GRPO on Graph Isomorphism: A Cold-Start Problem

Homework 2 — Research in LLMs, HSE × Central University

Ivan A. Novosad
inovosad@hse.ru

1. The Bet

I could have picked graph coloring or some other task where GRPO is known to work on a 1.5B model. But this is a research course, and repeating what has already been done many times did not seem like a good use of the opportunity. Then I came across the G1 paper (Guo et al., 2025), which applied GRPO to 50 graph reasoning tasks on Qwen2.5-3B and pushed aggregate accuracy from 22.7% to 59.8%. They classified isomorphic mapping as "Challenging" with near-zero base accuracy even at 3B and 7B, and solved it by warming up with SFT on reasoning traces rejection-sampled from Qwen2.5-32B, using 4096-token context on eight A800 GPUs. They never tried isomorphism on 1.5B. So I decided: who if not me, and where if not here.

Verification is cheap (NetworkX checks a mapping in polynomial time), you can generate unlimited training pairs on the fly, and difficulty scales with node count, giving ten levels from three-node graphs to twelve-node graphs. Nothing in pretraining teaches a 1.5B model to recover edge-preserving bijections from adjacency-list tokens. The 1024-token context limits chain-of-thought length. I found no published result of a model this small learning graph isomorphism through RL alone.

My thesis was specific: tasks with asymmetric subtask difficulty should collapse under vanilla GRPO. Predicting "NOT ISOMORPHIC" costs two tokens. Predicting "ISOMORPHIC" requires proposing a full node mapping and getting every edge right. I designed an anti-hacking stack to test whether this collapse could be prevented: class-aware asymmetric rewards (iso correct $+1.5$, iso wrong -1.0, non-iso correct $+0.5$, non-iso wrong -0.5), 75/25 isomorphic-heavy training composition, and batch-level advantage normalization to amplify even small within-batch reward differences.

The plan was five runs on an RTX 4090 within a sixteen GPU-hour budget: anti-hack versus vanilla versus vanilla-plus-batch-normalization-only, with extensions at harder difficulties. In practice, roughly a third of that budget went to fighting the infrastructure stack. FlashInfer refused to compile (missing nvcc, then linker errors requiring manual CUDA stub symlinks). vLLM crashed silently on standby until I found the `UNSLOTH_VLLM_STANDBY=0` workaround. Trl callbacks crashed on extra fields from unsloth's trainer. Stitching together vLLM + Unsloth + FlashAttention + trl is not something I can set up in a day. Between that and the evaluation bug (Section 6), the planned comparison never happened, but for a different reason than I expected.

2. Environment and Verification

Graph isomorphism task. Each instance presents two graphs in adjacency-list format with 1-indexed nodes. Difficulty levels 1 through 10 map to 3 through 12 nodes, with edge probability and non-isomorphic perturbation strength scaling with difficulty. The model must produce a `<think>...</think>` block followed by either a node mapping or "NOT ISOMORPHIC".

The verifier and test generation pipeline were implemented with Claude Code. I validated correctness through tests: the verifier accepts any valid mapping format (arrow, colon, or positional) and checks all edges exhaustively; isomorphism is not unique, so any valid mapping is accepted, not just the canonical one.

54 frozen test sets totaling roughly 10k instances were generated before training (also via Claude) and uploaded to HuggingFace. Each difficulty level has separate iso-only, non-iso-only (at three hardness tiers), and mixed evaluation sets. The eval sets use a 67/33 iso/non-iso split; the 75/25 ratio described in Section 1 applies only to the training data composition.

