GROVE: Graph-based Routing Over Variant Experts for Adaptive Two-Stage Training of Reasoning Models

Yiming Luo[1,2] **Chunlin Tian**[2,3]* **Li Li**[2]

[1]University of Michigan, Ann Arbor
[2]University of Macau
[3]National University of Singapore

ymluo@umich.edu tianclin0212@gmail.com llili@um.edu.mo

Abstract

Post-training reasoning models is computationally expensive, and the allocation between supervised fine-tuning (SFT) and reinforcement learning (RL) substantially affects reasoning quality. Existing methods operate at the training-procedure level and treat the reasoning corpus as homogeneous, ignoring the structural heterogeneity of individual traces. We propose **GROVE** (**G**raph-based **R**outing **O**ver **V**ariant **E**xperts), an adaptive framework that routes each example to SFT or RL based on the structural topology of its reasoning trace. GROVE parses each trace into a *Reasoning Tree*, scores its complexity via graph entropy, and partitions the corpus through a data-driven Gaussian Mixture Model with no hand-tuned threshold. Low-complexity traces are trained with SFT and high-complexity traces with GRPO as parallel LoRA adapters from a shared frozen base, merged into a single deployable model with no inference-time overhead. Across two training corpora and five reasoning benchmarks, GROVE consistently outperforms single-stage and naive-hybrid baselines, and a reversed-routing ablation confirms that this mapping is intrinsic rather than arbitrary.

1 Introduction

Large language models (LLMs) have made rapid progress on general natural language tasks, yet they remain unreliable on problems that demand deep, multi-step reasoning—competition mathematics, graduate-level science, and long-chain logical inference (Chen et al., 2026; Feng et al., 2025; Guo et al., 2025b; Yi et al., 2025; Aggarwal and Welleck, 2025). To close this gap, a class of *reasoning models*, exemplified by DeepSeek-R1 (Guo et al., 2025a), Qwen3 (Yang et al., 2025) and OpenAI-o1 (OpenAI et al., 2024) has emerged: these systems are post-trained to produce extended chain-of-thought (CoT) traces that expose intermediate steps, case analysis, and self-correction before any final answer is committed. Reasoning models have rapidly become the default choice for hard reasoning tasks, but their post-training is computationally expensive and their accuracy remains uneven across superficially similar problems. The practical question is therefore how to extract more reasoning capability from a fixed post-training budget.

Existing methods to post-train reasoning models form three broad families. *Reinforcement-learning-based methods*, including PPO and InstructGPT (Schulman et al., 2017; Ouyang et al., 2022) and Direct Preference Optimization (DPO) (Rafailov et al., 2024), build more robust training pipelines through higher-quality preference data and explicit or implicit reward signals. *Self-training methods*, exemplified by STaR (Zelikman et al., 2022), LMSI (Huang et al., 2022), and ReST$^{\mathrm{EM}}$ (Singh et al., 2024), let the model bootstrap its own reasoning capability from its correctly answered generations, removing the reliance on external human annotation. A third family controls the *difficulty composition* of the reasoning data itself, switching between System-1 (concise) and System-2 (elaborate) CoT data and re-weighting tokens by reasoning depth to balance training cost against reasoning quality, as in TL;DR (Li et al., 2025), Kimi k1.5 (Team et al., 2025), and LS-Mixture SFT (Yu et al., 2025).

Each family, however, leaves key cost or reliability issues unresolved. RLHF-style methods such as PPO and InstructGPT (Schulman et al., 2017; Ouyang et al., 2022) depend on an explicit reward model, while DPO (Rafailov et al., 2024) avoids this only by requiring large-scale paired preference data; both incur substantial annotation and compute costs. Self-training methods such as STaR (Zelikman et al., 2022) reduce annotation needs but suffer from self-confirmation bias (Wu et al., 2025) and require reliable answer verification, limiting their applicability to domains with clear ground

* Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 28210–28224
October 24-29, 2026 ©2026 Association for Computational Linguistics

truth. Difficulty-control methods lack a principled allocation rule: Kimi k1.5 (Team et al., 2025) uses a fixed System-1/System-2 ratio, whereas TL;DR (Li et al., 2025) requires additional SFT and RL runs to set performance bounds.

Hard Cost: Quality Trade-off. Existing methods fail to reconcile training cost with reasoning quality. Reward modeling, iterative self-training, and extra SFT/RL calibration all increase the cost of post-training long-CoT reasoning models, leaving stable and efficient training under a fixed compute budget unresolved. *Blind to the Structure of Reasoning Data.* These methods also ignore the intrinsic structure of reasoning data. By operating only through policy optimization, self-distillation, or schedule design, they treat the corpus as homogeneous and flatten the distinction between linear procedural traces and branched exploratory ones. They therefore never decide which examples should be learned by which training paradigm. A better solution should be structure-aware, require no external verifier or reward model, and avoid extra independent training stages.

We propose **GROVE** (**G**raph-based **R**outing **O**ver **V**ariant **E**xperts), a dynamic post-training framework that routes each example to a training paradigm according to the structural complexity of its reasoning trace. As shown in Figure 2, GROVE parses each trace into a *Reasoning Tree*, computes a graph-entropy complexity score, and partitions the corpus with a two-component Gaussian Mixture Model. Low-complexity traces are trained with SFT, whereas high-complexity traces are trained with RL via GRPO (Shao et al., 2024); both are implemented as independent LoRA adapters on the same frozen base and merged into a single model, introducing no inference-time overhead.

GROVE directly addresses the two limitations above. Its Reasoning-Tree-based routing signal is intrinsic to the data, making training sensitive to the structure of reasoning traces. Meanwhile, LoRA limits optimization to low-rank updates, and GRPO avoids a separate reward model, reducing post-training cost while preserving reasoning quality. The GMM partition further removes the need for hand-tuned SFT/RL thresholds, yielding a principled and data-driven allocation. Compared with single-stage SFT/RL and naive-hybrid baselines, GROVE achieves dynamic per-sample SFT/RL allocation based on trace structure. Experiments across five reasoning benchmarks show strong overall performance. Our contributions are summarized as follows:

- **A data-centric post-training framework.** GROVE reframes SFT-versus-RL selection as per-sample routing based on intrinsic trace structure, requiring no human preference labels, reward model, or external verifier.

