Benchmarking and Enhancing Relational Diagrams Reasoning for Multimodal Large Language Models

Tianyu Hong[1] , **Peng Wang**[1,2*] , **Wenjun Ke**[1,2] , **Yao He**[3] , **Hongao Wang**[1]
[1]School of Computer Science and Engineering, Southeast University, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
[3]Institute of Collaborative Innovation, University of Macau, Macau, China
{hongtianyu, pwang, kewenjun, wanghongao}@seu.edu.cn, mc46477@um.edu.mo

Abstract

Multimodal Large Language Models (MLLMs) have achieved strong performance on a wide range of vision–language tasks. However, their capabilities remain unclear in relational-diagram (RD) reasoning, where correct answers must satisfy diagram-defined constraints such as directed dependencies, branching conditions, and prerequisite relations. RDs, including hierarchical diagrams, fishbone diagrams, and flowcharts, require models to not only retrieve nodes and relations, but also maintain valid reasoning paths over multi-step dependency chains. In this paper, we introduce RD-Bench, a large-scale benchmark of verified diagram–question pairs covering three diagram categories and four difficulty levels. RD-Bench evaluates hierarchical localization, branch-condition selection, constraint-guided traversal, step counting, and prerequisite identification. Experiments on representative open-source and proprietary MLLMs reveal a consistent limitation: models perform much better on single-step reasoning than on multi-step dependency reasoning, especially for complex flowcharts with branching logic and long-range prerequisites. To reduce this gap, we propose RD-MCTS, a Monte Carlo Tree Search framework that incorporates diagram-derived structural priors, constraint-consistent state transitions, and inference-time step-level process rewards. RD-MCTS guides search toward dependency-relevant nodes and valid next steps, improving constraint-consistent reasoning on high-difficulty questions.

1 Introduction

Multimodal Large Language Models (MLLMs) [Liu *et al.*, 2024; Li *et al.*, 2024a; Yue *et al.*, 2024] have made rapid progress on vision–language tasks, including visual question answering [Li *et al.*, 2024b; Li *et al.*, 2022; Lyu *et al.*, 2023], cross-modal retrieval [Li *et al.*, 2022; Guo *et al.*, 2023], and broader multimodal reasoning [Alayrac *et al.*, 2022; Li *et al.*, 2025a; Shenoy *et al.*, 2024; Hong *et al.*, 2023; Ren *et al.*, 2024; Wu *et al.*, 2024a]. However, real-world procedural and organizational knowledge is often expressed through relational diagrams (RDs), such as hierarchical diagrams for system architectures and organizational charts, fishbone diagrams for root-cause analysis, and flowcharts for workflows and business processes. Unlike natural images or unstructured text, RDs explicitly encode dependencies and decision points, requiring models to follow branch conditions, prerequisites, and multi-step dependency chains. This raises a basic yet under-explored question: *Can MLLMs effectively leverage relational diagrams for reasoning, while producing answers and intermediate steps that satisfy the underlying graph constraints?*

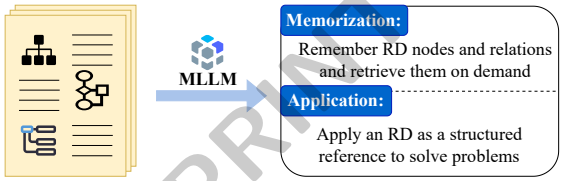

Figure 1: Relational Diagram Reasoning Tasks for MLLMs

We refer to this setting as *relational-diagram reasoning*, and characterize it through two tightly coupled tasks: **Memorization** and **Application**. As illustrated in Figure 1, **Memorization** evaluates whether an MLLM can identify, store, and retrieve RD structural elements, including nodes, directed edges, branching conditions, and prerequisite dependencies, which are not explicitly isolated in existing vision–language benchmarks [Liu *et al.*, 2024; Yue *et al.*, 2024]. **Application** assesses whether the model can use the memorized RD structure for constraint-consistent reasoning, including branch selection, prerequisite-aware traversal, valid step counting, and prerequisite node identification. Although existing multimodal benchmarks broadly cover natural-image VQA and general multimodal reasoning [Alayrac *et al.*, 2022; Li *et al.*, 2024b; Li *et al.*, 2022], systematic evaluation of RD memorization and RD-grounded application under diagram-defined graph constraints remains limited. This gap is important because RDs are common in document-centric work-

*Corresponding author.

flows, such as clinical guidelines, business process specifications, and system design documents, where failures often appear as dependency-order violations, invalid branch selection, or infeasible path traversal.

These two tasks are tightly coupled: weak memorization prevents reliable diagram use, while accurate memorization does not guarantee correct application under multi-step dependency constraints. Consequently, RD reasoning is challenging for current MLLMs for three reasons. First, the model must extract and retain correct dependency structures from pixels, including nodes, edges, directions, and condition attachments; omissions or confusions undermine subsequent retrieval and use [Ware, 2019; Heer *et al.*, 2010]. Second, many tasks require RD-grounded decisions with branching and long-range constraints, where single-step matching is insufficient and consistency must be preserved across transitions. Third, RD categories follow heterogeneous layouts and conventions, requiring models to generalize beyond a single rendering style [Ware, 2019; Heer *et al.*, 2010; Wang *et al.*, 2024]. Consistently, studies on charts and diagrams show that failures often come from incorrect relation handling or constraint violations rather than low-level perception alone [Kafle *et al.*, 2018; Masry *et al.*, 2022], and recent multimodal evaluations show that general benchmark strength does not imply reliability on structured inputs as dependency depth and constraint complexity increase [Yue *et al.*, 2024; Lu *et al.*, 2024; Wang *et al.*, 2024; Wu *et al.*, 2024b].