Edge counting control. As a positive control I implemented a second environment using the same graph format but with a simpler task: count the number of edges. The reward combines binary correctness (exact integer match: +1 correct,-1 wrong, weighted 0.7) with a format bonus (+0.5 for proper think/answer tags, weighted 0.3), giving a total range of $[-0.7, +0.85]$. The base model gets 40% accuracy

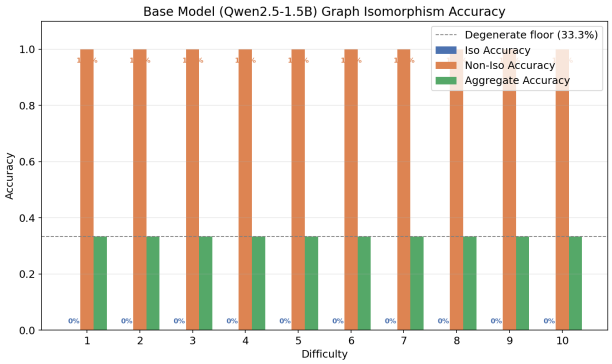

Figure 1. Base Qwen2.5-1.5B accuracy across difficulties 1–10. Iso accuracy is 0% at every level (blue bars at baseline); non-iso accuracy is 100% everywhere (orange bars). Aggregate accuracy sits exactly at the 33.3% degenerate floor.

Figure 2. Class Prediction Ratio (CPR = fraction of "NOT ISO" predictions) during training for three GI runs. Run 1 (red) stays pinned near CPR = 1.0. Run H1 (blue) fluctuates widely, reaching 0.28 at step 103. Run S1 (green) sits between the two. Dashed lines at 1.0 and 0.5 mark the "always NOT ISO" and balanced baselines.

at difficulty 1, which matters for everything that follows.

3. Base Model — The Zero Wall

Qwen2.5-1.5B-Instruct, tested on 4,500 graph isomorphism instances (450 per difficulty, each containing 300 isomorphic and 150 non-isomorphic pairs), produces the same 2-token response for every input: "NOT ISOMORPHIC." The numbers are uniform across all ten difficulty levels (Figure 1): 0% iso accuracy, 100% non-iso accuracy, 33.3% aggregate, exactly the degenerate floor. I define the **class prediction ratio (CPR)** as the fraction of predictions that are "NOT ISO"; it is 1.0 at every difficulty. Format compliance is 0%. Mean response length is 2 tokens.

I expected some iso accuracy at difficulty 1, three nodes, two edges. In practice it was zero everywhere. The model does not attempt mappings.

This is the *cold-start problem*: all $G = 8$ completions are identical, $\sigma = 0$, GRPO produces no gradient.

4. Three Strategies That Failed

A note on evaluation: the assignment requests paired bar charts comparing base and trained model accuracy across difficulties. The evaluation bug described in Section 6 rendered all trained-model evaluations invalid, so no such comparison is possible. Everything that follows relies on training dynamics (CPR traces, gradient norms, reward curves, adapter weight analysis), not on post-training accuracy.

4.1. Run 1: Anti-Hack GRPO

If "NOT ISO" predictions pay less than "ISO" predictions, the model should drift toward the harder subtask. That was the idea behind the anti-hack stack: asymmetric rewards, iso-heavy data, batch-level normalization. I ran with $G = 8$, standard GRPO loss, and $\beta = 0.0$ (I had planned DAPO with $\beta = 0.04$ but fell back to standard GRPO after Unsloth compatibility issues).

CPR stayed between roughly 0.85 and 1.0 for all 215 steps. Roughly 35% of batches had zero reward variance across all prompt groups, meaning every completion was identical and the gradient was exactly zero. Reward shaping cannot create signal when the model never produces the output that would collect the higher payoff. Batch normalization amplifies relative differences, but here every completion is identical.

In HW1 I found that RLOO with $K = 2$ dominated $K = 24$ because update frequency beats baseline quality. Same principle here, except its prerequisite (nonzero within-group diversity) is missing.

4.2. Run H1: Hints

If the bottleneck is completion diversity, I need the model to produce varied outputs. At difficulty 3 (five nodes), I revealed $k = n - 2 = 3$ of the $n = 5$ mappings in the prompt, reducing the problem to a binary choice for the remaining nodes.

The CPR trace (Figure 2) shows this partially worked: CPR dropped to 0.28 at step 103, meaning the model sometimes predicted ISO. When I saw that, I thought GRPO was finally getting signal. Then I checked the adapter weights. All 196 LoRA-B matrices had norm below 0.01; gradient norm was 0.0 for all 200 steps. The CPR variation came from sampling diversity in the base model: with strong hints, it

sometimes generates mapping-like text by chance. None of that reached the adapter.