- **A graph-entropy routing signal.** We introduce the Reasoning Tree and a parameter-free, corpus-normalized complexity score, combined with GMM-based partitioning and parallel LoRA composition with no inference-time overhead.

- **Evidence of structure–paradigm compatibility.** Across two corpora and five benchmarks, GROVE outperforms single-stage and naive-hybrid baselines on average. Reversed routing reduces performance by 7.11 points on s1K and 2.66 points on Open-RS, indicating that the routing captures a genuine structural property of the data.

2 Related Work

2.1 Reasoning Data in Reasoning Model Training

The data used to post-train reasoning models differs in kind from the prompt–response pairs that dominate general instruction tuning: each example pairs a problem with a long-form solution trace exposing every intermediate step, including case analysis, hypothesis exploration, and self-correction. This step-by-step structure is the central mechanism by which reasoning capability is elicited, enabling the model to learn the underlying reasoning procedure rather than just the input–output mapping and supporting generalization to problems beyond the training distribution. State-of-the-art reasoning models accordingly rely on extended chain-of-thought supervision as the core of their post-training (Muennighoff et al., 2025; Dang and Ngo, 2026; Guha et al., 2025; Ye et al., 2025; Fein-Ashley et al., 2025).

Less appreciated, however, is that reasoning traces are themselves structurally heterogeneous. As Figure 1 illustrates, a problem may admit either a linear, procedural solution or a branched, exploratory one that alternates between strategies and backtracks from dead ends. Current post-training pipelines flatten this heterogeneity by treating the

corpus as homogeneous demonstration data, leaving the structural signal already encoded in the traces unused.

2.2 Supervised Fine-Tuning vs Reinforcement Learning for Reasoning

Two paradigms dominate the post-training of reasoning models. Supervised fine-tuning (SFT) optimizes a token-level cross-entropy on reference traces, an efficient and effective approach for transferring reasoning behaviors. However, a growing body of work argues that maximum-likelihood imitation fits the trace surface rather than the underlying reasoning policy, leaving SFT-trained models memorization-prone and less reliable than RL-trained counterparts under distributional shift (Chu et al., 2025; Cai et al., 2025; Chen et al., 2025a).

Reinforcement learning (RL) takes the opposing approach: it optimizes a trajectory-level reward—typically the verifiable correctness of the final answer—that is agnostic to the specific token sequence (Chen et al., 2025b; Lyu et al., 2025). This signal encourages the policy to discover multiple viable paths and recover from dead ends, and is widely credited as the source of the generalization advantage of recent reasoning models, at the cost of higher compute and reliance on verifiable rewards (Shao et al., 2024; Qiu et al., 2025). This question motivates our work: by recovering the structural topology of each trace and routing it accordingly, GROVE reframes the SFT-versus-RL allocation problem from a scheduling decision into a data-driven one, turning structural heterogeneity into an actionable training signal.

2.3 Parameter-Efficient Fine-Tuning

The compute cost of post-training reasoning models has driven wide adoption of parameter-efficient fine-tuning (PEFT). Among the PEFT family—adapter modules, prefix tuning, and prompt tuning—LoRA has become the de facto standard for reasoning post-training, injecting low-rank updates into selected projections of every transformer layer while leaving the backbone frozen (Hu et al., 2021; Tian et al., 2024; Mao et al., 2024).

Two properties of LoRA are essential to GROVE. First, its small parameter footprint allows two adapters—one per paradigm—to be trained independently from the same frozen backbone within a fixed compute budget. Second, the adapters can be merged additively into the base weights, producing a single deployable model with no inference-time overhead. The recent Tina (Wang et al., 2025) recipe further demonstrates that GRPO can be applied effectively to LoRA-parameterized policies, establishing the empirical viability of GROVE's high-complexity RL branch.

3 Methodology

3.1 Overview

GROVE consists of three components that together resolve the SFT-versus-RL allocation problem entirely at training time. Given a training corpus $\mathcal{D} = \{x_i\}_{i=1}^{N}$ of reasoning traces and a frozen base model M_0, the pipeline produces a single deployable model M_{GROVE} in three stages, as illustrated in Figure 2. **(C1)** Each trace x_i is parsed into a *Reasoning Tree* T_i, from which a scalar structural-complexity score $\tilde{Q}_C(x_i) \in [0, 1]$ is computed via graph entropy (Section 3.2). **(C2)** The empirical distribution $\{\tilde{Q}_C(x_i)\}_{i=1}^{N}$ is partitioned into a low- and a high-complexity subset by a two-component Gaussian Mixture Model (Section 3.3). **(C3)** The low-complexity subset is fine-tuned with SFT and the high-complexity subset with RL, both as LoRA (Hu et al., 2021) adapters Δ_{SFT} and Δ_{RL} trained independently from M_0, and merged in parallel into M_{GROVE} (Section 3.4). The pipeline requires no human annotation, no reward model, no answer verifier, and no hand-tuned threshold; the only quantities estimated from data are the GMM parameters $\boldsymbol{\theta}_{\text{GMM}} = \{\pi_k, \mu_k, \sigma_k^2\}_{k=0}^{1}$.

3.2 Reasoning Tree and Structural Complexity Score

To make the structural topology of each trace operational, GROVE first recovers it as a rooted Reasoning Tree $T_i = (V_i, E_i)$ and then summarizes the tree by a single scalar score $\tilde{Q}_C(x_i)$.

Reasoning Tree construction. Each trace is parsed in three steps. **(i) Hierarchical segmentation.** A macro layer splits the trace into strategic blocks, each corresponding to one high-level approach pursued by the model; a micro layer further splits each block into finer tactical steps, yielding a segment sequence $\mathcal{U} = (u_1, \ldots, u_n)$ whose elements become the candidate nodes of T_i. **(ii) Functional tagging.** A deterministic rule-based parser labels each segment with one of (Action,Hypothesis,Derivation, Conclusion,Dead_End), marking respectively a new line of attack, a case assumption, a calculation or logical inference, an intermediate result, and