To enable diagnostic evaluation, we introduce *RD-Bench* (**R**elational **D**iagram **Bench**mark), a large-scale benchmark of rigorously verified diagram–question pairs spanning multiple RD categories and four difficulty levels. RD-Bench targets node/edge retrieval, hierarchical localization, branch-condition selection, constraint-guided path traversal, step counting, and prerequisite identification. Experiments on representative open-source and proprietary MLLMs reveal a consistent bottleneck: models often retrieve single-step RD facts but degrade sharply when applying RDs to multi-step dependency constraints, especially on flowcharts with branching and long-range prerequisites. These results show that current MLLMs struggle to maintain constraint-consistent RD-guided decisions as diagram complexity grows.

Motivated by these failure patterns, we propose *RD-MCTS*, a Monte Carlo Tree Search framework that strengthens RD-based application through inference-time search. RD-MCTS incorporates structural priors, constraint-consistent state expansion, and inference-time step-level process supervision. Empirically, RD-MCTS yields substantial gains on the highest-difficulty subsets, demonstrating the effectiveness of constraint-consistent search for multi-step dependency RD application.

Our contributions are summarized as follows:

- To the best of our knowledge, we are among the first to discuss the capabilities of MLLMs on the relational diagrams reasoning task. We introduce **RD-Bench**, a large-scale and rigorously verified benchmark spanning multiple RD categories and four difficulty levels, designed to diagnose RD *memorization* and RD-grounded *application* under explicit graph constraints.
- We conduct comprehensive evaluations of representative open-source and proprietary MLLMs on RD-Bench, and identify a consistent gap between *memorizing and retrieving* RD structures and *applying* them for constraint-consistent multi-step dependency reasoning, especially in complex diagrams with branching logic and long-range dependencies.
- To address these limitations, we propose **RD-MCTS**, a constraint-consistent MCTS framework with step-level supervision distilled without human-annotated rationales. We will release the benchmark, protocols, and code.

2 RD-Bench: Benchmark Construction

We construct **RD-Bench**, a diagnostic benchmark for evaluating MLLMs on *relational-diagram reasoning* across multiple diagram categories (flowcharts, hierarchical frameworks, and fishbone diagrams). Each instance contains (1) a diagram image rendered from an explicit XML specification and (2) an aligned QA set with level labels. Figure 2 summarizes a three-stage pipeline: structured representation extraction, template matching and difficulty-stratified QA construction, and adversarial verification and rendering.

Benchmark goal. RD-Bench targets *reasoning under graph constraints* and characterizes RD reasoning with two coupled tasks: (1) **memorization**, remembering RD nodes and relations and retrieving them on demand; and (2) **application**, applying a RD as a structured reference to solve problems under diagram-wide constraints. This decomposition helps distinguish failures caused by missing/incorrect structure retrieval from those caused by constraint-inconsistent use of an otherwise correct structure.

Construction principles. We follow four principles to ensure RD-Bench is reliable and diagnostic. **Typicality:** cover representative RD categories and common patterns. **Completeness:** ensure coverage via a unified difficulty taxonomy. **Discriminability:** guarantee each QA has a unique answer determined by diagram constraints. **Fairness:** standardize rendering and balance instances across categories/levels to reduce non-structural confounds.

Structured representation extraction. As shown in Figure 2, we collect (1) diagram templates with clear node–edge structures and (2) domain texts describing procedures, hierarchies, or cause–effect relations. Templates are sourced from public repositories and official galleries with editable files or high-quality exemplars, including drawio-diagrams[1], example draw.io diagrams and templates[2], and bpmn-for-research[3]. We re-create representative templates in draw.io and export XML skeletons as machine-readable formats. For each text segment, an LLM-based extractor produces a category-specific schema (nodes, relations, attributes), and we control structural complexity to support level-controlled memorization and application. We curate templates to cover common patterns and complexity ranges.

[1]https://github.com/jgraph/drawio-diagrams
[2]https://www.drawio.com/example-diagrams
[3]https://github.com/camunda/bpmn-for-research