Sampling diversity is not learning.

4.3. Run S1: SFT Warmup → GRPO

If the adapter needs to be alive before GRPO can work, SFT should activate it. I generated 50 chain-of-thought examples using Claude Code, each with step-by-step edge verification ("edge $(1, 2)$ in G1 maps to edge $(3, 1)$ in G2, check…"). Writing the prompt template and validating the examples took most of an evening. SFT ran for 20 steps. Post-SFT, all 196 LoRA-B matrices had nonzero weights, the first time any run produced an active adapter.

The GRPO phase ran for 200 more steps. CPR was 0.75–1.0, mean reward around -0.8 under the class-aware scheme, gradient norm 0.0 for every step. The adapter that SFT activated went silent the moment GRPO began. Fifty examples taught formatting but not graph isomorphism. The model could produce nicely structured wrong answers, and GRPO cannot refine a policy that never stumbles onto a correct one. G1 used rejection-sampled traces from a 32B teacher; I had no such teacher.

5. Edge Counting — The Positive Control

To separate task-level failure from code-level failure, I ran edge counting with the same Unsloth + LoRA + GRPO setup. The only differences: the task (count edges), the reward, and $G = 16$ instead of $G = 8$ (edge counting completions are shorter, so more fit in memory).

The base model produces diverse completions here: varied numeric answers, some right, some wrong, never identical across all completions. Reward standard deviation was nonzero for all 500 steps, so GRPO always had signal. The adapter was alive (194/196 LoRA-B matrices active). This is what working GRPO looks like, and it confirmed that the cold-start problem was about the task, not the code.

But the run diverged. Figure 3 shows the reward trajectory: an initial rise, then steady decline. Figure 4 shows what was happening underneath: loss and gradient norms grew by four orders of magnitude over 500 steps.

This happened because $\beta = 0.0$. Without the KL term, nothing anchors the policy to the reference model. I expect $\beta = 0.04$ fixes this, but I identified the cause after the GPU budget ran out.

The evaluation bug (Section 6) prevented measuring whether early checkpoints had improved accuracy. The reward curve suggests they might have.

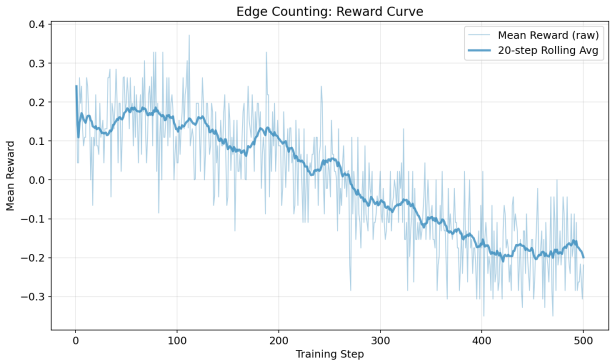

Figure 3. Edge counting reward over 500 training steps. The 20-step rolling average (dark line) shows an initial rise toward $+0.25$, then a steady decline to approximately -0.15 as the policy diverges.

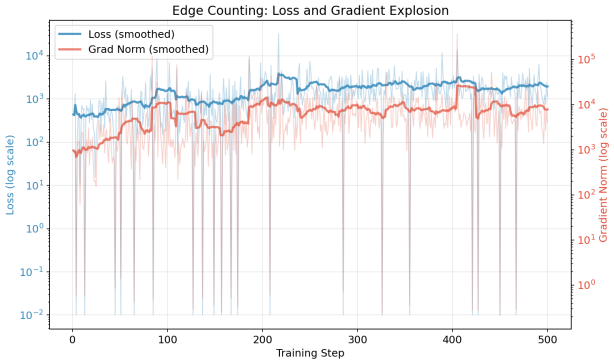

Figure 4. Edge counting loss (left axis, log scale) and gradient norm (right axis, log scale) over 500 training steps. Both grow by four orders of magnitude. Root cause: $\beta = 0.0$ (no KL penalty).