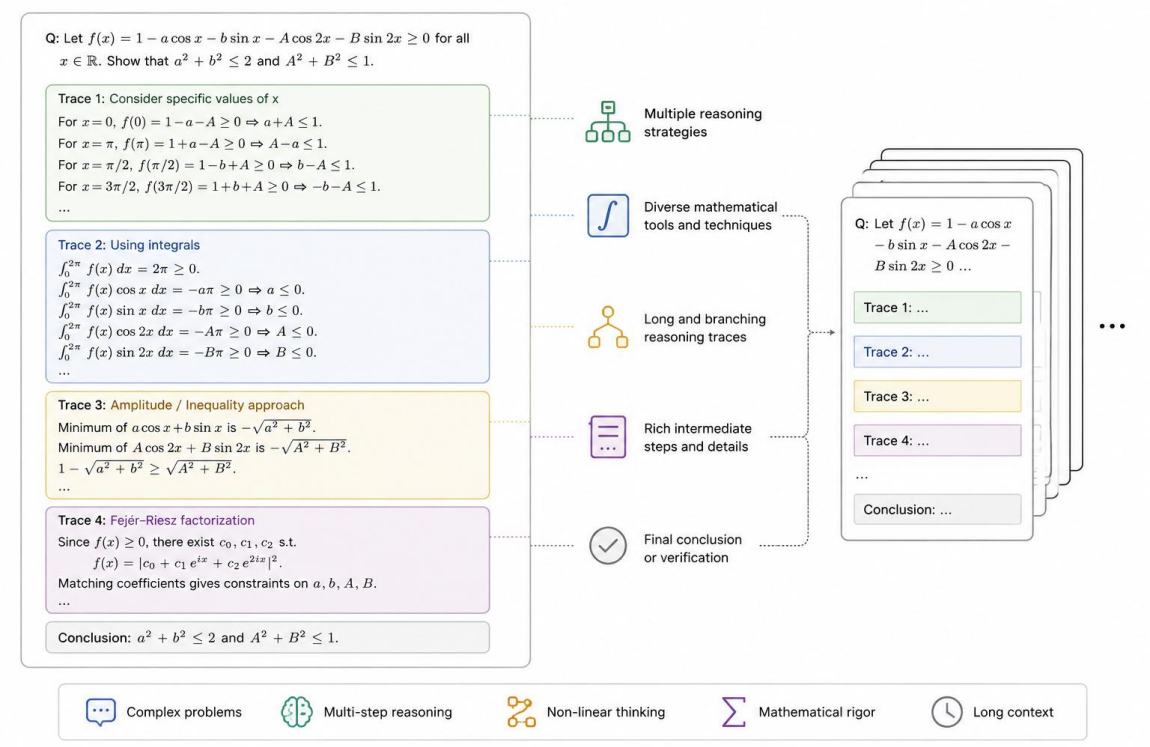


Figure 1: Characteristic of Reasoning data.

an attempt the model itself abandons. **(iii) Stack-based construction.** The tagged segments are assembled into a rooted tree by a single left-to-right pass that maintains a stack of open branches: *Action* or *Hypothesis* pushes (open a new branch), *Derivation* or *Conclusion* continues the current branch, and *Dead_End* pops. A direct consequence is that an *Action* immediately following a *Dead_End* attaches to a higher ancestor rather than to the abandoned branch, explicitly representing the abandonment of one strategy and the start of a fresh one as sibling subtrees. Figure 3 shows an illustrative example.

Graph-entropy complexity score. With the reasoning process modeled as a tree $T = (V, E)$, we quantify its structural complexity by the Shannon entropy of the tree's normalized node-degree distribution:

$$Q_C(T) = -\sum_{v \in V} p(v)\,\log_2 p(v), \qquad p(v) = \frac{\deg(v)}{\sum_{v' \in V} \deg(v')}. \tag{1}$$

Q_C increases with the size of the parsed tree and decreases with the concentration of its degree distribution. Linear procedural traces parse into small trees and therefore score low, whereas branched exploratory traces, which alternate between strategies and backtrack from dead ends, parse into substantially larger trees and score high. We note that the size term dominates in practice; Q_C should therefore be read as a measure of the elaboration of a trace's parsed structure rather than of branching alone. To make scores comparable across corpora, we min-max normalize Q_C within each training dataset to obtain $\tilde{Q}_C(x_i) \in [0, 1]$.

3.3 Data-Driven Partitioning via Gaussian Mixture Modeling

To partition $\mathcal{D}$ into the SFT and RL subsets, we fit a two-component Gaussian Mixture Model (GMM) on the normalized scores $\tilde{Q}_C$ rather than relying on a hand-tuned threshold:

$$p(\tilde{Q}_C) = \sum_{k=0}^{1} \pi_k \mathcal{N}(\tilde{Q}_C \mid \mu_k, \sigma_k^2), \tag{2}$$

where $\sum_k \pi_k = 1$. The parameters are fitted by EM, and we identify the high-complexity component by the larger fitted mean, $k_{\text{high}} = \arg\max_k \mu_k$. For each trace x_i we compute the posterior

$$\gamma_i = \frac{\pi_{k_{\text{high}}} \mathcal{N}\big(\tilde{Q}_C(x_i) \mid \mu_{k_{\text{high}}}, \sigma^2_{k_{\text{high}}}\big)}{\sum_{k=0}^{1} \pi_k \mathcal{N}\big(\tilde{Q}_C(x_i) \mid \mu_k, \sigma_k^2\big)}, \tag{3}$$

