Introduction

Hypospace Intro

Many scientific problems are characterized by uncertainty: multiple hypotheses with different mechanisms can be consistent with the same set of observations. Therefore Hypospace is introduced in three structured domains:

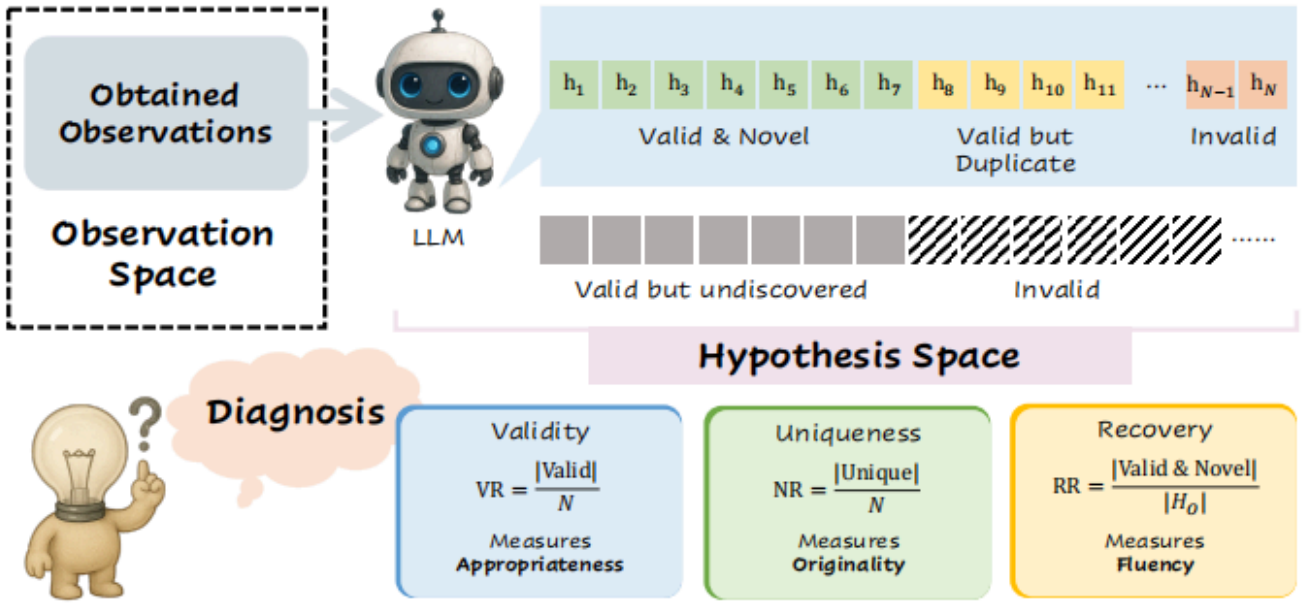
- causal graphs from perturbations
- gravity constrained 3D voxel reconstruction from top-down projections
- Boolean genetic interactions

The outcome is evaluated through three complementary indicators: Validity (precision of proposals consistent with observations), Uniqueness (non-redundancy among proposals), and Recovery (coverage of the enumerated admissible set).

Moreover, deterministic validators and enumerated ground truth are provided inside Hypospace to make the benchmark basis objective.

When a question is raised towards the target LLM, the hypothesis space consists of its responses and the admissible set $\mathcal{H}_O$, which can be split into four kinds:

- Valid and Novel: Both valid and not repeated with admissible set
  - VR: $\frac{|\text{Valid}|}{N}$
  - NR: $\frac{|\text{Unique}|}{N}$
- Valid but Duplicate: Valid but appeared in admissible set before
- Invalid: Make no sense
- Valid but Undiscovered: Existed in admissible set but LLM not successfully generated
  - RR: $\frac{|\text{Valid and Novel}|}{|\mathcal{H}_O|}$



(a) HypoSpace Evaluation Framework.

Math Intuition

Problem Setup:

H **(Hypothesis space)**: All theoretically possible hypothesis

$\mathcal{O}$ **(Observation)**: The actual observation input from LLM

$\mathcal{H}_O$ (Admissible set): The subset of all possible H with consistent data with observation $\mathcal{O}$

For Hypospace setup, all elements in $\mathcal{H}_O$ is given and known.

- Soundness: every element is the fact(correct)
- Completeness: $\mathcal{H}_O$ contains all correct hypothesis
- Controllability: benchmark difficulty is controlled through altering task parameters(nodes, grid , etc.)

