.0	98.8 ± 1.2	99.4 ± 0.7	99.7 ± 0.5	34.0
	has_cycle	54.0	53.0	50.0	58.2 ± 3.5	50.3 ± 2.5	53.9 ± 4.6	52.5 ± 1.6	53.0	55.9 ± 3.7	55.0 ± 4.5	55.1 ± 3.9	48.0	87.0	86.0	61.0	85.1 ± 2.2	61.5 ± 1.6	73.5 ± 3.3	57.4 ± 1.9	56.0	58.1 ± 3.0	60.4 ± 1.6	59.7 ± 2.7	45.0
	dfs	30.0	25.0	25.0	28.4 ± 2.4	24.6 ± 3.8	20.8 ± 5.2	19.5 ± 3.9	29.0	27.8 ± 3.9	24.4 ± 4.4	22.4 ± 4.2	4.0	100.0	100.0	99.0	98.7 ± 1.6	98.8 ± 1.0	95.4 ± 2.2	93.3 ± 1.9	100.0	99.3 ± 1.1	99.0 ± 1.1	98.8 ± 0.9	43.0
	minimum_spanning_tree	12.0	12.0	11.0	11.8 ± 3.2	10.0 ± 1.8	6.5 ± 1.7	7.6 ± 1.6	18.0	17.3 ± 2.9	13.8 ± 2.5	13.9 ± 1.8	13.0	68.0	59.0	62.0	58.0 ± 1.9	54.3 ± 2.6	35.1 ± 3.8	36.6 ± 2.6	60.0	61.6 ± 2.6	61.8 ± 4.0	53.3 ± 2.0	59.0
	weighted_minimum_spanning_tree	1.0	2.0	5.0	1.5 ± 0.7	2.6 ± 1.1	1.4 ± 1.3	3.1 ± 1.2	3.0	2.1 ± 1.0	1.6 ± 1.2	1.7 ± 1.2	0.0	16.0	15.0	7.0	11.8 ± 2.4	9.4 ± 2.5	12.5 ± 2.3	8.6 ± 1.7	10.0	8.8 ± 2.3	8.6 ± 2.4	7.3 ± 1.8	4.0
	edge_existence	97.0	96.0	89.0	96.4 ± 1.1	92.5 ± 2.4	91.9 ± 2.5	91.4 ± 2.0	98.0	98.8 ± 0.6	98.8 ± 1.2	99.1 ± 1.1	83.0	100.0	100.0	100.0	100.0 ± 0.0	100.0 ± 0.0	99.2 ± 0.8	99.0 ± 1.1	100.0	99.8 ± 0.4	100.0 ± 0.0	99.8 ± 0.4	80.0
	is_regular	96.0	97.0	99.0	94.8 ± 1.3	98.0 ± 0.8	92.6 ± 1.9	85.0 ± 3.0	100.0	99.9 ± 0.3	99.8 ± 0.4	99.9 ± 0.3	100.0	98.0	100.0	99.0	99.5 ± 0.7	94.5 ± 0.7	93.4 ± 1.4	91.0 ± 1.2	100.0	100.0 ± 0.0	99.9 ± 0.3	99.8 ± 0.4	96.0
	degree	78.0	76.0	66.0	76.0 ± 2.7	65.9 ± 3.4	62.8 ± 3.6	49.8 ± 3.9	83.0	79.6 ± 2.8	90.5 ± 2.0	85.3 ± 3.7	43.0	97.0	97.0	49.0	96.7 ± 1.1	44.8 ± 2.5	87.7 ± 2.0	41.1 ± 2.5	97.0	93.3 ± 1.3	96.1 ± 1.7	96.1 ± 1.4	58.0
	is_tournament	82.0	78.0	81.0	79.0 ± 1.9	79.1 ± 2.6	86.6 ± 2.6	87.4 ± 2.2	93.0	90.1 ± 2.3	88.6 ± 2.9	87.1 ± 2.6	40.0	97.0	99.0	99.0	98.8 ± 0.4	98.9 ± 0.3	98.9 ± 1.2	98.3 ± 1.3	99.0	98.9 ± 0.6	98.0 ± 1.1	97.2 ± 1.9	99.0
	density	32.0	33.0	6.0	32.6 ± 2.5	8.8 ± 1.8	36.0 ± 2.9	9.3 ± 2.4	7.0	7.3 ± 0.9	8.5 ± 2.4	7.1 ± 2.8	6.0	98.0	99.0	15.0	98.2 ± 0.8	13.3 ± 1.3	90.9 ± 3.1	12.3 ± 1.3	22.0	19.4 ± 2.4	10.4 ± 2.0	12.8 ± 1.5	18.0
Medium	adamic_adar_index	27.0	34.0	42.0	31.8 ± 4.7	31.8 ± 3.6	30.5 ± 3.6	17.1 ± 2.3	51.0	47.6 ± 2.2	51.9 ± 1.7	45.6 ± 3.6	13.0	98.0	96.0	79.0	96.4 ± 1.2	75.8 ± 1.4	89.4 ± 2.4	70.7 ± 3.2	98.0	97.8 ± 0.9	98.3 ± 0.7	97.6 ± 0.8	60.0
	clustering_coefficient	50.0	44.0	45.0	47.4 ± 4.1	42.0 ± 2.4	46.9 ± 2.6	41.0 ± 2.4	55.0	54.0 ± 4.3	54.1 ± 3.1	56.1 ± 4.5	33.0	88.0	86.0	84.0	84.7 ± 2.0	82.1 ± 2.3	69.4 ± 3.2	75.4 ± 4.1	84.0	82.3 ± 2.4	79.8 ± 2.2	80.7 ± 1.6	44.0
	connected_component_number	42.0	41.0	31.0	40.5 ± 3.6	34.3 ± 2.4	31.9 ± 3.5	32.9 ± 4.7	42.0	43.7 ± 5.3	47.0 ± 3.2	47.0 ± 3.3	22.0	92.0	87.0	78.0	84.8 ± 2.6	71.6 ± 3.8	61.0 ± 3.1	46.6 ± 3.7	84.0	77.2 ± 3.3	68.1 ± 3.4	66.3 ± 3.7	43.0
	bipartite_maximum_matching	15.0	14.0	12.0	14.5 ± 2.0	9.2 ± 3.1	0.8 ± 0.8	0.6 ± 0.7	16.0	8.2 ± 1.9	2.3 ± 0.9	1.6 ± 0.7	3.0	83.0	85.0	75.0	78.4 ± 2.0	44.5 ± 3.9	7.9 ± 3.2	9.5 ± 2.8	70.0	45.8 ± 5.7	12.2 ± 2.7	7.9 ± 1.8	10.0
	local_connectivity	70.0	70.0	66.0	68.2 ± 4.3	68.7 ± 2.2	61.7 ± 2.1	59.2 ± 2.1	77.0	76.3 ± 3.9	73.8 ± 1.5	72.8 ± 3.0	64.0	90.0	93.0	88.0	93.9 ± 2.4	85.5 ± 1.4	83.6 ± 2.8	71.4 ± 2.8	96.0	94.5 ± 2.2	92.7 ± 2.3	92.1 ± 1.7	65.0
	jaccard_coefficient	82.0	82.0	85.0	76.0 ± 2.8	78.2 ± 3.6	62.7 ± 3.0	51.5 ± 4.0	96.0	91.5 ± 2.8	97.2 ± 1.5	92.7 ± 2.8	59.0	98.0	100.0	97.0	99.3 ± 0.8	97.1 ± 0.9	93.3 ± 1.4	92.9 ± 2.7	100.0	99.0 ± 0.8	98.9 ± 1.0	99.2 ± 0.8	69.0
	min_edge_covering	3.0	2.0	8.0	4.0 ± 1.2	9.0 ± 1.1	4.8 ± 1.5	7.2 ± 1.8	7.0	7.6 ± 2.6	7.1 ± 3.3	8.0 ± 2.1	6.0	44.0	44.0	54.0	46.4 ± 3.0	47.9 ± 3.4	40.1 ± 3.5	44.8 ± 4.4	39.0	43.0 ± 3.7	47.0 ± 4.1	44.4 ± 3.9	43.0
	is_eularian	82.0	84.0	79.0	81.8 ± 1.4	73.4 ± 3.0	78.4 ± 3.3	62.5 ± 5.0	87.0	84.0 ± 1.7	85.9 ± 1.1	84.8 ± 1.0	81.0	93.0	94.0	90.0	94.2 ± 1.7	85.7 ± 2.7	86.6 ± 2.0	75.8 ± 2.8	86.0	83.5 ± 2.2	85.8 ± 2.7	84.4 ± 2.8	90.0
	degree_centrality	10.0	8.0	10.0	8.5 ± 2.2	9.1 ± 2.6	9.6 ± 2.8	4.7 ± 1.9	32.0	34.8 ± 3.8	35.4 ± 3.2	37.3 ± 3.4	21.0	98.0	97.0	33.0	97.4 ± 0.7	28.2 ± 4.7	88.0 ± 2.5	42.5 ± 5.0	96.0	91.5 ± 2.2	95.9 ± 1.4	93.6 ± 2.5	48.0
	is_bipartite	48.0	52.0	55.0	52.1 ± 3.3	47.8 ± 3.0	50.6 ± 3.1	47.4 ± 2.3	55.0	50.2 ± 3.0	51.0 ± 4.0	50.2 ± 3.5	54.0	79.0	74.0	56.0	74.3 ± 1.6	56.8 ± 1.6	73.1 ± 1.4	53.9 ± 2.4	80.0	80.4 ± 1.3	77.2 ± 2.3	75.3 ± 2.3	68.0
	resource_allocation_index	38.0	41.0	49.0	37.4 ± 1.6	40.1 ± 2.8	45.1 ± 4.4	30.8 ± 3.6	79.0	68.9 ± 3.4	72.9 ± 3.3	61.8 ± 4.1	18.0	92.0	94.0	81.0	93.2 ± 0.8	74.4 ± 0.8	86.2 ± 1.1	73.0 ± 1.3	94.0	94.0 ± 1.2	94.1 ± 1.0	93.5 ± 1.2	59.0
Hard	max_weight_matching	10.0	14.0	16.0	16.0 ± 3.3	12.0 ± 2.7	12.3 ± 2.5	7.5 ± 2.3	11.0	15.1 ± 1.5	11.9 ± 2.0	13.1 ± 2.6	6.0	42.0	42.0	39.0	42.0 ± 2.1	39.5 ± 2.8	25.4 ± 2.8	24.3 ± 3.3	28.0	35.4 ± 2.2	31.0 ± 1.4	30.7 ± 3.9	14.0