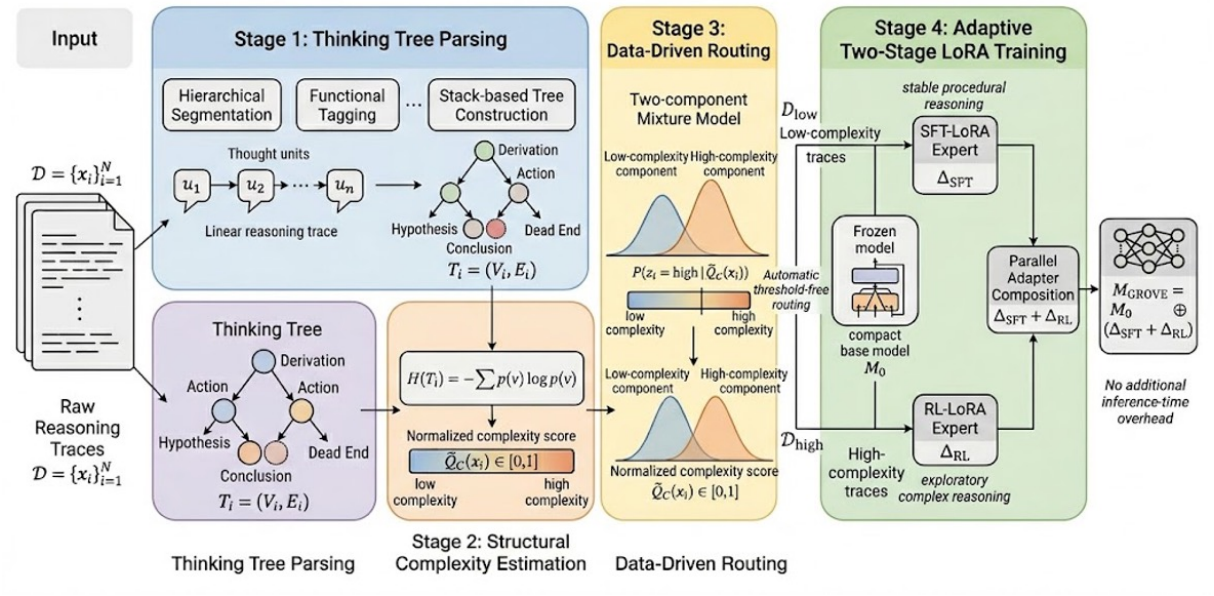


Figure 2: Overview of the GROVE framework. Each reasoning trace in the training corpus $\mathcal{D}$ is parsed into a Reasoning Tree and scored by graph entropy (C1, Section 3.2); the per-sample complexity scores $\tilde{Q}_C$ are partitioned into a low- and a high-complexity subset by a two-component Gaussian Mixture Model (C2, Section 3.3); the low-complexity subset $\mathcal{D}_{\text{low}}$ is then trained with SFT and the high-complexity subset $\mathcal{D}_{\text{high}}$ with RL, both as LoRA adapters, which are finally merged in parallel into a single deployable model M_{GROVE} (C3, Section 3.4).

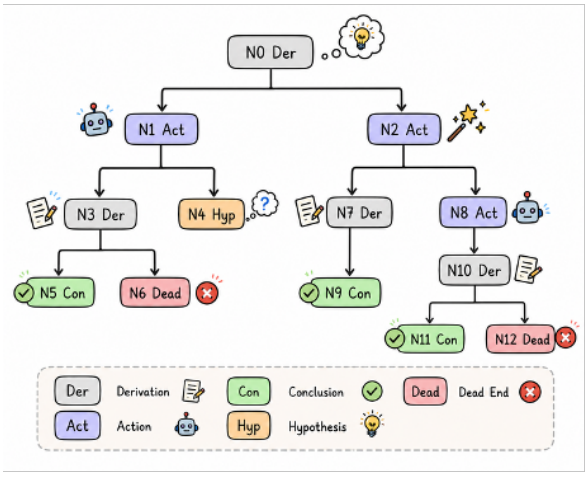


Figure 3: Reasoning Tree parsing on a worked example.

and assign the partition by the Bayes-optimal rule $\gamma_i > 0.5$:

$$\mathcal{D}_{\text{high}} = \{x_i \in \mathcal{D} : \gamma_i > 0.5\}, \quad \mathcal{D}_{\text{low}} = \mathcal{D} \setminus \mathcal{D}_{\text{high}}. \tag{4}$$

3.4 Adaptive Two-Stage LoRA Training with Parallel Merging

The two partitioned subsets are trained with complementary paradigms whose inductive biases match each data regime, and the resulting adapters are composed without inference-time cost. SFT minimizes a token-level negative log-likelihood and is therefore an imitation procedure: it is most effective when the demonstration is stable and the conditional next-token distribution is sharply peaked—conditions that hold for linear, procedural traces (low $\tilde{Q}_C$). RL, in contrast, optimizes a sequence-level reward and rewards the discovery of viable solution paths; it is most effective when the policy must select among multiple alternatives and recover from dead ends— conditions that hold for branched, exploratory traces (high $\tilde{Q}_C$). The mapping $\mathcal{D}_{\text{low}} \rightarrow$ SFT and $\mathcal{D}_{\text{high}} \rightarrow$ RL therefore aligns each paradigm's inductive bias with the regime of data it is best suited to consolidate.

LoRA training of both branches. Both branches are trained as Low-Rank Adaptation (LoRA) adapters. Instead of updating a pre-trained weight $W_0 \in \mathbb{R}^{d\times d}$ directly, LoRA injects a trainable rank-r correction BA with $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times d}$, $r \ll d$, so that the adapted forward computation becomes

$$y' = (W_0 + BA)\, x. \tag{5}$$

At initialization A is sampled from a Kaiming-uniform distribution and B is set to zero, so the

adapted model is bit-identical to the base model before any optimization step. The SFT branch is trained on $\mathcal{D}_{\text{low}}$ with a token-level cross-entropy objective, producing a trained update $\Delta_{\text{SFT}} = B_{\text{SFT}}A_{\text{SFT}}$. The RL branch is trained on $\mathcal{D}_{\text{high}}$ with Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which estimates advantages relative to a group of rollouts on the same prompt and therefore obviates a separately trained reward model; this produces $\Delta_{\text{RL}} = B_{\text{RL}}A_{\text{RL}}$. Crucially, both branches are trained *independently from the same frozen base model* M_0, not sequentially.

Parallel merging. The two trained adapters are composed into the final deployable model in a single non-destructive step:

$$M_{\text{GROVE}} = M_0 \oplus \big(\Delta_{\text{SFT}} + \Delta_{\text{RL}}\big), \quad (6)$$

where $\oplus$ denotes adding the summed low-rank update to every LoRA-targeted weight of M_0. This parallel composition introduces no inference-time overhead beyond M_0, and a singular-value analysis presented in Section 4.4.1 confirms that the SFT and RL adapters write spectrally distinct low-rank updates—evidence that parallel composition is principled rather than incidental. Pseudocode for the complete pipeline is provided in Algorithm 1 (Appendix A).

4 Experiments

We evaluate GROVE along three axes. First, we assess whether structural-complexity routing improves average pass@1 over single-stage and naive-hybrid baselines on a suite of five reasoning benchmarks across two heterogeneous training datasets (Section 4.3). Second, we examine the structure–paradigm mapping itself by reversing it, to determine whether the low-to-SFT, high-to-RL pairing reflects an intrinsic compatibility between data regime and training objective (Section 4.4.2). Third, we test the parallel adapter composition design—first through a singular-value decomposition (SVD) analysis of the trained adapters (Section 4.4.1), and then quantitatively against the conventional sequential merging schedule (Section 4.4.3).