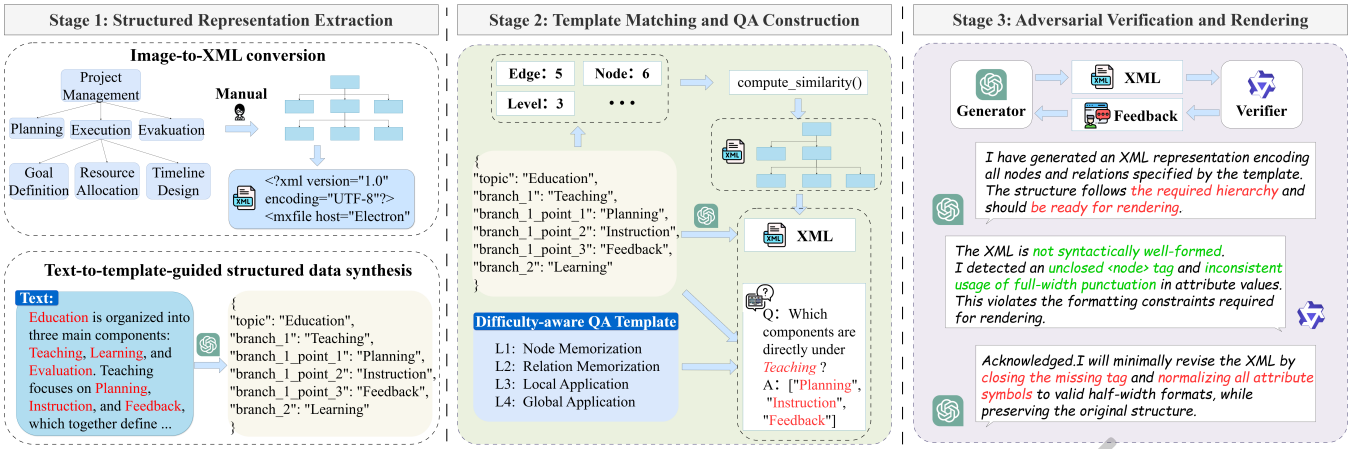

Figure 2: The overall workflow of RD-Bench construction

Template matching and difficulty-stratified QA construction. Following Stage 2 in Figure 2, given a structured schema, we retrieve a compatible XML skeleton using structural features and instantiate it to encode all required nodes and relations. We then generate QA items from a library of category-specific templates and assign them into four levels aligned with memorization–application: **L1** (node memorization), **L2** (relation memorization, including directionality and condition attachment), **L3** (local application under localized constraints), and **L4** (global application requiring diagram-wide constraint consistency). We filter ambiguous items and balance configurations across categories and levels.

Adversarial verification and rendering. As illustrated in Figure 2, we apply a generator–verifier protocol before rendering: a generator proposes an XML candidate; an independent verifier checks XML well-formedness and schema coverage (missing nodes/edges, wrong links/directions, incorrect branching or condition attachment); the generator iteratively revises until all constraints pass. Verified XMLs are rendered into images under standardized visual settings, and each instance is released as {image, QA set with level labels}. This step removes structural inconsistencies and controls non-structural visual factors.

Benchmark statistics and quality. RD-Bench contains **17K** images paired with **390K** QA items over four levels: **39K** (L1), **78K** (L2), **156K** (L3), and **117K** (L4). We split RD-Bench into **14K** images with **313K** QA pairs for training and a held-out **3K** images with **77K** QA pairs for testing. We further conduct human verification on **1K** randomly sampled questions per level, yielding low error rates of **0.2%** (L1), **0.7%** (L2), **2.3%** (L3), and **4.9%** (L4), indicating reliable annotations and scalable construction.

3 Evaluation Metrics

RD-Bench evaluates *relational-diagram reasoning* as formulated in Section 1, which consists of two coupled tasks: (1) **memorization** of RD nodes and relations with on-demand retrieval, and (2) **application** of an RD as a structured reference to solve problems under graph constraints. Our L1–L4 taxonomy is aligned with this view: **L1** targets *node memorization*, **L2** targets *relation memorization* (including directionality and condition attachment), **L3** targets *local application* under localized constraints, and **L4** targets *global application* requiring diagram-wide constraint satisfaction. Based on this decomposition, we define level-aware metrics that support diagnosis across memorization and application.

Following a lightweight canonicalization (lowercasing, punctuation removal, whitespace normalization), we convert both prediction and reference into normalized answer-item sets. Let Q_ℓ denote the set of questions at difficulty level $\ell \in \{1, 2, 3, 4\}$, and let $A_i^{\text{pred}}, A_i^{\text{gt}}$ denote the predicted and ground-truth answer-item sets for instance $i \in Q_\ell$. We add a small constant ϵ to the denominator for numerical stability.

Item-set micro Precision/Recall/F1 and its limitation. We compute overlap between A_i^{pred} and A_i^{gt} and report micro-averaged Precision/Recall/F1 by pooling intersections and set sizes within a level (or a specified subset). For multiple-choice items, micro-F1 reduces to accuracy. However, overlap-based scores are not fully diagnostic for RD-Bench: they can miss constraint violations in L4,such as invalid reachability or termination, and they conflate errors from *memorization* (missing/incorrect nodes or relations) with errors from *application* (using the structure inconsistently). These limitations motivate task-aligned metrics below.

RD-Bench metrics: RS, LSS, and GCSR. To align evaluation with the L1–L4 taxonomy and the memorization–application decomposition, we report three complementary metrics: **RS** for L1–L2 memorization, **LSS** for L3 local application, and **GCSR** for L4 global constraint success. Together, they disentangle structure retrieval from constraint-consistent use beyond surface-form overlap.