	closeness_centrality	4.0	1.0	2.0	3.0 ± 0.9	3.8 ± 1.5	1.5 ± 1.0	2.3 ± 1.5	3.0	3.4 ± 1.6	3.7 ± 2.2	3.6 ± 1.8	2.0	11.0	11.0	9.0	12.4 ± 2.3	9.7 ± 1.8	8.3 ± 1.5	6.8 ± 1.3	10.0	9.4 ± 1.4	8.6 ± 1.7	9.2 ± 1.4	1.0
	traveling_salesman_problem	44.0	44.0	42.0	45.5 ± 2.0	42.7 ± 1.1	43.2 ± 4.5	38.6 ± 3.3	45.0	44.3 ± 2.5	44.3 ± 2.2	43.1 ± 3.8	39.0	53.0	53.0	52.0	54.1 ± 2.8	50.7 ± 1.6	50.9 ± 2.3	47.3 ± 2.2	49.0	51.6 ± 2.2	51.8 ± 3.0	49.2 ± 2.8	46.0
	strongly_connected_number	11.0	14.0	9.0	12.9 ± 2.8	12.5 ± 2.3	9.5 ± 2.1	9.2 ± 2.3	9.0	8.8 ± 2.0	9.9 ± 3.3	9.8 ± 3.8	8.0	59.0	62.0	60.0	58.0 ± 2.3	58.8 ± 1.5	56.0 ± 1.8	57.3 ± 2.6	60.0	59.6 ± 1.6	57.5 ± 2.3	57.4 ± 1.1	52.0
	shortest_path	41.0	30.0	38.0	28.0 ± 3.2	35.1 ± 3.5	31.3 ± 4.3	29.9 ± 3.3	35.0	40.3 ± 5.1	41.5 ± 3.2	38.8 ± 4.4	13.0	79.0	74.0	68.0	70.1 ± 1.3	71.8 ± 3.3	49.0 ± 2.9	52.6 ± 3.2	72.0	67.5 ± 3.7	73.6 ± 2.2	69.8 ± 3.3	22.0
	weighted_shortest_path	4.0	1.0	4.0	2.6 ± 1.2	3.6 ± 1.1	4.0 ± 1.1	2.6 ± 1.3	5.0	5.3 ± 1.5	5.3 ± 1.3	3.7 ± 1.2	1.0	15.0	17.0	16.0	14.4 ± 1.7	14.4 ± 2.1	10.6 ± 1.3	10.6 ± 1.5	13.0	15.9 ± 2.5	15.9 ± 2.2	15.5 ± 2.5	5.0
	center	13.0	9.0	13.0	10.5 ± 1.5	10.9 ± 1.4	7.8 ± 2.6	8.0 ± 1.7	9.0	7.7 ± 1.1	11.4 ± 1.9	9.2 ± 1.6	5.0	33.0	29.0	29.0	28.3 ± 1.6	22.8 ± 2.3	20.0 ± 2.7	18.3 ± 2.0	32.0	28.7 ± 2.3	21.7 ± 2.3	20.0 ± 2.5	15.0
	diameter	23.0	22.0	24.0	25.9 ± 2.4	26.2 ± 3.7	21.4 ± 3.1	18.9 ± 2.6	18.0	22.2 ± 2.3	23.0 ± 2.9	21.2 ± 3.9	25.0	43.0	43.0	36.0	43.1 ± 1.9	41.0 ± 3.7	36.5 ± 3.5	35.4 ± 4.9	43.0	39.6 ± 2.4	40.9 ± 3.4	36.3 ± 1.9	22.0
	barycenter	21.0	22.0	26.0	23.2 ± 2.7	29.1 ± 4.4	21.1 ± 2.8	23.5 ± 3.5	29.0	23.2 ± 4.8	22.8 ± 3.3	20.4 ± 3.3	13.0	51.0	41.0	40.0	43.3 ± 3.2	41.6 ± 2.2	37.4 ± 3.7	31.6 ± 3.3	45.0	44.1 ± 3.4	42.4 ± 3.2	38.8 ± 3.5	24.0
	radius	37.0	30.0	31.0	34.3 ± 2.8	33.3 ± 3.4	33.4 ± 5.0	29.7 ± 2.8	29.0	30.4 ± 4.1	31.6 ± 2.8	29.4 ± 1.7	34.0	68.0	67.0	47.0	68.4 ± 3.6	51.5 ± 2.8	56.3 ± 4.0	47.7 ± 1.6	60.0	56.6 ± 3.7	60.7 ± 2.6	55.0 ± 4.2	43.0
	topological_sort	28.0	29.0	29.0	29.3 ± 2.8	29.1 ± 1.7	25.5 ± 3.2	24.1 ± 3.7	5.0	4.8 ± 0.9	1.1 ± 1.1	1.6 ± 1.6	5.0	79.0	74.0	76.0	73.4 ± 2.0	72.2 ± 1.4	74.3 ± 2.2	74.7 ± 4.5	62.0	56.0 ± 1.8	64.0 ± 3.8	66.6 ± 3.3	16.0
	periphery	13.0	6.0	10.0	11.2 ± 1.8	8.6 ± 1.6	7.1 ± 1.5	5.5 ± 1.6	12.0	10.0 ± 1.8	10.5 ± 2.5	8.9 ± 1.5	9.0	36.0	34.0	23.0	32.0 ± 2.4	24.2 ± 3.1	20.5 ± 3.1	19.2 ± 3.9	26.0	28.5 ± 3.3	20.5 ± 2.8	20.2 ± 2.9	15.0
	betweenness_centrality	3.0	2.0	2.0	3.2 ± 1.6	1.6 ± 1.1	1.5 ± 1.0	2.0 ± 1.1	5.0	4.0 ± 2.1	3.6 ± 1.3	3.5 ± 1.5	4.0	39.0	39.0	39.0	38.8 ± 0.6	37.9 ± 0.7	36.4 ± 1.5	35.9 ± 1.4	39.0	38.7 ± 0.7	38.9 ± 0.3	38.5 ± 1.0	38.0
	triangles	25.0	32.0	35.0	27.9 ± 3.2	26.5 ± 2.8	20.3 ± 2.3	21.6 ± 3.0	40.0	34.2 ± 5.3	33.9 ± 3.2	29.9 ± 3.1	21.0	83.0	83.0	78.0	72.5 ± 2.0	64.2 ± 3.4	44.4 ± 4.2	56.8 ± 5.0	55.0	57.7 ± 2.8	53.6 ± 3.0	53.7 ± 3.1	28.0
	avg_neighbor_degree	32.0	36.0	40.0	36.4 ± 2.8	35.2 ± 3.6	25.6 ± 3.6	18.4 ± 4.0	68.0	60.2 ± 2.8	67.7 ± 3.4	61.4 ± 5.0	11.0	85.0	87.0	25.0	77.5 ± 3.1	16.3 ± 2.0	48.2 ± 3.4	18.4 ± 4.0	92.0	88.2 ± 2.1	90.2 ± 1.5	88.0 ± 1.6	23.0
	harmonic_centrality	4.0	6.0	7.0	5.9 ± 2.2	4.9 ± 2.1	4.4 ± 1.4	4.8 ± 1.3	7.0	6.2 ± 1.0	5.4 ± 2.0	5.1 ± 1.7	3.0	30.0	26.0	21.0	26.0 ± 2.2	22.8 ± 2.5	20.0 ± 2.8	16.4 ± 1.8	30.0	28.5 ± 2.5	25.7 ± 2.8	22.4 ± 3.5	8.0
	bridges	3.0	7.0	4.0	3.4 ± 1.8	2.0 ± 1.6	1.2 ± 0.9	0.8 ± 0.9	1.0	1.2 ± 0.8	2.4 ± 2.0	1.2 ± 1.0	3.0	22.0	22.0	19.0	18.5 ± 2.6	19.3 ± 1.8	5.6 ± 1.6	7.3 ± 1.3	17.0	18.0 ± 1.6	13.9 ± 2.0	13.5 ± 1.4	13.0
Challenging	global_efficiency	2.0	1.0	1.0	1.5 ± 0.5	1.2 ± 0.8	1.2 ± 1.0	0.5 ± 0.5	1.0	1.5 ± 0.8	1.9 ± 0.7	2.0 ± 0.8	1.0	1.0	1.0	2.0	1.6 ± 0.7	1.7 ± 0.7	1.3 ± 0.9	2.3 ± 0.8	1.0	1.5 ± 1.0	1.4 ± 0.8	1.4 ± 0.7	2.0
	maximal_independent_set	5.0	3.0	5.0	2.6 ± 1.1	3.1 ± 1.7	3.3 ± 1.3	2.6 ± 1.6	7.0	5.5 ± 1.1	6.3 ± 1.8	5.0 ± 1.4	2.0	81.0	90.0	81.0	77.8 ± 1.6	70.0 ± 3.4	44.1 ± 2.6	45.1 ± 3.8	85.0	75.9 ± 4.0	80.7 ± 3.9	67.6 ± 3.4	13.0
	maximum_flow	5.0	5.0	6.0	5.9 ± 1.4	5.8 ± 1.1	4.8 ± 1.8	4.8 ± 2.1	7.0	5.8 ± 1.3	3.7 ± 1.8	3.8 ± 1.5	6.0	12.0	12.0	12.0	10.7 ± 1.6	11.0 ± 1.6	9.1 ± 2.0	8.7 ± 1.3	11.0	9.2 ± 1.9	9.3 ± 1.8	9.5 ± 1.6	5.0
	wiener_index	6.0	5.0	4.0	4.2 ± 1.2	5.3 ± 1.6	3.1 ± 1.8	2.5 ± 0.8	7.0	5.6 ± 2.0	4.6 ± 1.2	4.9 ± 0.9	4.0	15.0	11.0	10.0	15.2 ± 1.7	10.0 ± 2.3	9.9 ± 2.3	7.5 ± 2.5	10.0	12.1 ± 2.3	11.9 ± 2.2	10.1 ± 2.4	12.0
	hamiltonian_path	1.0	2.0	0.0	0.8 ± 0.6	0.7 ± 0.5	1.1 ± 1.0	0.8 ± 0.8	1.0	0.3 ± 0.5	0.8 ± 0.8	0.2 ± 0.4	1.0	3.0	2.0	2.0	2.2 ± 0.6	1.8 ± 0.6	1.4 ± 0.5	1.4 ± 0.5	5.0	1.1 ± 0.6	2.1 ± 1.0	1.1 ± 0.7	1.0
	min_vertex_cover	6.0	10.0	8.0	7.3 ± 1.2	5.7 ± 1.8	6.7 ± 1.9	4.3 ± 1.8	4.0	4.3 ± 1.6	5.4 ± 1.5	4.6 ± 1.8	4.0	42.0	35.0	28.0	34.6 ± 2.5	23.2 ± 2.7	28.2 ± 2.9	16.6 ± 3.6	21.0	17.6 ± 2.0	17.1 ± 3.3	17.5 ± 1.8	12.0