4.1 Experimental Setup

We instantiate all methods with **DeepSeek-R1-Distill-Qwen-1.5B** (Guo et al., 2025a) as the frozen backbone M_0. To test whether GROVE is robust across different training-data distributions, we train on two reasoning corpora: **s1K** (Muennighoff et al., 2025) and a randomly sampled 1,000-example subset of **Open-RS** (Dang and Ngo, 2026). This size-matched design keeps the training budget comparable while preserving differences in corpus curation and source.

The two GROVE branches are trained with matched LoRA (Hu et al., 2021) configurations. The SFT branch is fine-tuned with a token-level cross-entropy objective on the provided solution traces, whereas the RL branch is optimized with **GRPO-LoRA** (Wang et al., 2025), which estimates advantages from grouped rollouts and thus requires no separately trained reward model. All experiments are implemented in PyTorch with HuggingFace Transformers and run on a single node with 4× NVIDIA A40 GPUs; an identical configuration is used across GROVE and every baseline. Full training and evaluation details, including all hyperparameters, are provided in Appendix B.

4.2 Benchmarks and Baselines

Benchmarks. We evaluate on five reasoning benchmarks chosen to span a wide range of difficulty and problem type, from competition mathematics to graduate-level science: **AIME24** (Art of Problem Solving, 2024) and **AMC23** (Art of Problem Solving, 2023) draw from premier U.S. mathematics competitions; **MATH500** (Hendrycks et al., 2021; Lightman et al., 2023) aggregates competition-style problems across five difficulty levels; **GPQA** (Rein et al., 2023) probes PhD-level scientific reasoning across biology, chemistry, and physics; and **Minerva** (Lewkowycz et al., 2022) tests quantitative reasoning on advanced STEM problems. Full dataset descriptions are provided in Appendix B.4. For all benchmarks we report the standard pass@1 metric.

Baselines. We compare GROVE against two families of baselines.

- **The *single-stage* family** The *single-stage* family applies one training paradigm to the entire corpus: **SFT-only** fine-tunes M_0 with SFT-LoRA on all of $\mathcal{D}$; **RL-only** fine-tunes M_0 with GRPO-LoRA on all of $\mathcal{D}$; the unmodified **Base** model receives no fine-tuning.

- **The *naive-hybrid* family** The *naive-hybrid* family splits the corpus into SFT and RL

Table 1: Comparative performance of different tuning schemes across multiple benchmarks.

Scheme	AIME24	AMC23	MATH500	GPQA	Minerva	Average
Base	23.33	70.00	84.80	31.82	30.15	48.02
s1K						
SFT-only	33.33	72.50	83.40	35.35	27.57	50.43
RL-only	30.00	75.00	83.80	31.82	29.41	50.01
Random Mixture	**36.67**	72.50	83.80	35.86	29.41	51.65
GROVE	33.33	**82.50**	**84.80**	**39.39**	**30.51**	**54.11**
Open-RS						
SFT-only	30.00	72.50	82.60	34.34	28.68	49.62
RL-only	23.33	72.50	84.60	28.28	30.15	47.77
Random Mixture	**33.33**	67.50	**85.00**	33.33	**30.51**	49.93
GROVE	**33.33**	**77.50**	84.60	**35.86**	28.68	**51.99**

subsets, but without structure-aware partitioning. **Random Mixture** assigns samples to the SFT and RL subsets uniformly at random with a fixed proportion. Because GROVE is itself an adaptive partitioning method, the appropriate counterpart is a strong fixed-proportion baseline rather than a single arbitrary split, so we sweep this proportion in 20% increments—$20\%/80\%$, $40\%/60\%$, $60\%/40\%$, and $80\%/20\%$ (RL/SFT)—and report the best-performing configuration in the main table.

4.3 Main Results

4.3.1 Comparative Performance

Table 1 compares GROVE with the baselines in Section 4.2 using pass@1 across multiple benchmarks and training corpora. Base and GROVE rows are shaded in gray and green, respectively; best and second-best scores within each dataset block are marked in bold and underline.

On s1K, GROVE reaches 54.11%, outperforming the strongest baseline, Random Mixture, by $+2.46$ points; on Open-RS, it reaches 51.99%, exceeding Random Mixture by $+2.06$ points. This comparison directly evaluates the routing mechanism: Random Mixture uses the same SFT/RL training paradigm but assigns examples without structural information, so the remaining gap reflects the benefit of complexity-aware per-sample allocation rather than a different mixture ratio.

The benchmark-level results further indicate that the improvement is stable at the aggregate level, although not uniform on every task. GROVE outperforms alternative tuning schemes on nearly all benchmarks and achieves the highest average score under both training corpora. This pattern suggests that the routing signal generalizes across different training corpora and evaluation settings, while also leaving room for task-specific variation. In contrast, the single-stage baselines are less stable: neither SFT-only nor RL-only dominates across both corpora. This sensitivity supports the need for an adaptive allocation strategy instead of assigning the entire corpus to a single training paradigm.

4.4 Analysis and Ablation Studies

4.4.1 Adapter-Space Difference between SFT and RL

We first examine whether SFT and RL remain distinguishable after both are constrained to the same LoRA parameterization. For each trained adapter, we compute the Singular Value Decomposition (SVD) of its low-rank update matrices and aggregate the normalized energy of the leading singular modes σ_1–σ_5 across transformer layers. Figure 4 reports the resulting energy distributions.

The two training paradigms induce different adapter-space structures. The RL adapter concentrates its update energy more strongly in the leading singular mode, while the SFT adapter distributes more energy across subordinate modes. This pattern suggests that SFT and RL do not become interchangeable once implemented as LoRA adapters; instead, they produce different low-rank adaptation directions under the same rank and optimization budget. This observation is consistent with the different training signals of the two paradigms: GRPO provides sequence-level reward feedback, whereas SFT fits token-level demonstrations. It also motivates the following ablations: reversed routing tests whether each data regime should be paired with its designated paradigm (Section 4.4.2), and the merging study tests whether the two specialized adapters should be composed in parallel rather than sequentially (Section 4.4.3).