Retrieval Score (RS). RS measures *memorization* and aggregates L1–L2, which target node and relation retrieval. We compute RS using micro-F1 on $Q_1 \cup Q_2$:

$$\text{RS} = \frac{2 \sum_{i \in Q_1 \cup Q_2} |A_i^{\text{pred}} \cap A_i^{\text{gt}}|}{\sum_{i \in Q_1 \cup Q_2} |A_i^{\text{pred}}| + \sum_{i \in Q_1 \cup Q_2} |A_i^{\text{gt}}| + \epsilon}. \quad (1)$$

Model Name	Hierarchical diagram			Fishbone diagram			Flowchart		
	RS	LSS	GCSR	RS	LSS	GCSR	RS	LSS	GCSR
LLaVA-1.5-7B	0.775	0.502	0.382	0.689	0.352	0.226	0.718	0.382	0.253
Qwen-2.5-VL-7B	0.763	0.478	0.369	0.703	0.402	0.214	0.741	0.379	0.261
Gemini-2.5-pro	0.836	0.536	0.421	0.759	0.441	0.303	0.795	0.479	0.376
GPT-4o	0.891	0.578	0.492	0.788	0.505	0.389	0.844	0.574	0.451
Claude-3.7	0.913	0.612	**0.559**	**0.856**	0.513	0.492	**0.893**	0.603	**0.478**
Fine-tuned LLaVA-1.5-7B	0.873	0.605	0.519	0.757	0.489	0.483	0.775	0.562	0.397
Fine-tuned Qwen-2.5-VL-7B	**0.919**	**0.627**	0.535	0.817	**0.541**	**0.518**	0.869	**0.619**	0.459

Table 1: Main results on RD-Bench. Fine-tuned + MODEL denotes fine-tuning from the corresponding pretrained checkpoint on our dataset. Best results for each metric are in bold.

RS is diagnostic of whether an MLLM can correctly retrieve RD elements and relations needed for downstream application.

Local Solving Score (LSS). LSS measures *local application* on L3, where questions require constraint-aware use of a bounded diagram context, such as valid local transitions, localized verification, or bounded traversal/counting. We compute LSS using micro-F1 on Q_3:

$$\mathrm{LSS} = \frac{2 \sum_{i \in Q_3} |A_i^{\mathrm{pred}} \cap A_i^{\mathrm{gt}}|}{\sum_{i \in Q_3} |A_i^{\mathrm{pred}}| + \sum_{i \in Q_3} |A_i^{\mathrm{gt}}| + \epsilon}. \quad (2)$$

Compared with RS, LSS reflects whether the model can *use* retrieved structure to perform locally scoped, multi-step RD-grounded decisions.

Global Constraint Success Rate (GCSR). GCSR measures *global application* on L4, where correctness depends on diagram-wide constraints. Each $i \in Q_4$ is associated with a set of verifiable global constraints $\mathcal{G}_i = \{g_{i,1}, \ldots, g_{i,n_i}\}$ derived from the XML/graph representation, such as global reachability, mandatory/unreachable nodes, and termination consistency. We define the per-instance satisfaction indicator:

$$\mathrm{GS}_i = \prod_{k=1}^{n_i} \mathbb{I}\left[g_{i,k}\left(A_i^{\mathrm{pred}}\right) = \mathrm{true}\right], \quad (3)$$

and report the average success rate:

$$\mathrm{GCSR} = \frac{1}{|Q_4|} \sum_{i \in Q_4} \mathrm{GS}_i. \quad (4)$$

GCSR is intentionally stricter than overlap scores: a prediction is counted as correct only if it satisfies all required global constraints, matching RD-Bench's diagnostic goal for constraint-dominated RD application.

4 Empirical Study and Analysis

4.1 Experimental Setup

We conduct extensive empirical experiments on five representative multimodal large models, including Qwen2.5-VL-7B[Bai *et al.*, 2025], LLaVA-1.5-7B-hf[Liu *et al.*, 2023], Gemini 2.5 Pro[Reid *et al.*, 2024], GPT-4o[OpenAI, 2023], and Claude 3.7. For open-source models, we additionally fine-tune them on the RD-Bench training split (14K images with 313K QA pairs) to assess the effect of task-aligned supervision. All fine-tuning runs are performed on an RTX PRO 6000 GPU, using LoRA with $r = 16$ and $o = 32$, a learning rate of 1×10^{-5}, and training for 3 epochs. Unless otherwise specified, we report results on the held-out test split (3K images with 77K QA pairs) using the metrics defined in Section 3: RS (Retrieval Score on L1–L2 for memorization), LSS (Local Solving Score on L3 for local application), and GCSR (Global Constraint Success Rate on L4 for global application under diagram-wide constraints).

4.2 Experimental Results

Experimental results on RD-Bench are summarized in Table 1. We highlight the following findings.

Large gaps across diagram categories. Performance varies substantially across relational-diagram categories. In particular, fishbone diagrams obtain the lowest RS, suggesting that memorizing and retrieving fine-grained factors and their relations is more error-prone in this category. In contrast, flowcharts exhibit the lowest GCSR, indicating that applying the RD to satisfy diagram-wide constraints remains challenging under branching control flow, long-range prerequisites, and termination conditions. These results confirm that different RD categories stress distinct aspects of memorization and constraint-consistent application.

Fine-tuning yields consistent gains. Fine-tuning on the RD-Bench training split consistently improves the open-source models across all diagram categories and all three metrics (RS, LSS, and GCSR). Notably, the fine-tuned Qwen2.5-VL-7B matches or surpasses strong proprietary MLLMs on several settings. This indicates that task-aligned supervision can substantially enhance RD memorization and RD-based application beyond relying on model scale or general-purpose training alone.