Tested LLM is regarded as a hypothesis sampler. Given a fixed prompt, it will generate N independent answers, $P = \{\tilde{h}_1, \ldots, \tilde{h}_N\}$.

- N equals to $\mathcal{H}_O$ by default, maximizing the probability of finding out all standard answers one by one.

The math description of VR/NR/RR is listed below:

$$\mathrm{VR}(P) = \frac{1}{N} \sum |\{\tilde{h} \in P \mid \mathrm{val}_O(\tilde{h}) = 1\}|$$

$$\mathrm{VR}(P) = \frac{1}{N} \sum |\{\tilde{h} \in P \mid \mathrm{val}_O(\tilde{h}) = 1\}|$$

$$\mathrm{RR}(P) = \frac{1}{|\mathcal{H}_O|} |\{\tilde{h} \in P \mid \mathrm{val}_O(\tilde{h}) = 1 \wedge \mathrm{nov}(\tilde{h}; A_{\tilde{h}}) = 1\}|$$

Causal Inference

Our group has split the whole task into three parts, everyone in charge of one part. My part is the causal graph inference benchmark, so I will focus on this benchmark in the content below.

Problem Setup:

Define$V = \{v_1, \ldots, v_n\}$as the nodes with labels and $G^* = (V, E^*)$ a latent DAG.LLM can't directly observe the whole DAG, instead it receives numerous results on m times of single-node interventions and its results:

$$\mathcal{O} = \{(s^{(k)}, x^{(k)})\}_{k=1}^{m}, s^{(k)} \in V, x^{(k)} \in \{0,1\}^n$$

The results are defined as below:

$$[F_G(v_i)]_j = \begin{cases} 0, & j = i \\ 1, & \forall j \in \mathrm{desc}_G(v_i) \\ 0, & \text{otherwise.} \end{cases}$$

Remind that once the starting point's forward model is set to 1, then no matter the destination node, all its descendants will be 1 as well, while other nodes that are not connected to v_i will be set to 0.

LLM Task:

Model reads the $\mathcal{O}$ and T_{prompt}, and generated a candidate DAG:

$$\tilde{G} \leftarrow F_{\text{LLM}}(\mathcal{O}, A_{\tilde{G}}, T_{\text{prompt}})$$

where $A_{\tilde{G}}$ is the context history.

Validation:

When all the interventions are simulated other the candidate DAG and receive the positive outcomes, the DAG will be decided as affirmative:

$$\mathrm{val}_O(\tilde{G}) = 1 \Leftrightarrow F_{\tilde{G}}(s^{(k)}) = x^{(k)}\ \forall k$$

Especially, two DAGs are perceived identical only when their normalized forms are the same, which is decided by the nodes and the directions of all sides:

$$\mathrm{Canon}(G) = \mathrm{Canon}(G')$$

Scoring:

The LLM will be informed to generate N different candidate DAGs $P = \{\tilde{G}_i\}_{i=1}^{N}$,after which these DAGs will be evaluated through 3 indexes mentioned above.

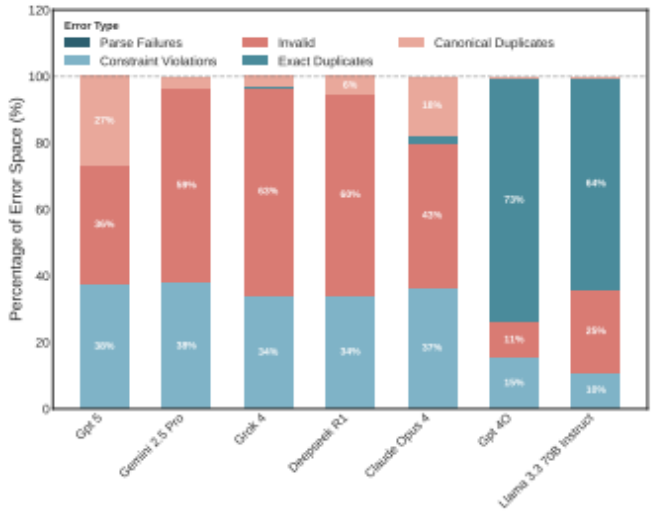
- The difficulty level depends on the nodes number n and intervention times m. Usually, causal graph problems with big n and small m will cause a high degree of underdetermination, which means a huge number of graphs will fit the small amount of experimental data. The admissible set $|\mathcal{H}_O|$ will be quite large in this scenario, making the LLM hard to capture all answers(low RR rate).