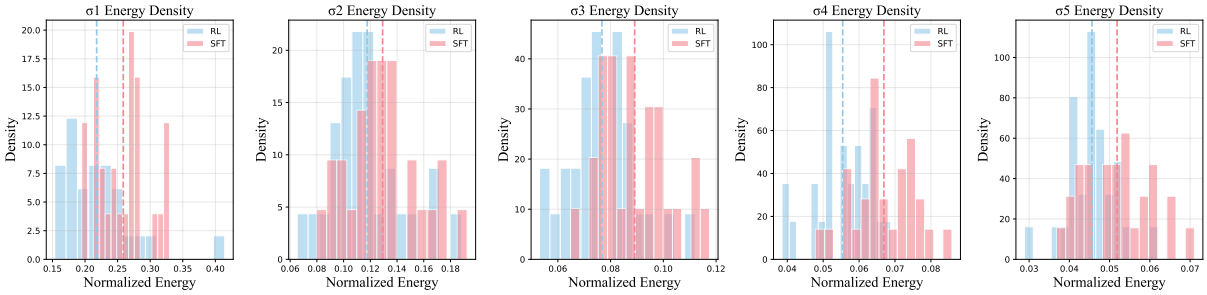


Figure 4: Normalized SVD energy of the LoRA update matrices trained under SFT and RL, aggregated over all transformer layers.

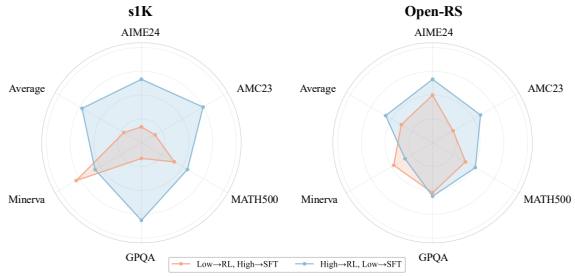


Figure 5: Reversed-routing ablation.

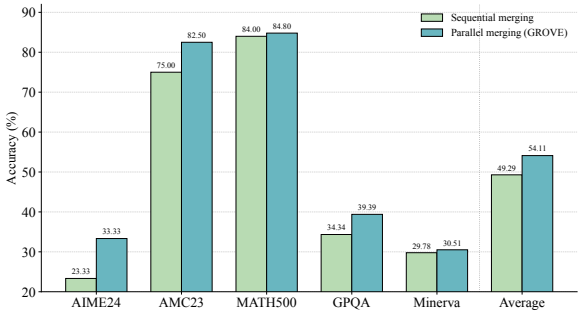


Figure 6: Results across adapter-merging techniques.

4.4.2 Reversed Routing

To test whether the routing direction is essential, we reverse the assignment while keeping the learned partition fixed: high-complexity traces are trained with SFT, and low-complexity traces are trained with RL. All other components, including the Reasoning Tree parser, GMM partition, LoRA configuration, and parallel merging strategy, remain unchanged. Figure 5 reports the results, with per-axis normalization applied for visual comparability.

Reversing the assignment consistently reduces average performance, from 54.11% to 47.00% on s1K and from 51.99% to 49.33% on Open-RS. The drops on both datasets show that disrupting the intrinsic paradigm–data pairing substantially harms fine-tuning. These results support the central design of GROVE: its gain comes not only from splitting the corpus, but from aligning low-complexity traces with SFT and high-complexity traces with RL.

4.4.3 Parallel versus Sequential Merging

We further examine whether the two LoRA adapters should be composed in parallel or sequentially. In the parallel setting used by GROVE, the SFT and RL adapters are trained independently from the same frozen base model and merged only after training. In the sequential setting, the SFT adapter is first merged into the base model, and the RL adapter is then trained on top of this SFT-adapted model. Figure 6 compares the two strategies under otherwise identical settings on s1K.

Parallel merging achieves a higher average score than sequential merging (54.11% vs. 49.29%), with the largest gain on AIME24 (33.33% vs. 23.33%). Together with the SVD analysis in Section 4.4.1, this result suggests that SFT and RL produce complementary low-rank updates that are better preserved when trained independently and composed at the final merge step. In contrast, sequential merging may introduce interference by adapting the RL branch on top of an already SFT-shifted parameter space.

5 Conclusion

We presented **GROVE**, an adaptive post-training framework that routes reasoning traces to SFT or RL based on their structural topology. Its data-intrinsic routing signal requires no human annotation, reward model, verifier, hand-tuned threshold, or inference-time overhead. Experiments across diverse corpora and benchmarks show that GROVE outperforms single-stage and naive-hybrid baselines, while reversed-routing ablations confirm that the learned structure–paradigm mapping reflects intrinsic compatibility rather than an arbitrary split.

These findings highlight structural trace signals as a promising basis for adaptive, data-centric post-training of reasoning models.

Limitations

This work has several limitations. First, GROVE depends on a deterministic Reasoning Tree parser and a graph-entropy complexity score. While efficient and parameter-free, this signal may be affected by noisy or stylistically diverse traces, and it captures structural topology rather than semantic difficulty or solution correctness. Second, the current two-component GMM imposes a binary low/high complexity split. Real reasoning data may exhibit more fine-grained or continuous structure, suggesting future extensions such as soft routing or multi-component allocation. Finally, although GROVE removes the need for a separate reward model, its RL branch still assumes automatically verifiable final-answer rewards. Extending the method to open-ended or non-verifiable reasoning tasks remains an important direction for future work.

Acknowledgments

This work is supported in part by the Science and Technology Development Fund of Macau (0107/2024/RIA2, 0061/2025/RIB2), Joint Science and Technology Research Project with Hong Kong and Macau in Key Areas of Nansha District's Science and Technology Plan (EF2024-00180-IOTSC) and the Multi-Year Research Grant of University of Macau (MYRG-GRG2023-00211-IOTSC-UMDF, MYRG-GRG2024-00180-IOTSC).

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *Preprint*, arXiv:2503.04697.

Art of Problem Solving. 2023. AMC problems and solutions. American Mathematics Competition. Accessed: 2026-08-31.

Art of Problem Solving. 2024. AIME problems and solutions. American Invitational Mathematics Examination. Accessed: 2026-08-31.

Hongyi James Cai, Junlin Wang, Xiaoyin Chen, and Bhuwan Dhingra. 2025. How much backtracking is enough? exploring the interplay of sft and rl in enhancing llm reasoning. *Preprint*, arXiv:2505.24273.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. 2025a. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *Preprint*, arXiv:2504.11468.