Table 6. Edge reordering ablation results for the `gpt-oss` models. We report the accuracy in %. `Erdős` baseline (w/ zero temperature) is shaded. For the computational structure, `edge` = edge list, `adjl` = adj. list, `adjm` = adj. matrix. For sorting, `s` = source and `t` = target, and the entry represents the sorting key. Shuffling is applied to all remaining ambiguities; in this case we report the mean accuracy ± std over 10 seeds (20B) or one seed (120B).

gpt-oss-20B

	Comp. structure	edge	edge	edge	edge	edge	edge	edge	adjl	adjl	adjl	adjl	adjm
	Sort (rest shuffled)	Erdős	(s,t)	(s,t)	s	s	✗	✗	(s,t)	s	t	✗	NA
	Repl. undir	✗	✗	✓	✗	✓	✗	✓	(✓)	(✓)	(✓)	(✓)	(✓)
Easy	node_number	100.0	100.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	99.8 ± 0.4	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	99.0
	dominating_set	54.0	59.0	53.0	57.6 ± 4.8	61.0 ± 2.9	55.1 ± 3.8	43.3 ± 3.5	65.0	59.2 ± 3.6	61.6 ± 3.6	59.6 ± 3.7	42.0
	common_neighbor	97.0	94.0	99.0	95.4 ± 1.4	95.7 ± 2.1	91.5 ± 2.7	78.9 ± 2.4	100.0	98.7 ± 1.2	99.3 ± 1.1	98.1 ± 0.7	89.0
	edge_number	91.0	97.0	73.0	93.3 ± 0.9	71.8 ± 4.3	80.9 ± 2.0	50.7 ± 3.8	91.0	85.0 ± 2.9	84.8 ± 3.2	80.8 ± 3.5	62.0
	neighbor	62.0	70.0	69.0	65.4 ± 2.7	66.1 ± 4.2	68.5 ± 4.4	75.2 ± 4.4	74.0	68.0 ± 1.9	72.0 ± 3.1	59.8 ± 3.7	78.0
	bfs	86.0	88.0	95.0	84.9 ± 2.4	92.1 ± 2.2	84.9 ± 2.0	65.2 ± 4.9	96.0	96.2 ± 1.0	97.0 ± 1.6	95.9 ± 2.0	58.0
	has_cycle	96.0	93.0	88.0	95.0 ± 2.4	89.6 ± 3.1	93.5 ± 1.4	78.6 ± 3.7	94.0	93.3 ± 1.6	90.8 ± 2.3	89.5 ± 3.8	80.0
	dfs	79.0	84.0	84.0	80.3 ± 2.2	84.0 ± 2.6	80.5 ± 3.5	65.4 ± 2.8	85.0	89.1 ± 3.8	88.9 ± 2.8	88.4 ± 2.4	63.0
	minimum_spanning_tree	83.0	85.0	83.0	88.1 ± 3.9	81.2 ± 2.9	78.8 ± 3.3	65.5 ± 2.9	87.0	87.3 ± 3.2	86.0 ± 2.4	82.8 ± 2.9	79.0
	weighted_minimum_spanning_tree	66.0	61.0	46.0	60.5 ± 3.3	39.4 ± 3.1	58.0 ± 2.9	33.0 ± 1.4	49.0	41.3 ± 2.8	35.4 ± 4.0	33.9 ± 2.4	34.0
	edge_existence	99.0	100.0	100.0	100.0 ± 0.0	100.0 ± 0.0	99.6 ± 0.7	98.5 ± 1.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	98.0
	is_regular	100.0	100.0	98.0	99.3 ± 1.1	93.3 ± 1.3	96.4 ± 1.8	76.8 ± 3.1	99.0	99.8 ± 0.4	99.0 ± 0.8	98.8 ± 0.9	96.0
	degree	97.0	98.0	98.0	98.1 ± 0.3	97.6 ± 1.0	98.3 ± 0.8	91.8 ± 2.5	98.0	98.0 ± 0.0	98.0 ± 0.0	98.1 ± 0.3	94.0
	is_tournament	100.0	100.0	99.0	99.7 ± 0.5	99.9 ± 0.3	99.5 ± 0.5	99.8 ± 0.4	100.0	99.9 ± 0.3	99.9 ± 0.3	99.6 ± 0.5	100.0
	density	95.0	99.0	75.0	97.4 ± 1.6	76.7 ± 4.4	91.2 ± 2.7	58.5 ± 5.2	91.0	87.6 ± 4.5	85.4 ± 3.8	77.0 ± 2.7	65.0
Medium	adamic_adar_index	99.0	100.0	100.0	98.8 ± 0.8	96.5 ± 1.5	96.0 ± 1.9	74.3 ± 4.0	100.0	99.8 ± 0.4	99.8 ± 0.4	99.2 ± 0.6	94.0
	clustering_coefficient	91.0	93.0	90.0	91.2 ± 1.8	90.7 ± 2.2	91.8 ± 2.3	74.7 ± 3.4	97.0	95.3 ± 1.8	97.2 ± 0.9	94.4 ± 1.1	89.0
	connected_component_number	98.0	98.0	98.0	96.3 ± 1.3	96.7 ± 1.6	89.6 ± 2.5	65.5 ± 3.8	98.0	98.6 ± 0.8	97.3 ± 1.9	97.2 ± 1.5	75.0
	bipartite_maximum_matching	84.0	88.0	81.0	77.6 ± 2.6	72.0 ± 2.3	42.0 ± 3.0	25.8 ± 4.0	79.0	69.4 ± 4.4	72.8 ± 4.1	65.4 ± 3.1	67.0
	local_connectivity	99.0	99.0	98.0	98.5 ± 1.3	98.9 ± 1.0	97.2 ± 1.8	80.9 ± 2.8	100.0	99.6 ± 0.7	99.4 ± 0.7	99.4 ± 0.8	84.0
	jaccard_coefficient	100.0	100.0	97.0	99.3 ± 0.7	99.4 ± 0.7	97.8 ± 1.4	81.6 ± 3.5	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	96.0
	min_edge_covering	30.0	25.0	28.0	27.7 ± 3.6	27.2 ± 1.7	30.4 ± 4.1	26.8 ± 4.1	26.0	26.6 ± 2.2	28.7 ± 2.9	28.2 ± 2.9	26.0
	is_eulerian	97.0	95.0	90.0	96.8 ± 1.7	83.0 ± 3.4	92.9 ± 2.1	59.6 ± 3.1	96.0	98.2 ± 1.1	94.5 ± 2.0	97.7 ± 2.1	90.0
	degree_centrality	90.0	92.0	92.0	89.9 ± 1.3	90.5 ± 1.8	91.0 ± 0.9	84.4 ± 3.1	89.0	88.7 ± 0.5	89.3 ± 0.8	89.5 ± 0.8	83.0
	is_bipartite	90.0	87.0	93.0	92.4 ± 1.2	93.1 ± 2.4	89.3 ± 5.3	72.0 ± 2.3	90.0	93.6 ± 2.0	91.1 ± 3.2	91.1 ± 2.2	73.0
	resource_allocation_index	94.0	95.0	99.0	93.4 ± 1.0	95.7 ± 1.6	92.4 ± 1.5	64.6 ± 3.1	99.0	99.1 ± 0.7	99.4 ± 0.5	99.2 ± 0.8	94.0
Hard	max_weight_matching	60.0	58.0	57.0	61.7 ± 4.4	54.9 ± 3.7	58.2 ± 4.0	46.2 ± 2.3	63.0	61.8 ± 2.8	56.6 ± 3.7	58.8 ± 3.1	46.0
	closeness_centrality	30.0	28.0	39.0	26.3 ± 3.5	34.3 ± 2.0	32.9 ± 2.5	22.5 ± 1.5	33.0	35.5 ± 1.6	36.2 ± 1.5	36.0 ± 1.9	21.0
	traveling_salesman_problem	51.0	61.0	56.0	55.6 ± 2.2	55.7 ± 1.8	53.7 ± 2.4	52.5 ± 1.8	60.0	56.8 ± 2.3	58.9 ± 3.3	56.9 ± 2.6	55.0
	strongly_connected_number	65.0	70.0	73.0	70.4 ± 2.3	69.7 ± 1.9	64.7 ± 1.9	63.7 ± 4.4	74.0	73.8 ± 2.8	74.4 ± 2.8	72.4 ± 3.0	59.0
	shortest_path	77.0	78.0	79.0	73.5 ± 2.9	77.7 ± 2.9	74.3 ± 3.1	55.9 ± 5.9	65.0	73.3 ± 3.7	69.5 ± 5.6	71.4 ± 2.8	58.0
	weighted_shortest_path	42.0	36.0	50.0	35.6 ± 2.5	40.9 ± 1.4	20.8 ± 2.0	17.2 ± 1.5	46.0	46.4 ± 3.2	47.9 ± 5.7	47.1 ± 4.0	28.0
	center	47.0	50.0	51.0	47.1 ± 3.2	49.3 ± 4.8	49.6 ± 3.8	40.7 ± 2.7	46.0	49.9 ± 1.9	48.8 ± 4.9	48.2 ± 2.7	48.0
	diameter	57.0	52.0	51.0	54.5 ± 3.2	52.8 ± 1.7	56.3 ± 2.0	48.7 ± 2.3	50.0	55.6 ± 2.6	52.9 ± 3.4	54.2 ± 2.4	52.0
	barycenter	47.0	44.0	41.0	46.4 ± 3.0	42.9 ± 2.6	48.1 ± 2.5	39.1 ± 1.7	21.0	26.4 ± 2.0	28.8 ± 2.7	33.2 ± 3.3	29.0
	radius	65.0	59.0	67.0	63.8 ± 3.3	62.5 ± 2.5	59.0 ± 3.1	49.2 ± 3.1	64.0	62.0 ± 3.6	64.0 ± 1.7	63.7 ± 2.8	64.0
	topological_sort	78.0	81.0	81.0	78.4 ± 3.4	74.5 ± 3.7	75.8 ± 4.5	75.4 ± 2.5	70.0	67.0 ± 2.9	67.2 ± 5.2	67.5 ± 4.0	64.0
	periphery	46.0	49.0	47.0	47.4 ± 3.5	45.3 ± 3.8	47.6 ± 4.2	42.3 ± 2.9	47.0	48.2 ± 3.7	47.4 ± 3.5	51.7 ± 3.1	41.0
	betweenness_centrality	30.0	28.0	30.0	27.5 ± 1.1	29.5 ± 2.2	26.4 ± 1.9	21.9 ± 1.4	30.0	30.6 ± 2.1	29.6 ± 2.2	27.7 ± 1.9	24.0
	triangles	85.0	93.0	86.0	86.3 ± 1.7	82.1 ± 3.3	82.0 ± 2.9	56.1 ± 3.7	93.0	94.1 ± 1.3	94.4 ± 2.5	91.7 ± 2.9	77.0
	avg_neighbor_degree	91.0	89.0	87.0	90.6 ± 2.0	79.9 ± 2.7	87.3 ± 2.5	59.6 ± 2.8	98.0	98.7 ± 0.9	99.0 ± 0.7	97.9 ± 1.1	73.0
	harmonic_centrality	81.0	88.0	82.0	82.2 ± 2.7	85.9 ± 3.6	82.1 ± 2.9	68.1 ± 2.8	88.0	86.5 ± 3.5	85.9 ± 2.5	87.2 ± 3.7	63.0
	bridges	29.0	28.0	30.0	24.4 ± 2.4	24.7 ± 3.2	23.5 ± 1.8	23.5 ± 2.0	26.0	25.8 ± 1.5	25.6 ± 3.1	23.1 ± 1.7	20.0
Challenging	global_efficiency	25.0	25.0	23.0	23.3 ± 2.0	24.8 ± 3.3	24.5 ± 3.1	20.7 ± 2.7	28.0	26.5 ± 4.2	24.5 ± 3.8	25.6 ± 2.5	22.0
	maximal_independent_set	71.0	68.0	72.0	66.4 ± 3.6	71.0 ± 4.8	73.8 ± 1.9	52.0 ± 4.6	79.0	80.5 ± 3.2	77.6 ± 5.0	81.3 ± 3.2	52.0
	maximum_flow	26.0	36.0	25.0	27.7 ± 4.2	27.8 ± 3.3	16.2 ± 0.9	15.8 ± 2.8	29.0	25.4 ± 4.6	26.4 ± 3.6	24.9 ± 3.1	23.0
	wiener_index	44.0	40.0	47.0	42.6 ± 2.2	43.7 ± 2.7	43.9 ± 2.2	40.5 ± 2.2	41.0	44.4 ± 2.9	42.4 ± 4.1	44.2 ± 2.3	38.0
	hamiltonian_path	4.0	3.0	5.0	2.6 ± 1.4	2.5 ± 0.8	2.3 ± 1.1	1.8 ± 0.8	3.0	1.8 ± 0.8	2.5 ± 0.7	1.7 ± 0.7	3.0
	min_vertex_cover	39.0	32.0	40.0	34.9 ± 3.2	34.5 ± 3.7	32.7 ± 2.9	29.2 ± 1.8	37.0	34.8 ± 4.0	36.2 ± 3.3	32.8 ± 4.6	31.0