Constraint-dominated application is the bottleneck. Across models, RS is often relatively high, while LSS and GCSR are markedly lower, revealing a clear degradation from memorization (RS) to local application (LSS) and further to global constraint-consistent application (GCSR). Even for

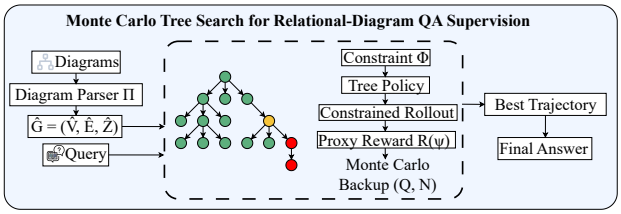

Figure 3: Overall workflow of RD-MCTS.

the strongest models, GCSR typically remains below 54%. This progressive drop highlights that current MLLMs struggle to maintain multi-step dependency consistency and enforce diagram-wide constraints, leaving substantial room for improvement.

5 RD-MCTS: Monte Carlo Tree Search for Relational-Diagram Reasoning

5.1 RD-MCTS Definition

RD-MCTS is an inference-time (test-time) scaling method for relational-diagram reasoning. Given only a relational-diagram image I and a query q, RD-MCTS parses a machine-readable graph and then searches over constraint-consistent reasoning trajectories under a fixed budget (Figure 3). RD-MCTS is used only at inference time.

Diagram parsing. As shown in Figure 3, RD-MCTS first applies an image-to-graph parser Π to obtain an initial diagram graph

$$\hat{\mathcal{G}} = \Pi(I) = (\hat{\mathcal{V}}, \hat{\mathcal{E}}, \hat{\mathcal{Z}}), \tag{5}$$

where $\hat{\mathcal{V}}$ are detected nodes, $\hat{\mathcal{E}}$ are directed relations, and $\hat{\mathcal{Z}}$ contains symbols encoding constraints. RD-MCTS operates on $\hat{\mathcal{G}}$ and allows constraint-consistent corrections during search (Section 5.2).

5.2 Search Space and Constraint Function

Trajectory and answer mapping. We model reasoning as a trajectory

$$x_{1:T} = (x_1, x_2, \ldots, x_T), \tag{6}$$

where each x_t is a valid operation over $\hat{\mathcal{G}}$ (Figure 3), such as traversing an edge, resolving a branch condition, or accumulating required nodes. For structured targets (path extraction, step counting, prerequisite sets), the final answer is deterministically derived from the terminal trajectory; for free-form targets, the terminal trajectory is used as structured evidence for the base MLLM.

State representation. Each state is

$$s_t = (v_t, \mathcal{P}_t, \mathcal{C}_t), \tag{7}$$

where v_t is the current node (or node set), $\mathcal{P}_t$ records satisfied requirements (covered nodes/relations), and $\mathcal{C}_t$ records resolved condition assignments.

Action space. An action $a_t \in \mathcal{A}(s_t)$ is a structure-respecting transition. To mitigate imperfect parsing, we include navigation and minimal repair actions: (1) **Traverse** to move along an outgoing edge; (2) **Condition resolve** to assign an unresolved branch condition and follow the consistent branch; (3) **Requirement completion** to add a missing required node/relation hypothesis when supported by local evidence in $\hat{\mathcal{G}}$.

Feasibility via constraint function Φ. We enforce feasibility with

$$\mathcal{A}(s_t) = \{a \mid \Phi(s_t, a; \hat{\mathcal{G}}, q) = 1\}, \tag{8}$$

which prunes transitions that violate constraints implied by $\hat{\mathcal{G}}$ and q, such as inconsistent condition assignments, skipping mandatory prerequisites, or entering unreachable regions. Thus, the search explores only constraint-consistent partial trajectories (Figure 3).

Terminal condition. A state is terminal if it contains sufficient information to derive the answer required by q. A rollout stops when this holds, when $T_{\max}$ is reached, or when no feasible action exists.

5.3 Tree Policy with Structural Priors

PUCT selection rule. RD-MCTS uses MCTS to explore feasible trajectories under a fixed budget (Figure 3). During selection, actions follow PUCT:

$$a_t^* = \arg\max_{a \in \mathcal{A}(s_t)} \left(Q(s_t, a) + c\, P_{\text{str}}(s_t, a) \frac{\sqrt{N(s_t)}}{1 + N(s_t, a)} \right) \tag{9}$$

where $Q(s_t, a)$ is the empirical mean return, $N(s_t)$ is the state visit count, $N(s_t, a)$ is the action visit count, and c controls exploration.

Structural prior P_{str}. The prior $P_{\text{str}}(s_t, a)$ is computed from $\hat{\mathcal{G}}$ and q, combining: (1) **Proximity** (reducing distance to a query-implied target), (2) **Requirement coverage** (increasing unresolved coverage in $\mathcal{P}_t$), and (3) **Semantic relevance** (embedding similarity between q and the node/edge/condition text in $\hat{\mathcal{G}}$). All signals are computed at inference time without any ground-truth XML.

Constraint-consistent expansion. After selection, we expand by adding feasible successors induced by $\mathcal{A}(s_t)$. Feasibility is guaranteed by Φ, so newly created states remain consistent with condition and requirement constraints.

5.4 Rollout, Value Backup, and Prediction

Constrained rollout. From an expanded state s_t, RD-MCTS samples feasible actions until termination, producing a terminal trajectory $x_{1:T}^{(\psi)}$ and a candidate answer $\hat{a}^{(\psi)}$ derived from that trajectory. Rollouts terminate early at dead-ends (no feasible action).