Liang Chen, Xueting Han, Li Shen, Jing Bai, and Kam-Fai Wong. 2025b. Beyond two-stage training: Cooperative sft and rl for llm reasoning. *Preprint*, arXiv:2509.06948.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2026. Towards reasoning era: a survey of long chain-of-thought for reasoning large language models. *Science China Information Sciences*, 69:161101.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *Preprint*, arXiv:2501.17161.

Quy-Anh Dang and Chris Ngo. 2026. Reinforcement learning for reasoning in small llms: What works and what doesn't. *Preprint*, arXiv:2503.16219.

Jacob Fein-Ashley, Dhruv Parikh, Rajgopal Kannan, and Viktor Prasanna. 2025. Mixture of thoughts: Learning to aggregate what experts think, not just what they say. *Preprint*, arXiv:2509.21164.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Efficient reasoning models: A survey. *Preprint*, arXiv:2504.10903.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *Preprint*, arXiv:2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025a. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. 2025b. Reward reasoning model. In *Advances in Neural Information Processing Systems*, volume 38, pages 150477–150510. Curran Associates, Inc.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *Preprint*, arXiv:2210.11610.

Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, Yikai Zhang, Yuqing Yang, Ting Wu, Binjie Wang, Shichao Sun, Yang Xiao, Yiyuan Li, Fan Zhou, Steffi Chern, and 9 others. 2025. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *Preprint*, arXiv:2406.12753.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Preprint*, arXiv:2206.14858.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. NuminaMath. Hugging Face repository, Numina. Accessed: 2026-08-31.

Zhong-Zhi Li, Xiao Liang, Zihao Tang, Lei Ji, Peijie Wang, Haotian Xu, Xing W, Haizhen Huang, Weiwei Deng, Yeyun Gong, Zhijiang Guo, Xiao Liu, Fei Yin, and Cheng-Lin Liu. 2025. Tl;dr: Too long, do re-weighting for efficient llm reasoning compression. *Preprint*, arXiv:2506.02678.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Bochen Lyu, Yiyang Jia, Xiaohao Cai, and Zhanxing Zhu. 2025. Transformers with rl or sft provably learn sparse boolean functions, but differently. *Preprint*, arXiv:2511.17852.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2024. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7).

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 246 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Haibo Qiu, Xiaohan Lan, Fanfan Liu, Xiaohu Sun, Delian Ruan, Peng Shi, and Lin Ma. 2025. Metis-rise: Rl incentivizes and sft enhances multimodal reasoning model learning. *Preprint*, arXiv:2506.13056.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *Preprint*, arXiv:2311.12022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, and 22 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Preprint*, arXiv:2312.06585.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 77 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems*,

volume 37, pages 9565–9584. Curran Associates, Inc.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger. 2025. Tina: Tiny reasoning models via lora. *Preprint*, arXiv:2504.15777.

Zongqian Wu, Baoduo Xu, Ruochen Cui, Mengmeng Zhan, Xiaofeng Zhu, and Lei Feng. 2025. Rethinking chain-of-thought from the perspective of self-training. *Preprint*, arXiv:2412.10827.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Jingyang Yi, Jiazheng Wang, and Sida Li. 2025. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pages 39011–39043. Curran Associates, Inc.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *Preprint*, arXiv:2505.03469.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Algorithm 1: GROVE training pipeline.

Input: Training corpus $\mathcal{D} = \{x_i\}_{i=1}^{N}$; frozen base model M_0; LoRA rank r.

Output: Merged model M_{GROVE}.

// Phase 1: Reasoning Tree parsing and scoring (§3.2)

1 **for** $x_i \in \mathcal{D}$ **do**

2 $\quad T_i \leftarrow \textsc{ParseReasoningTree}(x_i)$;

3 $\quad Q_C(T_i) \leftarrow -\sum_{v \in V_i} p(v) \log_2 p(v)$;

4 $\{\tilde{Q}_C(x_i)\}_{i=1}^{N} \leftarrow \textsc{MinMaxNormalize}\big(\{Q_C(T_i)\}_{i=1}^{N}\big)$;

// Phase 2: Data-driven partitioning (§3.3)

5 $\{\pi_k, \mu_k, \sigma_k^2\}_{k=0}^{1} \leftarrow \textsc{FitGMM}\big(\{\tilde{Q}_C(x_i)\}_{i=1}^{N}\big)$;

6 $k_{\text{high}} \leftarrow \arg\max_k \mu_k$;

7 $\mathcal{D}_{\text{high}}, \mathcal{D}_{\text{low}} \leftarrow \textsc{PartitionByPosterior}(\gamma_i > 0.5)$;

// Phase 3: Parallel LoRA training and merging (§3.4)

8 $\Delta_{\text{SFT}} \leftarrow \textsc{TrainLoRA}(M_0, \mathcal{D}_{\text{low}}, \text{obj} = \text{SFT})$;

9 $\Delta_{\text{RL}} \leftarrow \textsc{TrainLoRA}(M_0, \mathcal{D}_{\text{high}}, \text{obj} = \text{GRPO})$;

10 $M_{\text{GROVE}} \leftarrow M_0 \oplus (\Delta_{\text{SFT}} + \Delta_{\text{RL}})$;

11 **return** M_{GROVE};

A GROVE Training Pipeline

Algorithm 1 provides pseudocode for the full GROVE training pipeline introduced in Section 3. The three phases of the algorithm correspond to the three method components: parsing each trace into a Reasoning Tree and computing its graph-entropy score (lines 1–4, Section 3.2); fitting the two-component Gaussian Mixture Model to partition the corpus into $\mathcal{D}_{\text{low}}$ and $\mathcal{D}_{\text{high}}$ (lines 5–7, Section 3.3); and training the SFT and RL LoRA adapters independently from the same frozen base model M_0 before merging them in parallel (lines 8–10, Section 3.4). Concrete hyperparameter values—LoRA rank, learning rate, GRPO group size, optimizer settings, and compute configuration—are reported separately in Appendix B.1.

B Experimental details

B.1 Training Details

LoRA configuration. Both GROVE branches are trained as LoRA (Hu et al., 2021) adapters

with an identical configuration: rank $r = 32$, scaling factor $\alpha = 64$, and dropout 0.05. The adapters are inserted into seven projections in every transformer layer of M_0—the attention query, key, value, and output projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and the MLP gate, up, and down projections (`gate_proj`, `up_proj`, `down_proj`). All non-LoRA parameters of M_0 remain frozen throughout training, and the SFT and RL adapters are optimized independently from the same frozen backbone.