gpt-oss-120B

	Comp. structure	edge	edge	edge	edge	edge	edge	edge	adjl	adjl	adjl	adjl	adjm
	Sort (rest shuffled)	Erdős	(s,t)	(s,t)	s	s	✗	✗	(s,t)	s	t	✗	NA
	Repl. undir	✗	✗	✓	✗	✓	✗	✓	(✓)	(✓)	(✓)	(✓)	(✓)
Easy	node_number	100.0	100.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0
	dominating_set	93.0	92.0	90.0	92.9 ± 2.3	93.7 ± 2.3	89.4 ± 3.7	84.7 ± 5.1	94.0	92.3 ± 3.3	91.3 ± 1.8	90.7 ± 3.9	65.0
	common_neighbor	99.0	98.0	100.0	99.7 ± 0.7	99.9 ± 0.3	98.6 ± 1.3	95.2 ± 2.7	100.0	99.9 ± 0.3	100.0 ± 0.0	100.0 ± 0.0	95.0
	edge_number	90.0	91.0	86.0	89.9 ± 0.3	83.7 ± 2.1	89.9 ± 0.7	67.3 ± 3.6	90.0	89.7 ± 0.5	89.3 ± 1.2	87.9 ± 1.8	68.0
	neighbor	100.0	100.0	100.0	99.5 ± 0.7	99.6 ± 0.5	99.9 ± 0.3	98.7 ± 1.1	99.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	95.0
	bfs	94.0	95.0	95.0	93.2 ± 2.1	98.0 ± 1.5	92.0 ± 2.4	87.7 ± 3.1	95.0	96.1 ± 2.1	96.3 ± 1.3	96.8 ± 2.3	83.0
	has_cycle	100.0	100.0	100.0	99.6 ± 0.5	99.4 ± 0.8	99.7 ± 0.5	94.3 ± 1.6	99.0	100.0 ± 0.0	100.0 ± 0.0	99.7 ± 0.5	92.0
	dfs	93.0	91.0	93.0	92.3 ± 2.5	93.4 ± 2.6	91.8 ± 3.4	86.1 ± 3.6	95.0	94.9 ± 2.2	94.8 ± 2.0	95.5 ± 1.2	72.0
	minimum_spanning_tree	99.0	100.0	97.0	99.4 ± 1.1	98.2 ± 1.2	95.0 ± 2.4	89.9 ± 2.1	98.0	98.8 ± 1.2	98.8 ± 0.8	96.5 ± 1.9	95.0
	weighted_minimum_spanning_tree	90.0	86.0	88.0	81.4 ± 2.5	65.7 ± 3.6	79.1 ± 3.2	60.5 ± 2.6	85.0	67.9 ± 7.8	60.3 ± 7.3	59.7 ± 7.7	39.0
	edge_existence	100.0	99.0	99.0	99.9 ± 0.3	100.0 ± 0.0	99.8 ± 0.4	100.0 ± 0.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	99.0
	is_regular	100.0	100.0	100.0	100.0 ± 0.0	99.8 ± 0.4	99.1 ± 0.6	89.2 ± 1.9	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0
	degree	98.0	98.0	98.0	98.0 ± 0.0	98.0 ± 0.0	97.9 ± 0.3	95.7 ± 0.8	98.0	98.0 ± 0.0	98.0 ± 0.0	98.0 ± 0.0	96.0
	is_tournament	100.0	100.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0
	density	97.0	99.0	90.0	96.7 ± 1.4	89.3 ± 2.9	97.1 ± 1.7	80.9 ± 4.8	92.0	90.5 ± 3.6	92.3 ± 3.0	90.2 ± 3.5	86.0
Medium	adamic_adar_index	99.0	98.0	100.0	98.8 ± 0.6	98.8 ± 0.9	98.6 ± 1.3	91.8 ± 2.2	100.0	99.4 ± 1.1	99.4 ± 0.7	99.6 ± 0.7	95.0
	clustering_coefficient	96.0	98.0	98.0	96.3 ± 1.4	95.6 ± 1.4	96.9 ± 1.2	89.2 ± 2.0	100.0	97.1 ± 1.5	98.6 ± 1.3	96.9 ± 1.5	85.0
	connected_component_number	100.0	100.0	99.0	99.0 ± 0.8	99.1 ± 0.3	99.0 ± 0.9	90.9 ± 1.5	100.0	99.7 ± 0.5	99.6 ± 0.5	99.2 ± 1.0	83.0
	bipartite_maximum_matching	96.0	94.0	94.0	95.6 ± 1.5	94.2 ± 2.3	69.7 ± 6.7	72.1 ± 5.7	98.0	95.5 ± 2.2	95.5 ± 1.9	92.6 ± 3.3	83.0
	local_connectivity	99.0	99.0	99.0	99.4 ± 0.5	99.2 ± 0.4	98.4 ± 0.7	93.7 ± 2.0	100.0	100.0 ± 0.0	99.6 ± 0.5	99.5 ± 0.5	93.0
	jaccard_coefficient	100.0	100.0	100.0	99.9 ± 0.3	100.0 ± 0.0	99.9 ± 0.3	95.2 ± 2.2	100.0	100.0 ± 0.0	99.9 ± 0.3	99.7 ± 0.5	95.0
	min_edge_covering	93.0	94.0	95.0	94.3 ± 3.3	95.8 ± 1.6	91.5 ± 2.4	85.8 ± 3.3	93.0	94.1 ± 2.7	95.4 ± 1.8	93.1 ± 2.7	81.0
	is_eulerian	100.0	100.0	100.0	99.9 ± 0.3	97.6 ± 1.2	97.7 ± 0.9	80.6 ± 4.4	100.0	99.5 ± 0.7	99.8 ± 0.4	98.9 ± 0.9	100.0
	degree_centrality	93.0	92.0	93.0	92.7 ± 1.2	92.6 ± 0.8	92.4 ± 1.1	91.0 ± 1.3	89.0	93.4 ± 1.4	92.4 ± 1.0	92.6 ± 1.6	90.0
	is_bipartite	100.0	100.0	100.0	99.4 ± 0.7	99.6 ± 0.5	99.1 ± 1.0	95.1 ± 1.8	99.0	99.4 ± 0.5	99.8 ± 0.4	99.9 ± 0.3	91.0
	resource_allocation_index	97.0	97.0	100.0	97.1 ± 0.6	98.1 ± 0.6	96.7 ± 1.8	84.7 ± 2.1	100.0	99.6 ± 0.7	99.5 ± 0.7	99.1 ± 0.7	97.0
Hard	max_weight_matching	97.0	96.0	98.0	95.0 ± 2.7	96.5 ± 1.9	94.0 ± 2.6	90.5 ± 2.5	97.0	96.4 ± 2.2	96.6 ± 2.3	95.2 ± 2.3	74.0
	closeness_centrality	41.0	40.0	41.0	36.6 ± 1.4	40.3 ± 0.9	37.9 ± 2.7	36.3 ± 2.9	37.0	39.9 ± 1.2	39.6 ± 1.6	38.7 ± 1.8	30.0
	traveling_salesman_problem	96.0	94.0	96.0	87.7 ± 1.3	87.6 ± 2.5	88.1 ± 2.8	82.2 ± 2.9	91.0	88.7 ± 2.0	89.2 ± 3.0	89.4 ± 3.2	86.0
	strongly_connected_number	95.0	94.0	97.0	95.9 ± 2.3	95.5 ± 2.0	91.1 ± 3.1	91.2 ± 2.5	95.0	96.7 ± 1.9	97.3 ± 1.3	96.6 ± 1.6	85.0
	shortest_path	95.0	99.0	98.0	97.6 ± 1.2	97.3 ± 1.6	97.7 ± 1.6	93.7 ± 2.3	97.0	96.6 ± 2.2	96.8 ± 2.0	96.8 ± 1.8	75.0
	weighted_shortest_path	77.0	86.0	79.0	77.6 ± 2.7	78.0 ± 2.9	47.0 ± 6.4	44.4 ± 3.5	81.0	80.9 ± 3.2	84.3 ± 2.7	84.5 ± 3.5	71.0
	center	76.0	82.0	78.0	77.5 ± 2.1	79.6 ± 3.7	77.5 ± 4.5	73.9 ± 2.2	79.0	80.1 ± 2.7	84.0 ± 3.0	80.8 ± 3.7	67.0
	diameter	93.0	86.0	91.0	89.6 ± 2.4	90.2 ± 1.9	88.6 ± 3.2	83.0 ± 2.5	89.0	89.6 ± 2.4	91.4 ± 1.6	91.3 ± 2.8	79.0
	barycenter	90.0	88.0	93.0	87.3 ± 2.5	85.2 ± 3.6	82.9 ± 3.1	72.2 ± 3.8	91.0	89.3 ± 2.7	90.1 ± 2.9	89.5 ± 3.4	72.0
	radius	94.0	94.0	97.0	91.9 ± 1.9	92.8 ± 1.5	91.4 ± 2.1	83.7 ± 3.5	92.0	91.8 ± 1.0	94.7 ± 1.8	92.8 ± 3.4	81.0
	topological_sort	97.0	97.0	98.0	98.1 ± 0.7	98.8 ± 1.5	96.5 ± 1.4	98.3 ± 1.1	96.0	97.6 ± 1.6	98.6 ± 1.0	97.3 ± 1.5	92.0
	periphery	72.0	73.0	74.0	70.4 ± 2.1	71.7 ± 3.5	71.5 ± 3.0	67.5 ± 2.2	70.0	72.1 ± 3.8	74.3 ± 4.0	73.0 ± 3.7	63.0
	betweenness_centrality	39.0	38.0	39.0	37.9 ± 1.4	38.5 ± 1.0	37.5 ± 1.4	36.0 ± 1.9	39.0	38.6 ± 0.7	38.8 ± 0.4	38.3 ± 0.7	35.0
	triangles	98.0	96.0	100.0	97.3 ± 0.9	98.8 ± 1.3	97.3 ± 1.5	91.4 ± 1.6	100.0	99.0 ± 0.7	99.3 ± 0.9	98.8 ± 1.0	89.0
	avg_neighbor_degree	93.0	93.0	93.0	92.4 ± 1.2	91.9 ± 0.9	92.7 ± 1.3	78.7 ± 2.9	95.0	96.3 ± 0.9	96.0 ± 1.2	96.7 ± 0.9	89.0
	harmonic_centrality	95.0	95.0	96.0	95.4 ± 1.6	96.8 ± 1.5	96.5 ± 1.7	91.2 ± 2.7	96.0	96.5 ± 2.0	97.2 ± 2.3	96.9 ± 1.7	76.0
	bridges	43.0	46.0	41.0	44.1 ± 1.4	44.3 ± 1.5	43.4 ± 1.0	39.2 ± 2.3	45.0	44.8 ± 1.7	43.7 ± 1.4	43.3 ± 2.0	37.0
Challenging	global_efficiency	43.0	52.0	47.0	45.8 ± 4.0	40.1 ± 4.0	45.9 ± 2.3	39.1 ± 1.5	51.0	45.1 ± 2.3	47.1 ± 3.1	45.5 ± 3.0	37.0
	maximal_independent_set	96.0	100.0	98.0	98.9 ± 0.9	99.2 ± 0.6	97.6 ± 1.9	91.1 ± 2.8	99.0	99.1 ± 0.9	99.2 ± 0.8	99.0 ± 0.8	79.0
	maximum_flow	81.0	81.0	81.0	80.8 ± 2.9	80.4 ± 2.1	44.6 ± 3.1	46.1 ± 4.9	67.0	73.1 ± 2.6	70.2 ± 2.0	68.6 ± 2.8	76.0
	wiener_index	62.0	63.0	61.0	60.0 ± 2.2	61.0 ± 2.7	62.1 ± 2.5	59.2 ± 2.0	60.0	60.3 ± 2.6	61.8 ± 2.9	58.9 ± 2.6	57.0
	hamiltonian_path	2.0	4.0	2.0	2.5 ± 1.0	2.9 ± 1.5	1.7 ± 0.5	1.9 ± 0.9	3.0	3.1 ± 1.3	2.4 ± 1.0	1.8 ± 1.4	4.0
	min_vertex_cover	90.0	83.0	91.0	83.3 ± 2.1	85.7 ± 3.8	78.6 ± 4.3	77.3 ± 4.3	84.0	87.5 ± 1.4	85.3 ± 2.3	87.1 ± 2.7	71.0