Inference-time step-level reward (no gold answers). Since RD-MCTS runs at reasoning time without gold answers, we compute each rollout ψ with a proxy reward (Shown in Figure 3):

$$R(\psi) = \lambda \sum_{t=1}^{T} r_{\text{con}}(s_t) + \mu\, r_{\text{prog}}(x_{1:T}^{(\psi)}, q) + \nu\, r_{\text{ver}}(I, q, \hat{a}^{(\psi)}) \tag{10}$$

Model Name	Method	Hierarchical diagram			Fishbone diagram			Flowchart		
		RS	LSS	GCSR	RS	LSS	GCSR	RS	LSS	GCSR
LLaVA-1.5-7B	Vanilla	0.775	0.502	0.382	0.689	0.352	0.226	0.718	0.382	0.253
	Few-shot	0.791	0.523	0.399	0.693	0.381	0.261	0.741	0.427	0.264
	RD-MCTS	0.775	0.552	0.443	0.689	0.415	0.303	0.718	0.439	0.291
	Self-Training	0.873	0.605	0.519	0.757	0.489	0.483	0.775	0.562	0.397
	Self-Training + Few-shot	0.902	0.614	0.532	0.775	0.503	0.487	0.815	0.589	0.412
	Self-Training + RD-MCTS	0.873	0.646	0.562	0.757	0.531	0.509	0.775	0.634	0.464
Qwen-2.5-VL-7B	Vanilla	0.763	0.478	0.369	0.703	0.402	0.214	0.741	0.379	0.261
	Few-shot	0.767	0.516	0.429	0.727	0.445	0.261	0.766	0.417	0.313
	RD-MCTS	0.763	0.532	0.477	0.703	0.461	0.330	0.741	0.443	0.376
	Self-Training	0.919	0.627	0.535	0.817	0.541	0.518	0.869	0.619	0.459
	Self-Training + Few-shot	**0.927**	0.646	0.550	**0.841**	0.572	0.545	**0.882**	0.633	0.539
	Self-Training + RD-MCTS	0.919	**0.662**	**0.603**	0.817	**0.590**	**0.563**	0.869	**0.687**	**0.592**

Table 2: Main results on RD-Bench under different inference strategies. Best results for each metric are in bold.

where $r_{\text{con}}(s_t)$ encourages intermediate constraint consistency and requirement coverage, r_{prog} measures progress toward the terminal condition, and r_{ver} optionally verifies that $\hat{a}^{(\psi)}$ is supported by the image and parsed evidence without gold labels.

Concrete form of $r_{\text{con}}(s_t)$. We instantiate

$$r_{\text{con}}(s_t) = \alpha_1 \, \Delta\text{Cov}(\mathcal{P}_t) - \alpha_2 \, \text{Conf}(\mathcal{C}_t) - \alpha_3 \, \text{Dead}(s_t). \tag{11}$$

where $\Delta\text{Cov}(\mathcal{P}_t)$ is incremental requirement coverage, $\text{Conf}(\mathcal{C}_t)$ penalizes inconsistent condition assignments, and $\text{Dead}(s_t)$ penalizes states close to dead-ends under Φ.

Monte Carlo value estimation and backup. We estimate state value by averaging over K rollouts:

$$V(s_t) = \mathbb{E}_{\psi \sim \Psi(s_t)}[R(\psi)] \approx \frac{1}{K} \sum_{\psi=1}^{K} R(\psi), \tag{12}$$

and backpropagate returns along the path to update $Q(s_t, a)$ and visit counts.

Inference-time prediction. After exhausting the computation budget (Figure 3), we select the final trajectory by maximum visit count:

$$x^*_{1:T} = \arg\max_{x_{1:T}} N(x_{1:T}), \tag{13}$$

(or equivalently by maximizing Q). The selected trajectory yields the final answer, via deterministic extraction for structured targets or evidence-grounded generation for natural-language targets.

6 Experiments

6.1 Experimental Setup

We evaluate RD-MCTS on two open-source MLLMs, LLaVA-1.5-7B and Qwen2.5-VL-7B, on RD-Bench. For each backbone, we report results in both *pre-finetuning* and *post-finetuning* regimes, where *post-finetuning* denotes supervised fine-tuning on the RD-Bench training split (14K images, 313K QA pairs). We compare three inference configurations: **Vanilla**, **Few-shot**, and **RD-MCTS**. Unless otherwise specified, we evaluate on the held-out test split (3K images, 77K QA pairs) using Section 3: **RS** (L1–L2 memorization/retrieval), **LSS** (L3 local application), and **GCSR** (L4 global application success under constraints).

6.2 Main Results

RD-MCTS improves constrained solving and complements finetuning. RD-MCTS targets *application* rather than *memorization*. Accordingly, within the same training regime, RS (L1–L2) stays the same when switching from Vanilla/Few-shot to RD-MCTS, while gains concentrate on LSS and GCSR (L3–L4). As shown in Table 2, RD-MCTS outperforms Few-shot on constrained solving across both backbones and all diagram categories. For LLaVA-1.5-7B, RD-MCTS improves GCSR from 0.382 to 0.443 on hierarchical diagrams (+0.061) and from 0.226 to 0.303 on fishbone diagrams (+0.077), while also raising LSS from 0.502 to 0.552 (+0.050) and from 0.352 to 0.415 (+0.063), respectively. On flowcharts, RD-MCTS yields a clear boost on LSS (0.382→0.439, +0.057) and a consistent gain on GCSR (0.253→0.291, +0.038), indicating better constraint-consistent multi-step decisions under branching control flow.