Experiment Analysis

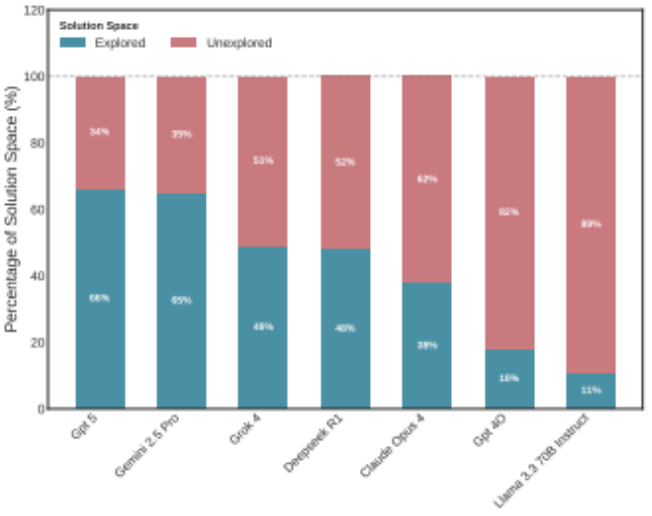
The essay varies the benchmark by changing the nodes number. Results demonstrate that causal inference benchmark doesn't pose to be difficult towards most SOTA LLM. Almost all top reasoning models remain robust throughout the benchmark and reach almost 100% correctness on the three indexes. In comparison, non-reasoning models face performance deterioration and the NR/RR indexes drastically collapse when more nodes are integrated.

		Reasoning Models					Non-Reasoning Models	
Difficulty	**Metric**	**gpt-5**	**gemini-2.5-pro**	**claude-opus-4**	**deepseek-r1**	**grok4**	**gpt-4o**	**llama-3.3-70b-instruct**
Task 1: Causal Inference								
1 (nodes=4)	*VR*	100.00% ± 0.00%	100.00% ± 0.00%	89.30% ± 21.70%	100.00% ± 0.00%	100.00% ± 0.00%	73.80% ± 32.70%	30.00% ± 40.10%
	NR	100.00% ± 0.00%	100.00% ± 0.00%	86.20% ± 20.80%	98.20% ± 4.50%	100.00% ± 0.00%	83.20% ± 23.00%	83.30% ± 27.00%
	RR	100.00% ± 0.00%	100.00% ± 0.00%	77.50% ± 27.10%	98.20% ± 4.50%	100.00% ± 0.00%	60.90% ± 34.50%	24.00% ± 34.40%
2 (nodes=5)	*VR*	100.00% ± 0.00%	100.00% ± 0.00%	80.10% ± 23.10%	99.00% ± 3.30%	100.00% ± 0.00%	49.00% ± 36.20%	17.60% ± 28.10%
	NR	100.00% ± 0.00%	100.00% ± 0.00%	79.90% ± 23.60%	93.50% ± 12.00%	100.00% ± 0.00%	83.50% ± 24.60%	86.00% ± 21.00%
	RR	100.00% ± 0.00%	100.00% ± 0.00%	65.40% ± 28.40%	92.50% ± 12.80%	100.00% ± 0.00%	38.30% ± 32.80%	17.60% ± 28.10%
3 (nodes=6)	*VR*	100.00% ± 0.00%	99.80% ± 0.80%	59.30% ± 21.20%	98.20% ± 3.00%	100.00% ± 0.00%	72.80% ± 34.10%	10.40% ± 26.00%
	NR	99.20% ± 1.40%	99.40 % ± 1.30%	90.00% ± 7.90%	81.20% ± 9.80%	99.80% ± 1.20%	22.90% ± 27.20%	38.00% ± 22.40%
	RR	99.20% ± 1.40%	99.20% ± 1.40%	51.10% ± 22.10%	79.50% ± 9.30%	99.80% ± 1.20%	6.70% ± 8.30%	2.30% ± 2.70%

Moreover, models fail at Recovery Rate (RR) not because they (the inference models) cannot generate the correct answer, but because they systematically miss most of the known correct answers. Even the top-performing GPT-5 and Gemini-2.5-Pro systematically missed 30%-40% of the known correct answers (Unexplored). When the collapse happens, the model is trapped in a few preferred templates. Instead of exploring new answers, they repeatedly submit semantically duplicate answers (canonical duplicates) or invalid guesses (invalid generations).