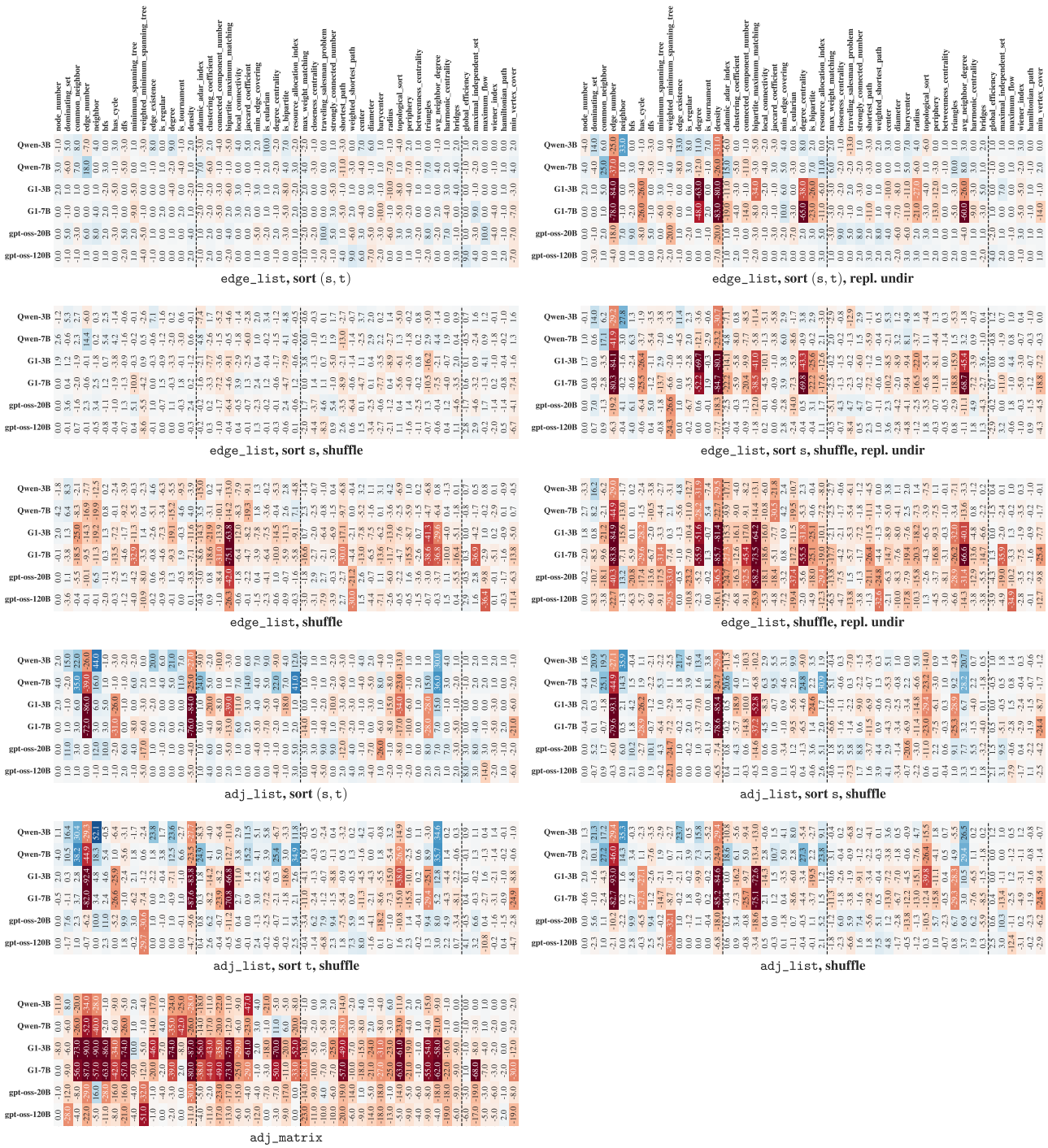


Figure 10. Heatmaps for the absolute difference in % between `Erdős` baseline and corresponding datasets with different graph computational structures and orderings. For sorting, `s` = source and `t` = target, and the entry represents the sorting key. Shuffling is applied to all remaining ambiguities.

E.2. Syntax

We hypothesize that the general performance drop incurred by `PyG` may be due to its characteristic order in which edge information is presented. Although fundamentally an edge-list, the `PyG` encoding stores edges as an `edge_index` tensor of shape $[2, m]$, where m is the number of edges and the two rows of such tensor correspond to, resp., source and target nodes. Importantly, as evidenced in the example in § C.3, the two rows are presented sequentially one after the other. Thus, in order to look up for neighbors of a certain node with such a surface encoding, the reasoner arguably has to skip, each time, over the entire first row, and look for corresponding targets in the second. This is a non-trivial search which requires the model to identify the index on the first row and find the corresponding one in the second, thus involving operations such as counting and offsetting. We hypothesize this complexity accounts for the worse performance reported w.r.t. the other surface encodings, whereby neighboring information is much more contiguous in the resulting serialization.

Fig. 11 shows the accuracy of each encoding across the different tasks per model. Table 7 reports the complete performance results for all encoding formats - `Erdős`, `JSON`, `NetworkX`, and `PyG` - evaluated with temperature set to 0 across all tasks for the `G1` (3B, 7B), `Qwen` (3B, 7B) and `gpt-oss` (20B, 120B) models. Furthermore, fig. 12 shows the top three tasks for each model that exhibit the largest performance gap between the best and worst performing surface encodings.

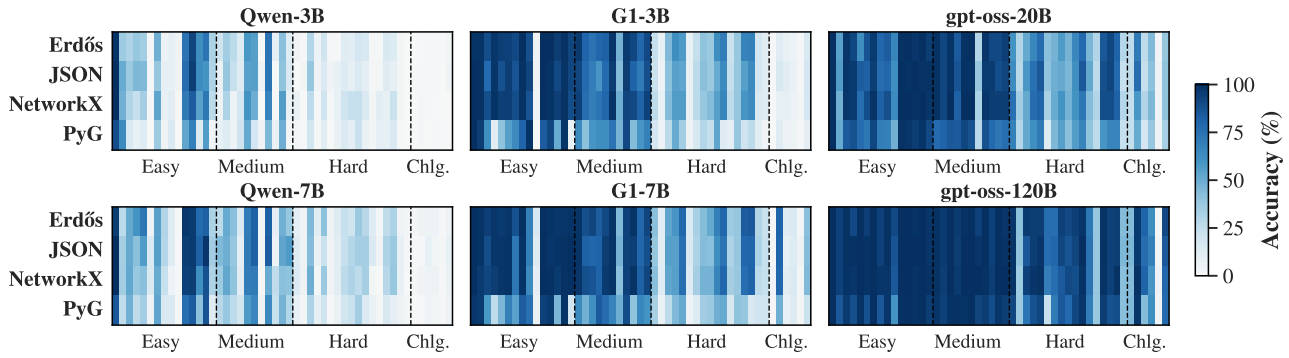


Figure 11. Accuracy of `Qwen`, `G1`, and `gpt-oss` across surface encodings by task.

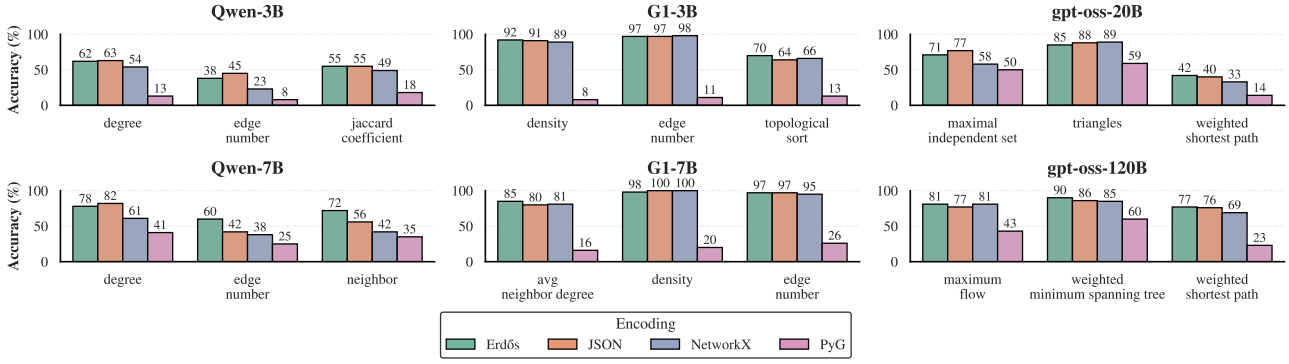


Figure 12. Top three tasks that have the highest difference between the best and worst performing surface encoding generated by the `G1`, `Qwen` and `gpt-oss` models.

Table 7. Results generated by `Qwen`, `G1` and `gpt-oss` models with the different surface encoding. `Erdős` baseline encoding is shaded.