The improvements are larger on Qwen-2.5-VL-7B, suggesting that a stronger base model benefits more from constraint-consistent search. RD-MCTS increases GCSR by more than +0.10 across all three categories: 0.369→0.477 on hierarchical diagrams, 0.214→0.330 on fishbone diagrams, and 0.261→0.376 on flowcharts. Similar trends hold for LSS, showing that RD-MCTS improves both locally bounded application and diagram-wide constraint satisfaction. Compared with Few-shot prompting, which brings modest gains (e.g., Qwen flowchart GCSR 0.261→0.313), RD-MCTS delivers substantially larger improvements on the constraint-dominated metric (0.261→0.376), highlighting the advantage of explicit search over purely prompt-based guidance.

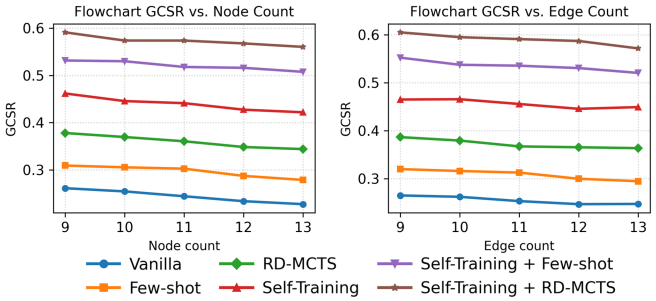

Figure 4: Effect of Flowchart Complexity on GCSR: Node Count vs. Edge Count

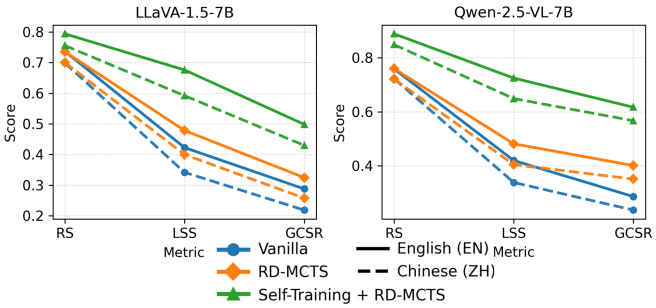

Figure 5: Language Impact on Flowchart Reasoning: English vs. Chinese across RS, LSS, and GCSR

6.3 Analysis Experiments

We conduct fine-grained analyses on the flowchart subset of RD-Bench, reporting RS, LSS, and GCSR unless otherwise specified.

Flowchart size (node count). Figure 4 (left) shows that GCSR decreases as node count increases (9→13) across all inference settings, indicating that larger flowcharts introduce longer dependency chains and make globally constraint-consistent application harder.

Flowchart dependency load (edge count). Figure 4 (right) shows a similar downward trend as edge count increases, suggesting that denser dependencies further exacerbate global consistency. Under the same x-axis value, edge-count curves are slightly higher than node-count curves, implying that within this range, additional edges may offer explicit structural cues, while additional nodes mainly lengthen reasoning chains and increase compounding errors.

Language effect (Chinese vs. English). Figure 5 compares Chinese and English flowcharts with matched structure/content. English consistently outperforms Chinese on RS, LSS, and GCSR for both backbones and all settings. Moreover, the RS→LSS drop is generally smaller in English, suggesting that Chinese inputs incur a larger degradation when moving from memorization/retrieval to multi-step local application, which then propagates to lower global success.

Diagram form (layout sensitivity). Figure 6 compares three renderings with identical text: vertical frameworks, horizontal frameworks, and fishbone diagrams. Vertical frameworks score highest, horizontal frameworks are intermediate, and fishbone diagrams perform worst across both backbones and metrics, showing that layout conventions affect RD-grounded application beyond text, likely through relation readability and implied reading order.

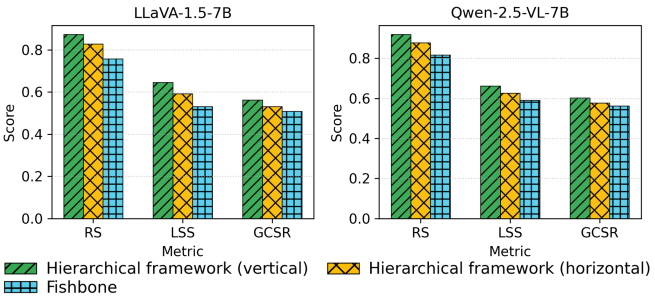

Figure 6: Layout Sensitivity in Relational-Diagram Reasoning: Vertical vs. Horizontal Frameworks vs. Fishbone

7 Related Work

Multimodal Large Language Models. MLLMs combine language models with vision encoders and achieve strong performance via instruction tuning on general vision–language benchmarks [Achiam *et al.*, 2023; Liu *et al.*, 2024; Li *et al.*, 2024a; Yue *et al.*, 2024]. Modular tool-based variants further decompose perception and reasoning, such as Visual ChatGPT and MM-ReAct [Wu *et al.*, 2023; Yang *et al.*, 2023]. Our work targets a less-studied setting: relational-diagram reasoning where correctness is determined by *global* structural constraints.