(a) Collapse Type Analysis



(b) Exploration Analysis

Optimization

Model Preparation

Model Selected: Openrouter - Qwen2.5 72B Instruct(Free)

Generate Datasets with Different Nodes:

```
python generate_causal_dataset.py --nodes 3 --seed 33550336 --output
"datasets/n3_observations.json"
python generate_causal_dataset.py --nodes 4 --seed 33550336 --output
"datasets/n4_observations.json"
```

Due to the high time lapse of benchmarks with more nodes, the report only covers the benchmark with 3 and 4 nodes.

Benchmark Command:

```
python run_causal_benchmark.py --dataset "datasets/n3_all_observations.json" --config
"config/config.yaml" --n-samples 30 --query-multiplier 1.0 --seed 33550336
```

Raw Outcome:

	Parse_success_rate	Valid_rate	Novelty_rate	Recovery_rate
Mean	1	0.604444	0.897222	0.540555
Std	0	0.382436	0.242176	0.383953
Var	0	0.146258	0.058649	0.147420
P_value	0	2.234953e-10	1.754505e-10	2.335179e-09

Understanding the Metrics:

Metric	Range	Good Score	Interpretation
🎯 Validity	0-1	> 0.90	Model proposes correct hypotheses
✨ Uniqueness	0-1	> 0.80	Model avoids redundant proposals
📈 Recovery	0-1	> 0.80	Model explores solution space well

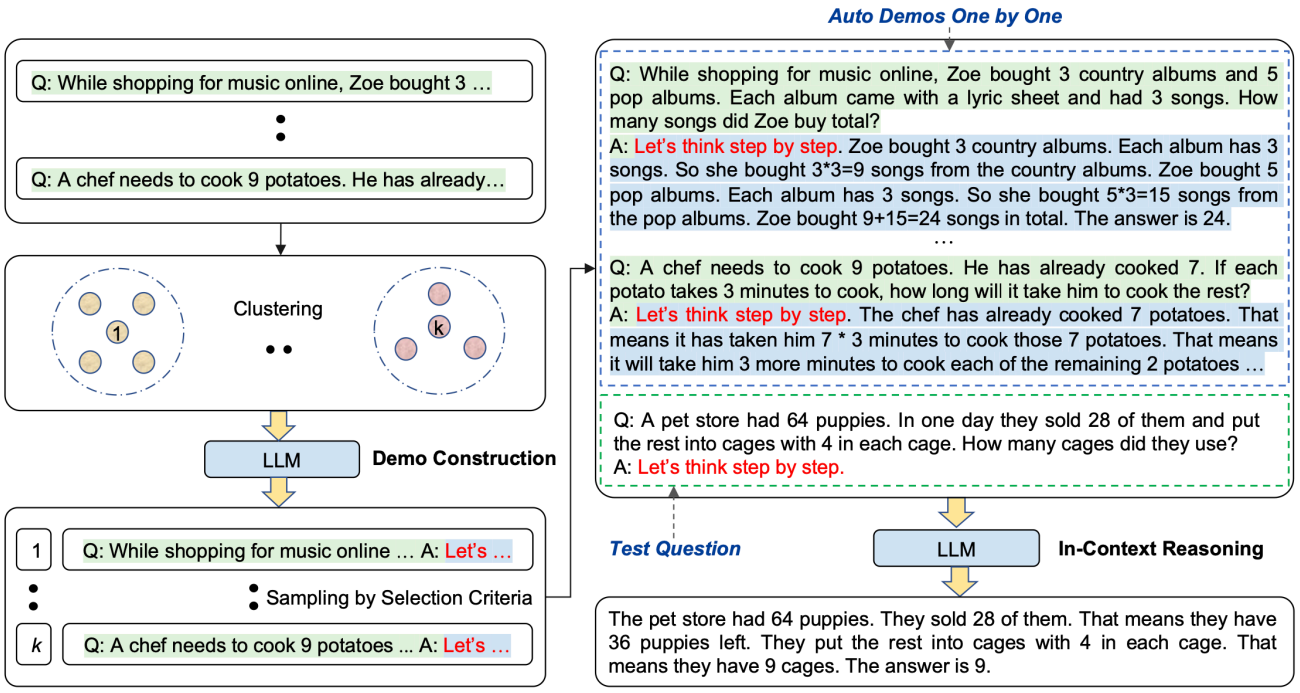
According to the author's point, Qwen2.5-72B-Instruct performs well only on the novelty rate of causal graph benchmark, scoring over 0.80.