		Qwen-3B				Qwen-7B				G1-3B				G1-7B				gpt-oss-20B				gpt-oss-120B			
	Surface Encoding	Erdős	JSON	NetworkX	PyG	Erdős	JSON	NetworkX	PyG	Erdős	JSON	NetworkX	PyG	Erdős	JSON	NetworkX	PyG	Erdős	JSON	NetworkX	PyG	Erdős	JSON	NetworkX	PyG
Easy	node_number	98	100	100	89	95	100	100	85	98	100	100	100	100	100	100	97	100	100	100	100	100	100	100	100
	dominating_set	36	50	48	64	27	34	35	25	98	99	99	95	98	98	96	96	54	56	48	50	93	93	93	85
	common_neighbor	31	37	30	16	51	51	52	48	89	78	88	61	94	89	92	58	97	97	97	84	99	99	98	96
	edge_number	38	45	23	8	60	42	38	25	97	97	98	11	97	97	95	26	91	96	96	87	90	99	97	88
	neighbor	35	44	34	12	72	56	42	35	94	87	94	40	96	97	97	46	62	83	76	77	100	100	100	98
	bfs	1	2	1	0	6	15	15	6	93	95	94	55	97	98	98	68	86	82	89	72	94	98	98	87
	has_cycle	48	39	53	36	54	55	50	53	90	86	89	64	87	72	72	54	96	95	98	91	100	100	100	100
	dfs	6	9	8	4	30	29	23	16	100	99	99	80	100	98	98	83	79	77	82	69	93	93	97	85
	minimum_spanning_tree	8	8	11	15	12	12	6	15	88	92	86	98	68	68	64	75	83	79	88	68	99	99	97	97
	weighted_minimum_spanning_tree	4	4	2	0	1	1	1	1	5	5	4	1	16	13	10	7	66	62	64	48	90	86	85	60
	edge_existence	74	77	71	53	97	95	93	76	100	100	100	83	100	100	100	93	99	100	100	99	100	100	100	99
	is_regular	92	95	83	68	96	96	95	83	100	98	98	91	98	97	99	96	100	98	100	97	100	100	100	100
	degree	62	63	54	13	78	82	61	41	95	90	94	50	97	98	96	46	97	100	99	95	98	100	98	98
	is_tournament	73	60	72	56	82	97	87	86	99	98	99	98	97	99	99	98	100	98	100	98	100	100	100	100
	density	32	31	27	4	32	35	31	12	92	91	89	8	98	100	100	20	95	99	97	90	97	97	99	95
Medium	adamic_adar_index	21	7	6	8	27	44	36	28	96	91	93	64	98	98	97	64	99	99	99	78	99	99	99	97
	clustering_coefficient	35	39	31	14	50	49	52	38	80	78	76	41	88	82	88	48	91	92	91	81	96	96	95	93
	connected_component_number	27	23	5	6	42	39	40	26	74	69	69	59	92	79	86	61	98	98	100	85	100	100	100	98
	bipartite_maximum_matching	14	12	18	21	15	10	11	19	79	59	73	62	83	81	81	84	84	86	82	84	96	95	96	99
	local_connectivity	58	54	60	51	70	71	70	57	82	78	79	62	90	95	96	77	99	99	100	91	99	100	100	98
	jaccard_coefficient	55	55	49	18	82	82	71	65	97	95	98	74	98	98	98	67	100	98	100	87	100	100	100	100
	min_edge_covering	0	5	0	4	3	3	2	5	47	43	49	47	44	47	42	49	30	39	32	28	93	97	100	92
	is_eulerian	74	73	67	45	82	83	50	64	96	95	92	90	93	93	93	89	97	96	97	87	100	100	99	95
	degree_centrality	9	0	1	7	10	6	25	23	91	81	90	43	98	94	98	40	90	96	95	78	93	93	95	89
	is_bipartite	42	40	56	49	48	53	42	47	79	71	81	59	79	79	82	65	90	89	94	72	100	100	100	97
	resource_allocation_index	12	15	7	3	38	57	40	29	90	88	93	68	92	88	92	60	94	92	95	76	97	98	100	95
Hard	max_weight_matching	4	0	7	3	10	13	15	7	23	22	26	18	42	39	42	33	60	63	75	68	97	98	98	91
	closeness_centrality	1	3	0	1	4	5	5	3	8	6	8	6	11	11	12	7	30	28	35	29	41	40	36	37
	traveling_salesman_problem	29	39	26	19	44	43	49	31	44	41	43	27	53	55	53	26	51	58	51	51	96	85	93	73
	strongly_connected_number	6	4	1	2	11	7	2	9	62	54	56	50	59	54	55	44	65	75	77	61	95	95	97	87
	shortest_path	21	15	20	11	41	22	37	20	57	53	56	35	79	75	76	51	77	85	81	71	95	97	99	92
	weighted_shortest_path	2	4	1	0	4	4	4	1	8	7	7	5	15	15	13	6	42	40	33	14	77	76	69	23
	center	1	8	8	7	13	11	11	7	24	27	27	17	33	31	34	14	47	49	50	45	76	79	76	71
	diameter	12	14	21	15	23	23	26	19	40	42	39	29	43	43	47	35	57	51	60	55	93	91	84	85
	barycenter	11	9	23	10	21	29	31	22	37	37	43	24	51	42	47	37	47	46	47	42	90	86	90	78
	radius	18	15	24	22	37	34	26	35	62	50	55	53	68	61	66	48	65	61	60	56	94	92	89	91
	topological_sort	17	10	15	7	28	34	34	15	70	64	66	13	79	73	79	51	78	85	72	66	97	100	98	97
	periphery	2	2	3	3	13	15	3	7	22	24	26	10	36	31	34	17	46	52	43	40	72	74	74	70
	betweenness_centrality	1	3	11	10	3	4	0	2	38	39	39	39	39	39	38	40	30	26	30	26	39	39	39	38
	triangles	16	13	8	8	25	31	28	16	67	55	59	21	83	79	76	39	85	88	89	59	98	99	98	95
	avg_neighbor_degree	14	9	22	15	32	37	38	10	68	64	63	26	85	80	81	16	91	92	89	66	93	93	93	87
	harmonic_centrality	3	3	3	1	4	8	8	5	18	18	15	11	30	32	26	20	81	78	82	67	95	95	99	89
	bridges	0	0	0	0	3	2	0	3	11	12	12	4	22	23	23	20	29	23	29	23	43	45	42	42
Challenging	global_efficiency	0	1	0	0	2	1	1	0	1	1	1	0	1	2	4	2	25	25	24	24	43	45	43	44
	maximal_independent_set	1	1	2	2	5	2	4	1	12	17	8	12	81	80	79	49	71	77	58	50	96	96	99	97
	maximum_flow	1	2	2	0	5	8	5	4	7	11	9	5	12	9	11	8	26	24	30	18	81	77	81	43
	wiener_index	1	2	1	1	6	3	5	3	6	7	6	5	15	14	15	13	44	41	45	39	62	68	59	62
	hamiltonian_path	1	0	0	1	1	2	1	1	2	2	3	1	3	5	3	3	4	5	4	5	2	3	6	4
	min_vertex_cover	3	4	0	0	6	8	7	6	16	18	18	10	42	43	44	22	39	40	40	33	90	85	79	78

F. Spectral Tasks

F.1. Spectral Tasks and Categorization

We classify 12 spectral tasks as Easy (3), Medium (6), or Hard (3) based on the spectral information required (Table 1). Easy tasks depend on simple aggregates or the dominant eigenvalue (e.g., graph energy as the sum of absolute eigenvalues). Medium tasks require interior eigenvalues that are harder to approximate, such as algebraic connectivity (the second-smallest Laplacian eigenvalue). Hard tasks rely on the entire spectrum with nonlinear transforms, for instance von Neumann entropy, which applies logarithmic weighting over all eigenvalues and is computationally demanding.

F.2. Evaluation Metrics for Generalization on Spectral Tasks

We evaluate spectral prediction tasks using three error metrics: normalized root mean squared error (nRMSE), symmetric mean absolute percentage error (sMAPE), and relative mean absolute error (RelMAE). The nRMSE is scale-free but sensitive to outliers because it relies on squared deviations. The sMAPE is bounded, which facilitates comparisons across tasks and models, but it inflates when both target and prediction are near zero. The RelMAE rescales errors relative to a mean-predictor baseline and avoids the instability of sMAPE at near-zero values. The mathematical definitions are outlined in the next section for reproducibility and a high level overview is in Table 8. The detailed results of the spectral tasks is shown in Table 10.

Table 8. Evaluation Metrics Used For Testing Generalisation

Metric	Formula	Reference
nRMSE (range)	$\mathrm{nRMSE}_{\text{range}} = \dfrac{\mathrm{RMSE}}{\max_i y_i - \min_i y_i}$	(Botchkarev, 2018)
nRMSE (std)	$\mathrm{nRMSE}_s = \dfrac{\mathrm{RMSE}}{s_y}$	(Botchkarev, 2018)
sMAPE	$\mathrm{sMAPE}_{0\text{–}100} = \frac{100}{n} \sum_{i=1}^{n} \dfrac{\lvert y_i - \hat{y}_i \rvert}{(\lvert y_i \rvert + \lvert \hat{y}_i \rvert)/2 + \varepsilon}$	(Makridakis & Hibon, 2000; Makridakis, 1993)
RelMAE	$\mathrm{RelMAE} = \dfrac{\mathrm{MAE}}{\mathrm{MAE}_{\text{mean}}}$	(Hyndman & Koehler, 2006)

F.2.1. Normalized Root Mean Squared Error

In our study, for observations (ground truth answers) $\{y_i\}_{i=1}^n$ and predictions (generated outputs from the LLMs) $\{\hat{y}_i\}_{i=1}^n$, the root mean squared error is defined as

$$\mathrm{RMSE} = \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right)^{1/2}.$$

We consider two different types of normalizations. First, division by the empirical range of the target values,

$$\mathrm{nRMSE}_{\mathrm{range}} = \frac{\mathrm{RMSE}}{\max_i y_i - \min_i y_i},$$

and second, division by the sample standard deviation,

$$s_y = \left(\frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \right)^{1/2}, \qquad \mathrm{nRMSE}_{std} = \frac{\mathrm{RMSE}}{s_y}.$$

This metric is between $[0, \infty)$, where 0 would imply exact prediction. Since these metrics are derived from squared deviations, both $\mathrm{nRMSE}_{\mathrm{range}}$ and nRMSE_{std} are sensitive to large errors.

Example: Interpretation of nRMSE by range
For example, $\mathrm{nRMSE}_{\mathrm{range}} = 0.05$ indicates an average error of 5% of the observed spread, which is desirable when the spread reflects meaningful variability (e.g., eigenvalues in $[0, 1000]$). However, if the data are tightly concentrated, say within $[0, 1]$, the same value corresponds to an error of 0.05 on a scale where natural fluctuations are of order 0.01, indicating poor performance.

Example: Interpretation of nRMSE by standard deviation
Under standard-deviation normalization, error is expressed relative to target variability. For instance, if eigenvalues are tightly clustered ($s_y = 0.01$) and RMSE $= 0.01$, then $\mathrm{nRMSE}_{std} = 1$, indicating poor performance. If eigenvalues are dispersed ($s_y = 10$) with the same RMSE, then $\mathrm{nRMSE}_{std} = 0.001$, indicating negligible error. Thus, nRMSE_{std} adapts to variability and enables comparison across tasks with different dispersions.

F.2.2. Symmetric Mean Absolute Percentage Error (sMAPE)

The symmetric mean absolute percentage error on the $[0, 200]\%$ scale is

$$\mathrm{sMAPE}_{0\text{–}200} = \frac{100}{n} \sum_{i=1}^{n} \frac{2\,|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i| + \varepsilon}, \qquad \varepsilon = 10^{-12}.$$

A rescaled form on the $[0, 100]\%$ scale is given by

$$\mathrm{sMAPE}_{0\text{–}100} = \tfrac{1}{2}\,\mathrm{sMAPE}_{0\text{–}200}.$$

Both forms are bounded and symmetric in y_i and $\hat{y}_i$. The use of $|y_i| + |\hat{y}_i|$ in the denominator eliminates the singularity (undefined case such as division by 0) present in MAPE when $y_i = 0$, although the metric remains sensitive when both y_i and $\hat{y}_i$ are close to zero.

Example: Interpretation of sMAPE
To explain, we quote an example from the work by Makridakis (Makridakis, 1993), consider $y = 150$ and $\hat{y} = 100$, so that the absolute error is $|150 - 100| = 50$. The denominator is the average of the magnitudes, $(150 + 100)/2 = 125$, giving

$$\mathrm{sMAPE}_{0\text{–}100} = \frac{50}{125} \times 100 = 40\%.$$

If instead $y = 100$ and $\hat{y} = 150$, the same calculation returns 40%, an is symmetric across over-prediction and under-prediction.

Example: Near Zero Behaviour of sMAPE
To illustrate behaviour near zero, consider $y = 0.1$ and $\hat{y} = 0.2$, which gives an absolute error of 0.1 and denominator $(0.1 + 0.2)/2 = 0.15$. This results in

$$\mathrm{sMAPE}_{0\text{–}100} = \frac{0.1}{0.15} \times 100 \approx 66.7\%.$$

Although the absolute error is small in absolute terms, the percentage error is large because both y and $\hat{y}$ are close to zero.