6. The Eval Bug

Symptom. After all runs, I evaluated every adapter on the frozen test sets. The edge counting results came back first: 40%, 56%, 8% at difficulties 1, 2, 3. Then I evaluated a different checkpoint. Same numbers. And another. Same again. I spent several hours assuming the training genuinely produced no improvement before noticing that even the base model evaluation gave exactly the same figures down to the last decimal (Table 1). That is when I stopped looking at the models and started looking at the evaluation code.

Diagnosis. The eval scripts loaded the adapter path but never applied the LoRA weights on top of the base model. The correct sequence is: load base, apply adapter via PEFT, fuse for inference. Without the middle step, every evaluation silently runs against the naked base model. I confirmed the fix by manually loading and fusing one edge counting

Table 1. Evaluation results across adapters and base model. All rows are identical, proving the adapter was never loaded during inference.

GI Eval Dir	Iso Acc.	Non-Iso Acc.	CPR
base	0.0%	100.0%	1.0
run_h1	0.0%	100.0%	1.0
run_s1_grpo	0.0%	100.0%	1.0
EC Eval Dir	**Accuracy (Diff 1 / 2 / 3)**		
ec_base	40.0% / 56.0% / 8.0%		
ec_trained	40.0% / 56.0% / 8.0%		
ec_trained_v2	40.0% / 56.0% / 8.0%		
ec_ckpt150	40.0% / 56.0% / 8.0%		

adapter, which produced outputs different from the base model, but had no remaining GPU budget to re-evaluate systematically.

Impact. All post-training evaluation results are invalid. Every quantitative claim in this report relies on training dynamics (CPR traces, gradient norms, reward curves, adapter weight analysis), not eval accuracy.

7. Discussion

GRPO assumes the base model sometimes gets the answer right. DeepSeek-R1 (AI, 2025) and G1 both start from that assumption. When base accuracy is zero and every completion is the same two tokens, none of that machinery helps.

The failure that surprised me most was Run H1. I expected that diverse outputs would be enough for GRPO to work. They were not: the outputs were diverse but the adapter was dead. All three things — reward signal, output diversity, and a live adapter — have to work at the same time. Each run fixed one of the three and left the others broken.

The edge counting control (Section 5) confirmed this is about the task: 40% base accuracy on edge counting versus 0% on isomorphism is the difference between a working RL loop and a dead one.

The most promising fix is a strong SFT warmup. Run S1 showed SFT activates the adapter, but fifty examples only taught formatting. Hundreds of correct reasoning traces are needed to also teach the task. I would rejection-sample them from a 32B-class teacher, the way G1 did. A curriculum from edge counting through subgraph matching to full isomorphism is the other direction I would try. Either way, β is non-negotiable – edge counting proved that.

DOTS (Liu et al., 2025) assumes there exists a difficulty level where the model gets some answers right. This experiment has no such level. The emerging work on training with "unsolvable" tasks (LUFFY, RL-PLUS, SRFT) is designed for exactly this situation, and that is where HW3 picks up.

8. Conclusion

GRPO cannot bootstrap learning from a fully collapsed starting policy, and graph isomorphism on Qwen2.5-1.5B is precisely such a case. Three training runs and one positive control characterized the cold-start problem, showed that reward shaping and prompt-based diversity are not sufficient on their own, and pointed to SFT warmup with a capable teacher as the most direct path forward.

Artifacts. Adapters and test sets are on HuggingFace:

- SFT adapter: https://huggingface.co/Melodiz/qwen2.5-1.5b-gi-sft
- GRPO adapter: https://huggingface.co/Melodiz/qwen2.5-1.5b-gi-grpo-antihack
- Test sets: https://huggingface.co/datasets/Melodiz/graph-isomorphism-test-sets

HW3 asks for training on "unsolvable" tasks (pass@128 $= 0$), exactly the regime characterized here.

References

AI, D. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Guo, X., Li, A., Wang, Y., Jegelka, S., and Wang, Y. G1: Teaching llms to reason on graphs with reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18499.

Liu, T., Zhao, Z., Chen, J., et al. Dots: Learning to reason dynamically in llms via optimal reasoning trajectories search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5e5d6f9ac33ba9349ba7b2be9f21bad9-Paper-Conference.pdf.