Chain of Thought

Since the tested LLM can only be called through API input, I hope to propose a solution without altering the actual performance of the LLM itself, such as SFT or RL.

Chain-of-Thought (CoT) reasoning represents a breakthrough prompting technique that fundamentally transforms how Large Language Models handle complex reasoning tasks. This technique was originally introduced by Wei et al. from Google Research in their seminal 2022 paper "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." The essay raises that when the LLM is correctly guided to generate intermediate reasoning graphs, its performance can be greatly improved on sophisticated tasks such as arithmetic reasoning and inference. People tend to naturally split problems into multiple sub-problems to degrade the difficulty. CoT thus simulates the human cognitive process by explicitly requiring the model to

generate intermediate reasoning steps, effectively expanding the model's "working memory capacity." In the specific task of causal graph discovery, CoT's role is particularly prominent because causal reasoning is inherently a process requiring multi-step logical deduction.



In the actual solution, theCoT is put inside the prompt creation functions. Firstly, the precise definition of causal graph perturbation is given:

```
When we "Perturb X", the result shows:
1. X = 0 (the perturbed node is always 0)
2. Y = 1 if there exists a directed path FROM X TO Y in the graph
3. Y = 0 if there is NO path from X to Y

Paths are TRANSITIVE!
- If graph has X->Y->Z, then perturbing X affects both Y and Z
```

After that, the complete reasoning examples and corresponding logic chains are listed to enhance the LLM's memory on perturbation:

```
EXAMPLE (learn from this):

Nodes: A, B, C
Observations:
- Perturb A -> A=0, B=1, C=1
- Perturb B -> A=0, B=0, C=1
- Perturb C -> A=0, B=0, C=0

Step-by-step reasoning:
1. Valid edges: Obs 1 shows A reaches B and C...
2. Invalid edges: No node reaches A...
3. Candidate: A->B, B->C (this creates transitive path A->B->C)
4. Verify: [detailed verification for each observation]
5. Diversity: (no priors in this example)
```

```
FINAL ANSWER: Graph: A->B, B->C
```

Since the benchmark also includes the novelty index, which represents how many new answers the LLM can generate during the iteration process, we also hope that after ensuring higher success rate, more new answers can be generated.

```
REASONING STEPS (be concise, 3-5 sentences):
1. Identify valid edges: For each observation, which edges are required/possible?

2. Eliminate invalid edges: Which edges are ruled out by observations?

3. Generate candidate graph: Construct a graph that explains all observations.

4. VERIFY CORRECTNESS (CRITICAL):
   - Test your graph against EACH observation
   - Ensure perturbing each node produces the correct downstream effects
   - If ANY observation fails, revise your graph

5. Diversity check (IMPORTANT when priors exist):
   - Compare your graph's edges with ALL prior predictions above
   - If your graph matches any prior, actively seek an alternative valid graph
   - Apply diversity strategies: minimize edges / add optional edges / ...
   - Ensure your final graph has a DIFFERENT edge set from all priors
```

Moreover, it is easy to notice that the promising outcome from the LLM has a significant trend shifting from accuracy to novelty, and this trend generally progresses gradually with the increase in the number of rounds of reasoning. However, this trend contradicts the cognitive inertia of large models. In causal graph discovery tasks, this phenomenon is particularly prominent because typically multiple equivalent or approximately equivalent causal graphs can explain the same set of observational data, yet the model often becomes overly reliant on the first solution it finds.

To relieve this issue, an adaptive diversity emphasis system is implemented. The core idea of this system is: as the number of hypotheses generated by the model increases, the system progressively elevates the importance of "exploring new solutions." This design draws on the exploration-exploitation trade-off theory from reinforcement learning, gradually increasing the reward for diversity while ensuring solution validity. As the model generates more and more hypotheses, the risk of repeating previous solutions also increases. Dynamic diversity emphasizes progressively increasing diversity requirements based on the number of hypotheses generated.

```
# Original: Static prompt
prior_block = "\n".join(prior_lines)

# Enhanced: Dynamic emphasis
if prior_hypotheses:
    n_priors = len(prior_hypotheses)
    if n_priors >= 4:
        diversity_emphasis = "\n\nIMPORTANT: You have tried many graphs already. " \
                             "PRIORITIZE finding a DIFFERENT valid solution!"
    elif n_priors >= 2:
        diversity_emphasis = "\n\nNote: Try to find a different valid graph " \
                             "from the ones above."
```

```
    else:
        diversity_emphasis = ""
```