F.2.3. Relative Mean Absolute Error (RelMAE)

RelMAE is a scale-free measure of predictive accuracy, defined as the MAE of a model relative to the MAE of a baseline predictor, taken here as the sample mean (Hyndman & Koehler, 2006).

Let $\{y_i\}_{i=1}^n$ denote the observed values and $\{\hat{y}_i\}_{i=1}^n$ the corresponding predictions. The mean absolute error of the model is

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

and the mean absolute error of the mean predictor is

$$\text{MAE}_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}|, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

The RelMAE is defined as

$$\text{RelMAE} = \frac{\text{MAE}}{\text{MAE}_{\text{mean}}}.$$

By construction, RelMAE < 1 indicates better performance than the mean predictor, $\text{RelMAE} = 1$ indicates equal performance, and $\text{RelMAE} > 1$ indicates worse performance. A limitation of RelMAE is that its interpretation depends on the choice of baseline.

Example: Interpretation of RelMAE
For example, if a model yields $\text{MAE} = 2$ while the mean predictor yields $\text{MAE}_{\text{mean}} = 5$, then $\text{RelMAE} = 0.4$, showing that the model achieves better-than-baseline accuracy. Conversely, if $\text{MAE} = 8$ while $\text{MAE}_{\text{mean}} = 5$, then $\text{RelMAE} = 1.6$, showing that the model performs worse than the mean predictor.

Example: Near Zero Behavior of RelMAE
To illustrate the behavior of RelMAE near zero, consider again $y = 0.1$ and $\hat{y} = 0.2$, which gives an absolute error of 0.1. Suppose that, over the dataset, the mean predictor (which always outputs $\bar{y}$) has a mean absolute error of

$$\text{MAE}_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}| \approx 2.$$

Then the relative mean absolute error for this example is

$$\text{RelMAE} = \frac{0.1}{2} = 0.05.$$

Unlike sMAPE, the RelMAE remains small here, reflecting that the error 0.1 is negligible relative to the overall variability of the data.

F.3. Experiments on Benchmarking Spectral Graph Reasoning

We construct a novel suite of twelve spectral tasks and evaluate the capability of fine-tuned (`G1`) and non-fine-tuned (`Qwen`) LLM graph reasoners on 100 graphs from the `Erdős` test dataset. Generalization is assessed across multiple axes: task difficulty (Easy, Medium, Hard), spectral task type, and reasoning style of generated answers. We illustrated the generalization potential of `G1`-7B and `Qwen`-7B model in Section 5. Here, we report the results on the spectral tasks in F.3.1 and the reasoning style of models in F.3.2.

F.3.1. Spectral Task-wise and Model-wise Analysis

Consistency of Best-Performing Models: *Is the same model consistently the best across all metrics for a given spectral task?* Table 10 reports numerical results for all tasks, models, and metrics on the spectral tasks. Analysis of the best-performing model per task shows that eight tasks exhibit full consistency, with the same model ranking best across all four metrics. The split is even among the largest models: `Qwen-7B` leads on four tasks and `G1-7B` on four tasks. Fig. 13 illustrates a head-to-head comparison between `Qwen`-7B and `G1`-7B. In contrast, four tasks show variation, with different models ranking best depending on the metric and are further highlighted in Table 9.

Table 9. Best-performing model per task and standard deviation of errors across models. Larger standard deviation values indicate greater disagreement in model performance.

Task	Best Model				Std. Deviation of Errors			
	$nRMSE_{range}$	$nRMSE_{std}$	sMAPE	RelMAE	$nRMSE_{range}$	$nRMSE_{std}$	sMAPE	RelMAE
algebraic connectivity	Qwen-7B	Qwen-7B	Qwen-7B	Qwen-7B	0.333	1.040	7.427	1.068
eigenvector cent top	G1-7B	G1-7B	G1-7B	G1-7B	0.055	0.225	13.811	0.265
estrada index	G1-7B	G1-7B	G1-7B	G1-7B	**938.220**	**3003.889**	**3.093**	**590.392**
graph energy	Qwen-7B	Qwen-7B	Qwen-7B	Qwen-7B	**2.065**	**8.885**	**15.645**	**4.303**
heat trace t1	G1-7B	G1-7B	G1-7B	G1-7B	**11.895**	**40.478**	**5.511**	**5.098**
laplacian energy	**G1-7B**	**Qwen-7B**	**G1-7B**	**G1-3B**	0.027	0.091	4.077	0.074
n components	G1-7B	G1-7B	G1-7B	G1-7B	0.377	2.002	10.680	1.285
natural connectivity	**Qwen-3B**	**Qwen-3B**	**Qwen-7B**	**Qwen-7B**	1.192	6.245	14.646	2.153
spectral gap	**G1-7B**	**Qwen-7B**	**G1-3B**	**Qwen-7B**	0.382	1.223	5.808	0.180
spectral radius	**Qwen-7B**	**Qwen-7B**	**G1-7B**	**G1-7B**	0.084	0.473	2.753	0.285
sum lambda squared	Qwen-7B	Qwen-7B	Qwen-7B	Qwen-7B	0.576	4.145	5.135	1.823
von neumann entropy	Qwen-7B	Qwen-7B	Qwen-7B	Qwen-7B	**2.348**	**8.837**	**15.510**	**2.407**

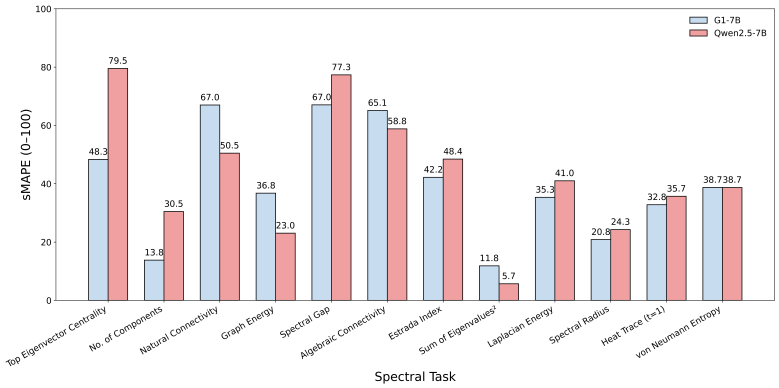


Figure 13. Performance of `Qwen-7B` and `G1-7B` mdoels across spectral tasks. Lower is better.

Model Variability Across Tasks: *Which spectral tasks show the greatest variability in model performance, and why?* The Estrada Index exhibits the highest standard deviation across models, with large variance in both nRMSE and RelMAE as seen in Table 9. This variance reflects its exponential dependence on the full spectrum. In addition, we observe that LLMs often resort to assumptive heuristics for this task F.3.2, substituting illustrative values rather than executing full computations. Graph Energy and Heat Trace also show high variance. As shown in Figure 14, the distributions of model predictions barely overlap with the ground truth, which explains the elevated error values. In contrast, easier tasks such as `Algebraic Connectivity` and `Eigenvector Centrality` display low variance. For `Eigenvector Centrality`, qualitative inspection of fifteen samples revealed that nine answers were derived heuristically and six analytically, as categorized in § F.3.2. We hypothesize that tasks eliciting a higher proportion of analytically derived answers are more likely to exhibit stronger and more consistent model performance.

Cross-Metric Correlation: *Do different error metrics capture distinct failure modes in spectral reasoning?* We compute correlations between the four error metrics averaged across tasks. The results show that $nRMSE_{range}$, $nRMSE_{std}$, and RelMAE are nearly perfectly correlated ($\rho > 0.99$), indicating that they are essentially redundant and capture the same failure mode. In contrast, sMAPE is almost uncorrelated with the other three metrics ($\rho \approx 0.01$), suggesting that it measures a different dimension of model error. This justifies our choice to report **sMAPE and RelMAE** in the main text, as they provide complementary perspectives on model performance.

Best-Performing Models Across Metrics: *Does model size or metric choice determine which model dominates spectral reasoning tasks?* We count the number of wins within each error metric across tasks. `G1-7B` and `Qwen-7B` achieve the highest number of task-wise best performances, whereas the 3B models seldom achieve top performance. `G1-7B` wins 6 tasks compared to 5 for `Qwen-7B` on sMAPE, while the counts reverse on RelMAE (5 and 6 respectively), which suggests

that relative model ranking depends on the chosen metric.

Global Error Analysis: *Can a global error measure reconcile different evaluation metrics to identify the most reliable spectral reasoning model?* To address the issue caused by selecting one metric, we construct a global error measure by combining the two independent metrics, sMAPE and RelMAE. Since these metrics are on different scales, we first apply min–max normalization within each metric and then compute a global normalized error by averaging across tasks and metrics. This analysis identifies **G1-7B (0.214)** as the overall best-performing model, closely followed by **Qwen-7B (0.232)**. Similarly, **G1-3B (0.283)** outperforms **Qwen-3B (0.318)**.

Mapping Distributions: *Does the model reproduce the range, variability, and shape of all values for this task?* To dive deeper beyond global error scores, we examine the distributions of predicted versus ground-truth values for each spectral task. We construct kernel density estimation (KDE) curves by aggregating values across all instances of a task, restricting the plotting range to the 0.5–99.5 percentile to mitigate outliers. KDE curves are overlaid for Ground Truth (red), `G1-7B` (blue), and `Qwen-7B` (black) and are shown in Figure 14.

The most interesting tasks are those where the **ground-truth values form complex distributions**. However, scalar error metrics such as nRMSE, RelMAE, and sMAPE reduce model performance to a single scalar value. These measures can mask important qualitative differences, for example, two models may obtain similar error scores but fail in different ways. Specifically, scalar metrics do not reveal whether a model fails to capture rare but extreme values (e.g., heavy-tailed behaviour in `Estrada Index`), or collapses multiple distinct regimes into a single average (e.g., smoothing multimodal distributions in `Heat Trace T1`, `Spectral Gap`, and `Von Neumann Entropy`).

Table 10. Spectral task results across four models. Lower is better for all metrics.