Evaluation benchmarks for MLLMs. General multimodal benchmarks are dominated by natural images and offer limited coverage of structured diagrams that require constraint-consistent reasoning [Liu *et al.*, 2024; Li *et al.*, 2024a; Yue *et al.*, 2024]. Prior diagnostic evaluations and chart/diagram datasets show sharp degradation as layouts and structures grow more complex [Fu *et al.*, 2025; Yu *et al.*, 2024; Chen *et al.*, 2024; Masry *et al.*, 2022; Lu *et al.*, 2024; Wang *et al.*, 2024]. Recent vector-graphic benchmarks further reveal that MLLMs struggle with fine-grained structured visual understanding and code-level generation under strict graphical constraints, complementing bitmap-centric evaluations [Li *et al.*, 2025b]. RD-Bench fills the remaining gap with multi-category relational diagrams, structure-grounded verification, and L1–L4 difficulty stratification; RD-MCTS further improves high-difficulty performance via constraint-consistent search.

8 Conclusion

We introduce **RD-Bench** for benchmarking MLLMs on relational-diagram reasoning under global graph constraints, covering multiple diagram categories with verified diagram–question pairs and L1–L4 difficulty stratification. We further propose **RD-MCTS**, a test-time constraint-consistent search method with structural priors that consistently improves high-difficulty performance over vanilla and few-shot inference, and complements supervised finetuning.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work was supported by the National Science Foundation of China (Grant No. 62376057), the Start-up Research Fund of Southeast University (RF1028623234), and the Big Data Computing Center of Southeast University.

References

[Achiam *et al.*, 2023] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Alayrac *et al.*, 2022] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 23716–23736, 2022.

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

[Chen *et al.*, 2024] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? In *AProceedings of the 38th International Conference on Neural Information Processing Systems*, pages 27056–27087, 2024.

[Fu *et al.*, 2025] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *NeurIPS Datasets and Benchmarks Track*, 2025.

[Guo *et al.*, 2023] Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, et al. Point-bind & point-llm: Aligning point cloud with multi-modality for 3d understanding, generation, and instruction following. *arXiv preprint arXiv:2309.00615*, 2023.

[Heer *et al.*, 2010] Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *Communications of the ACM*, 53(6):59–67, 2010.

[Hong *et al.*, 2023] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 20482–20494, 2023.

[Kafle *et al.*, 2018] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering. In *CVPR*, pages 5648–5656, 2018.

[Li *et al.*, 2022] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900, 2022.

[Li *et al.*, 2024a] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *CVPR*, pages 13299–13308, 2024.

[Li *et al.*, 2024b] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *CoRR*, 2024.

[Li *et al.*, 2025a] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Joshua Adrian Cahyono, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47:7543–7557, 2025.

[Li *et al.*, 2025b] Jinke Li, Jiarui Yu, Chenxing Wei, Hande Dong, Qiang Lin, Liangjing Yang, Zhicai Wang, and Yanbin Hao. Unisvg: A unified dataset for vector graphic understanding and generation with multimodal large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 13156–13163, 2025.

[Liu *et al.*, 2023] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[Liu *et al.*, 2024] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, pages 216–233. Springer, 2024.

[Lu *et al.*, 2024] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024.

[Lyu *et al.*, 2023] Chenyang Lyu, Minghao Wu, Longyue Wang, Xinting Huang, Bingshuai Liu, Zefeng Du, Shuming Shi, and Zhaopeng Tu. Macaw-llm: Multi-modal language modeling with image, audio, video, and text integration. *arXiv preprint arXiv:2306.09093*, 2023.

[Masry *et al.*, 2022] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of ACL*, pages 2263–2279, 2022.

[OpenAI, 2023] GPT OpenAI. 4v (ision) system card https://cdn. openai. com/papers. *GPTV_System_Card. pdf*, 2023.

[Reid *et al.*, 2024] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou,

Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, 2024.

[Ren *et al.*, 2024] Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. Timechat: A time-sensitive multimodal large language model for long video understanding. In *CVPR*, pages 14313–14323, 2024.

[Shenoy *et al.*, 2024] Ashish Shenoy, Yichao Lu, Srihari Jayakumar, Debojeet Chatterjee, Mohsen Moslehpour, Pierce Chuang, Abhay Harpale, Vikas Bhardwaj, Di Xu, Shicong Zhao, et al. Lumos: Empowering multimodal llms with scene text recognition. In *KDD*, pages 5690–5700, 2024.

[Wang *et al.*, 2024] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 113569–113697, 2024.

[Ware, 2019] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[Wu *et al.*, 2023] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

[Wu *et al.*, 2024a] Jianzong Wu, Xiangtai Li, Chenyang Si, Shangchen Zhou, Jingkang Yang, Jiangning Zhang, Yining Li, Kai Chen, Yunhai Tong, Ziwei Liu, et al. Towards language-driven video inpainting via multimodal large language models. In *CVPR*, pages 12501–12511, 2024.

[Wu *et al.*, 2024b] Yifan Wu, Lutao Yan, Leixian Shen, Yunhai Wang, Nan Tang, and Yuyu Luo. Chartinsights: Evaluating multimodal large language models for low-level chart question answering. In *Findings of EMNLP*, pages 12174–12200, 2024.

[Yang *et al.*, 2023] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

[Yu *et al.*, 2024] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: evaluating large multimodal models for integrated capabilities. In *ICML*, pages 57730–57754, 2024.

[Yue *et al.*, 2024] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *CVPR*, pages 9556–9567, 2024.