Prior Count	Emphasis Level	Effect
0-1	No special emphasis	Focus on accuracy
2-3	Gentle reminder	Encourage exploration
≥4	Strong requirement	Prioritize diversity

Validation

In the original version, invalid hypotheses generated by the LLM were directly discarded without opportunities for self-correction. This led to resource waste and missed learning opportunities. Therefore we add the iterative validation to deal with the potential failures.

```
for validation_round in range(max_validation_retries + 1):
    # First round: Normal prompt
    if validation_round > 0 and self.use_cot:
        # Subsequent rounds: Add validation feedback
        feedback_prompt = base_prompt + f"""

        FEEDBACK ON YOUR PREVIOUS ATTEMPT:
        Your previous graph was INCORRECT. Here's what went wrong:
        {validation_feedback}

        Please revise your reasoning and provide a CORRECT graph...
        """
        prompt = dedent(feedback_prompt).strip()
    else:
        prompt = base_prompt

    # Query LLM
    hypothesis = self.parse_llm_response(response)

    if hypothesis:
        is_valid = self.validate_hypothesis(hypothesis, observations)
        if is_valid:
            break  #Success!
        elif self.use_cot and validation_round < max_validation_retries:
            #Generate feedback and continue
            validation_feedback = self.generate_validation_feedback(
                hypothesis, observations
            )
            validation_attempts += 1
            continue
```

Dynamic Temperature

Temperature parameters control the output randomness of LLM:

- **Low temperature (0.0-0.3)**: Deterministic output, selects highest probability tokens
- **Medium temperature (0.4-0.7)**: Balances accuracy and diversity
- **High temperature (0.8-1.0)**: Highly random, increases creativity