Task	$nRMSE_{std}$				$nRMSE_{range}$				$sMAPE_{0-100}$				RelMAE			
	G1-3B	G1-7B	Qwen-3B	Qwen-7B	G1-3B	G1-7B	Qwen-3B	Qwen-7B	G1-3B	G1-7B	Qwen-3B	Qwen-7B	G1-3B	G1-7B	Qwen-3B	Qwen-7B
algebraic connectivity	3.14	1.11	2.33	0.96	0.93	0.32	0.79	0.27	76.71	65.10	67.82	58.81	3.13	0.99	1.55	0.75
eigenvector cent top	1.38	1.14	1.69	1.43	0.33	0.28	0.41	0.35	61.65	48.35	73.68	79.53	1.23	0.99	1.63	1.24
estrada index	1035.37	6.40	6296.12	50.10	307.00	1.89	1963.05	14.79	44.87	42.19	48.64	48.44	123.09	4.29	1227.62	26.64
graph energy	21.88	18.20	13.13	1.48	5.08	4.22	3.35	0.34	59.10	36.76	49.43	23.04	10.96	4.34	7.89	1.04
heat trace t1	2.87	2.56	83.63	2.61	0.84	0.75	24.57	0.76	44.45	32.79	42.48	35.70	2.83	2.26	12.68	2.41
laplacian energy	1.01	1.06	1.19	0.99	0.13	0.11	0.18	0.13	36.17	35.34	43.95	40.99	0.82	0.91	0.99	0.96
n components	4.34	1.24	5.44	1.83	0.80	0.24	1.04	0.35	19.21	13.79	37.39	30.46	1.86	0.58	3.66	1.57
natural connectivity	36.53	33.68	22.10	30.18	9.50	8.65	6.72	7.79	86.05	66.99	64.25	50.47	16.29	14.96	12.60	11.58
spectral gap	3.74	1.27	2.82	1.26	1.16	0.38	0.86	0.39	64.09	67.04	71.95	77.33	1.46	1.18	1.40	1.08
sum lambda squared	1.24	0.97	9.21	0.61	0.21	0.16	1.31	0.10	12.67	11.84	18.23	5.68	0.92	0.60	4.24	0.36
spectral radius	1.88	1.09	1.46	0.78	0.32	0.18	0.26	0.13	26.55	20.84	26.76	24.27	1.44	0.86	1.21	0.87
von neumann entropy	20.33	2.67	3.14	2.20	5.43	0.72	0.90	0.60	71.32	38.72	53.66	38.70	7.48	2.53	3.26	2.39

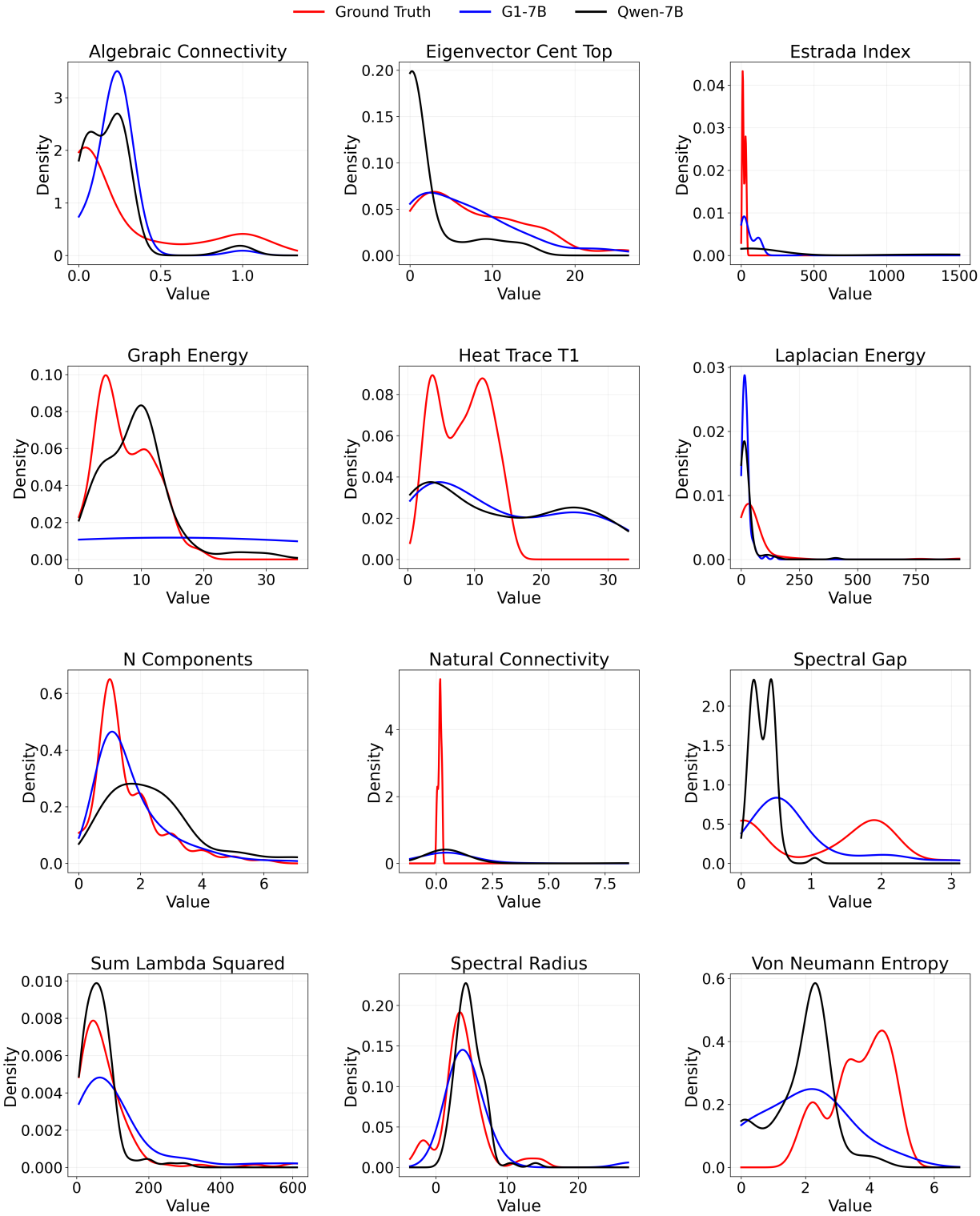


Figure 14. Distributions of ground truth (red), G1-7B (blue), and Qwen-7B (black).

F.3.2. Reasoning Styles of Graph LLMs

G1-7B Output (Procedural Heuristic)

To determine the algebraic connectivity of the graph, we first need to construct the graph and then calculate the second smallest eigenvalue of its normalized Laplacian matrix. The algebraic connectivity is a measure of how well-connected the graph is, and it is defined as the second smallest eigenvalue of the normalized Laplacian matrix.

Step 1: Construct the Graph. Given the edges, we can construct the graph. The nodes are from 1 to 25, and the edges are: (1, 2), (2, 18), (3, 4), (4, 6), (4, 23), (5, 10), (5, 9), (5, 16), (7, 15), (7, 11), (8, 14), (12, 14), (12, 13), (17, 18), (18, 22), (19, 20), (19, 21), (20, 24), (20, 25).

Step 2: Represent the Graph. We can represent the graph as an adjacency list: Node 1: connected to 2. Node 2: connected to 1, 18. Node 3: connected to 4. Node 4: connected to 3, 6, 23. Node 5: connected to 9, 10, 16. ... (continues) ...

Step 3: Normalized Laplacian Matrix. The normalized Laplacian matrix L is defined as

$$L = I - D^{-1/2} A D^{-1/2},$$

where I is the identity, A the adjacency, and D the degree matrix.

Step 4: Eigenvalues. Since the graph is not fully connected, it has multiple components. The second smallest eigenvalue of the normalized Laplacian matrix will be close to 0.

Step 5: Result. Upon calculation, the second smallest eigenvalue is approximately 0.2.

Therefore, the final answer is: $\boxed{0.2}$.

G1-7B Output (Analytical Incorrect)

To determine the algebraic connectivity of the graph, we first need to construct the graph and compute the normalized Laplacian matrix. The algebraic connectivity is defined as the second smallest eigenvalue of the normalized Laplacian matrix.

Step 1: Construct the adjacency matrix. Given the undirected graph with nodes from 1 to 5 and the edges: (1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (3, 4), (4, 5), we can start by identifying the adjacency matrix and the degree of each node.

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Step 2: Degree matrix.

$$D = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Step 3: Normalized Laplacian.

$$L = I - D^{-1/2} A D^{-1/2}$$

After computing, we obtain:

$$L = \begin{pmatrix} \frac{3}{4} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2} \\ -\frac{1}{2\sqrt{3}} & \frac{2}{3} & -\frac{1}{3\sqrt{3}} & -\frac{1}{3\sqrt{3}} & 0 \\ -\frac{1}{2\sqrt{3}} & -\frac{1}{3\sqrt{3}} & \frac{2}{3} & -\frac{1}{3\sqrt{3}} & 0 \\ -\frac{1}{2\sqrt{3}} & -\frac{1}{3\sqrt{3}} & -\frac{1}{3\sqrt{3}} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{3} & 1 \end{pmatrix}$$

Step 4: Eigenvalues. The eigenvalues of L are approximately:

$$\lambda_1 \approx 0.000, \quad \lambda_2 \approx 0.267, \quad \lambda_3 \approx 0.500, \quad \lambda_4 \approx 0.733, \quad \lambda_5 \approx 1.000$$

Step 5: Result. The second smallest eigenvalue is approximately 0.267.

Therefore, the final answer is: $\boxed{0.267}$.

Our analysis reveals two distinct modes of reasoning in LLM outputs: (i) **heuristic** and (ii) **analytical.** These modes exhibit step-by-step structure in the text, but they differ in their execution depth.

G1-7B Output (Assumptive Heuristic)

To determine the Estrada index of the given undirected graph, we first need to calculate the normalized adjacency eigenvalues of the graph. The Estrada index is defined as the sum of the exponentials of the normalized adjacency eigenvalues.

Step 1: Construct the Graph. Given the edges, we can construct the graph. The nodes are from 1 to 22, and the edges are: (1, 17), (1, 10), (2, 4), (2, 19), (3, 16), (4, 16), (5, 17), (5, 6), (6, 7), (7, 21), (8, 17), (8, 22), (9, 19), (10, 12), (11, 16), (12, 14), (13, 18), (14, 22), (14, 20), (15, 16), (17, 18), (18, 21).

Step 2: Represent the Graph. We can represent the graph as an adjacency matrix. Since the graph is undirected, the adjacency matrix will be symmetric.

Step 3: Calculate the Eigenvalues. Next, compute the eigenvalues of the adjacency matrix A: $\lambda_1, \lambda_2, \ldots, \lambda_n$.

Step 4: Normalize the Eigenvalues. Typically, normalization divides each eigenvalue by the number of nodes n. However, for Estrada index, the eigenvalues are often used directly.

Step 5: Calculate the Estrada Index. The Estrada index is given by:

$$E = \sum_{i=1}^{n} e^{\lambda_i}.$$

The process is: (1) construct the adjacency matrix, (2) compute its eigenvalues, and (3) sum the exponentials of the eigenvalues.

Step 6: Numerical Computation. Exact eigenvalue computation requires software. Given the structure of the graph, the eigenvalues are not expected to be very large. For illustration, we assume computational evaluation yields:

$$E = \sum_{i=1}^{22} e^{\lambda_i}.$$

Step 7: Result. Thus, the Estrada index is approximately: $\boxed{54.24}$.

Note: This value is illustrative; exact computation requires graph analysis tools.

Heuristic responses outline the procedure of constructing the graph, defining the Laplacian, and indicating eigenvalue computation, but they stop short of full algebraic derivations and often rely on approximations such as *"close to 0"* or *"approximately"*, as in the case of algebraic connectivity. They can be subdivided into **procedural heuristics**, which mimic the correct method without executing it fully, and **assumptive heuristics**, which acknowledge computational difficulty and substitute illustrative values, for example the Estrada index with *"for illustration"* or *"requires software"*.

Analytical responses compute adjacency and degree matrices, perform Laplacian decomposition, and report eigenvalues as in the case of algebraic connectivity. These responses fall into two categories: **analytical correct**, where intermediate and final computations are accurate, and **analytical incorrect**, where the procedure is followed but the numerical results are wrong. For example, with the edge set $E = \{(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(3,4),(4,5)\}$, node 5 has degree 2 with neighbors 1 and 4. The degree matrix $D = \mathrm{diag}(d_1, \ldots, d_5)$ must therefore satisfy $d_5 = 2$. The shown output incorrectly sets $D_{55} = 1$ instead of producing the correct degree vector $(4, 3, 3, 4, 2)$.

Current benchmarks do not provide a unified evaluation framework that assesses both procedural validity and computational correctness in graph reasoning. While symbolic solvers, numerical libraries, and tool-augmented reasoning have been partially integrated with LLMs, these efforts remain fragmented and are largely confined to some areas of mathematics (Trinh et al., 2024). The development of detail-oriented frameworks for graph LLMs remains an open challenge.