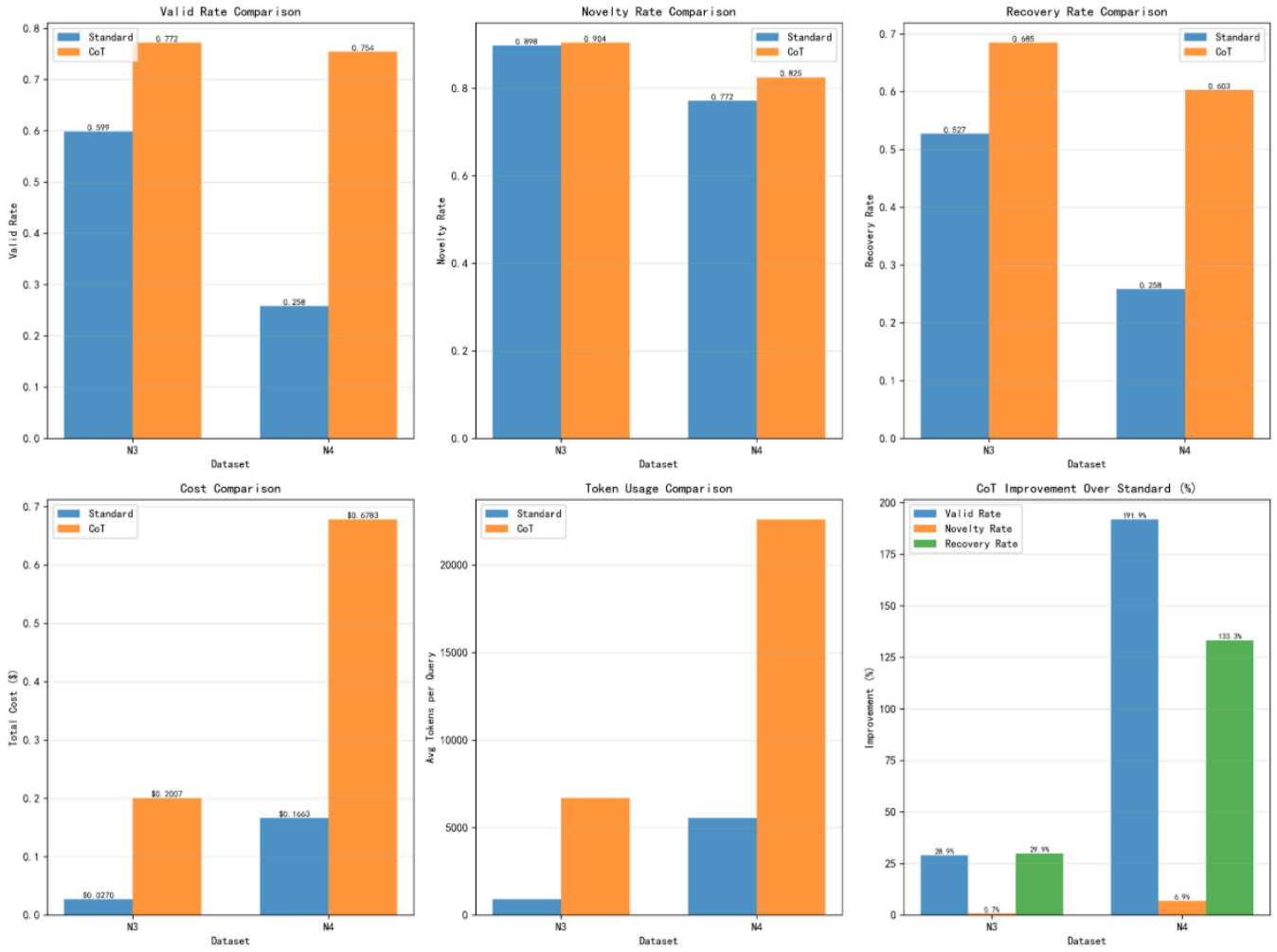
It is clear that the temperature setting is integrated with the tendency of accuracy versus novelty mentioned above. So a dynamic temperature adjustment can help correspond to the shifting trend, which improves both accuracy and novelty.

```
# temperature = 0.7 (constant)
original_temp = llm.temperature if hasattr(llm, 'temperature') else 0.7

for i in range(n_queries):
    if self.use_cot and hasattr(llm, 'temperature'):
        progress = i / n_queries

        # First 40%: Keep original temperature (accuracy focus)
        if progress < 0.4:
            llm.temperature = original_temp

        # Middle 30%: Increase by 50% (transition)
        elif progress < 0.7:
            llm.temperature = min(original_temp * 1.5, 1.0)

        # Last 30%: Increase by 100% (diversity boost)
        else:
            llm.temperature = min(original_temp * 2.0, 1.0)
```

Results Analysis

The improved script can generate the comparison benchmark of(Original/CoT) LLM mode on both n3/n4 datasets. And the output data will be saved in the .csv file together with the graph.

The overall comparison results are illustrated below:



The chart compares different approaches and vividly shows how the combined strategies-CoT, dynamic diversity, iterative validation, and dynamic temperature-further improved the performance of the model on the causal inference benchmark for both 3-node and 4-node datasets (n3_all and n4_all).

On the simpler 3-node task, the CoT-enhanced model achieves almost perfect scores on all three key metrics: Validity, Novelty, and Recovery, reaching 100% (1.0). In contrast, the original model struggled significantly with Validity at around 0.6 and Recovery at around 0.55, although its Novelty already stood quite high at about 0.9.

However, the improvements are even more striking on the significantly harder 4-node dataset, which, according to the document, increases both the underdetermination and difficulty of the problem. While the original model performance collapsed especially in Validity at around 0.4 and Recovery at about 0.35, the optimized "CoT" performed strongly. Its Validity rate jumped to about 0.9, with more than a doubling in its Recovery Rate to about 0.8.

Summary

In summary, these optimizations together address the key challenges: CoT and iterative validation improve the VR or accuracy of answers, while dynamic diversity emphasis and dynamic temperature push the model out of its comfort zone with large gains in NR and RR.

However, this large gain in performance does not come for free. As can be seen from the "token_usage" metric in the chart, the CoT version consumes far more tokens than its original counterpart for both 3-node and 4-node tasks. This result is fully expected, since Chain-of-Thought prompts are longer and involve an explanation and examples of how to approach a problem. If added to potential iterative validation and feedback loops, each query needs more context; hence, token costs increase significantly. This result, therefore, illustrates a

tradeoff: greater robustness and more comprehensive exploration in complex reasoning tasks are achieved at the cost of more sophisticated prompt engineering and higher computational overhead.