SFT branch. The SFT branch is fine-tuned on the low-complexity partition $\mathcal{D}_{\text{low}}$ with a token-level cross-entropy objective over the reference solution traces, producing the adapter Δ_{SFT}.

RL branch (GRPO-LoRA). The RL branch is trained on the high-complexity partition $\mathcal{D}_{\text{high}}$ with GRPO-LoRA (Shao et al., 2024; Wang et al., 2025), in which GRPO is applied to a LoRA-parameterized policy and the reward signal is the exact-match correctness of the extracted final answer against the ground truth. This removes the need for a separately trained reward model. Optimization uses a peak learning rate of 1×10^{-6} with a warmup ratio of 0.1 and cosine decay, where the minimum LR is set to 10% of the peak; training runs for 180 optimization steps. We use a per-device prompt batch size of 4 and gradient accumulation of 4 steps, yielding an effective optimization batch of 64 prompts per update across the four GPUs. For each prompt, GRPO samples a group of $G = 4$ rollouts with a maximum prompt length of 512 tokens and a maximum completion length of $3{,}584$ tokens at sampling temperature $T = 0.7$.

Implementation and hardware. All experiments use bfloat16 precision and FlashAttention-2 for attention computation. Gradient checkpointing is enabled throughout training to bound activation memory, and the GRPO rollouts are generated through vLLM. The entire pipeline runs on a single node with $4\times$ NVIDIA A40 GPUs; the same backbone, LoRA configuration, and hardware setup are used across GROVE and every baseline.

B.2 Datasets

s1K (Muennighoff et al., 2025) is a curated corpus of $1{,}000$ challenging reasoning problems, each paired with a distilled reasoning trace, aggregated from NuminaMATH (LI et al., 2024), historical AIME problems (1983–2021), and OlympicArena (Huang et al., 2025), which together span mathematics, physics, chemistry, biology, and computer science. Examples are selected under three criteria—difficulty, solution quality, and domain diversity—yielding a compact but heterogeneous reasoning corpus suitable for reasoning-centric post-training.

Open-RS (Dang and Ngo, 2026) is a $7{,}000$-problem mathematical reasoning corpus assembled for reward-driven learning, combining $3{,}000$ challenging instances from Open-S1 with $4{,}000$ problems from Deepscaler, after multi-stage filtering with Qwen2.5-7B-Instruct and Qwen2.5-Math-7B-Instruct to remove redundant and low-difficulty examples. To match the size of s1K and isolate corpus effects from scale effects, all our experiments use a uniformly randomly sampled $1{,}000$-example subset of the original $7{,}000$ problems.

B.3 Evaluation Details

Evaluation framework. All five benchmarks are evaluated under a unified generative extractive-match protocol implemented in lighteval, with model completions sampled through vLLM. For each problem, the model generates a free-text completion; the framework then extracts the candidate final answer from the completion and tests its equivalence with the ground truth, rather than performing character-level string match.

Mathematical benchmarks. For **AIME24**, **AMC23**, **MATH500**, and **Minerva**, the gold answer is parsed as a symbolic expression in the case of AIME24 and AMC23, and as LaTeX in the case of MATH500 and Minerva. In all four cases, the candidate answer is located in the model's completion by first searching for the content enclosed in `\boxed{}` and falling back to the first matched expression when no boxed form is present; equivalence is then verified by symbolic and numeric checking at 5-digit precision.

Multiple-choice benchmark. For **GPQA**, the framework extracts the model's choice as a letter index in $\{A, B, C, D\}$ and compares it against the gold index of the correct option. Because the position of the correct option is randomly permuted in the prompt at construction time, this protocol measures content understanding rather than positional bias.

Per-question scoring. For each question i in a benchmark, the model generates a single comple-

tion y_i, from which an answer is extracted and compared against the gold answer g_i. The per-question score is binary:

$$x_i = \text{verify}\big(\text{extract}(y_i),\, g_i\big) \in \{0, 1\}, \quad (7)$$

where $\text{verify}(\cdot, \cdot)$ is the equivalence check defined for each benchmark's answer type—symbolic/numeric equivalence for mathematical benchmarks and index match for the multiple-choice benchmark. The reported pass@1 score on a benchmark $\mathcal{T}$ is the mean of these per-question scores:

$$\text{pass@1} = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} x_i. \quad (8)$$

Standard protocol. All evaluations follow the default protocols implemented in lighteval for each benchmark; we make no custom modifications to the extraction logic, the equivalence checker, or the aggregation rule. Our reported pass@1 numbers are therefore directly comparable to values obtained under standard lighteval evaluation.

B.4 Evaluation Benchmarks

- **AIME24** (Art of Problem Solving, 2024) consists of 30 problems from the 2024 American Invitational Mathematics Examination, requiring precise multi-step solutions resistant to shallow pattern matching.
- **AMC23** (Art of Problem Solving, 2023) comprises 40 problems from the 2023 American Mathematics Competition, emphasizing conceptual understanding and strategic flexibility across algebra, geometry, and number theory.
- **MATH500** (Hendrycks et al., 2021; Lightman et al., 2023) is a 500-problem subset of competition-style mathematics spanning five difficulty levels, designed to evaluate step-by-step reasoning, symbolic derivation, and precise calculation.
- **GPQA** (Rein et al., 2023) contains 198 PhD-level multiple-choice science questions (the Diamond subset) in biology, chemistry, and physics, with carefully designed distractors that require nuanced reasoning rather than surface retrieval.
- **Minerva** (Lewkowycz et al., 2022) is a 272-problem quantitative reasoning benchmark drawn primarily from MIT OpenCourseWare, covering advanced STEM topics that typically require multi-step modeling or numerical computation.

C Case Study

This appendix illustrates the Reasoning Tree parser of Section 3.2 end to end on a single reasoning trace, making the three parsing steps—hierarchical segmentation, functional tagging, and stack-based tree construction—concrete.

Input trace. For this case study we use the 1,695-character reasoning trace shown below. It pursues several distinct solution strategies in sequence and includes two attempts the model itself abandons, exercising precisely the branching and backtracking behaviors the parser is designed to recover.

> Let $f(x) = 1 - a\cos x - b\sin x - A\cos 2x - B\sin 2x \geq 0$ for all $x \in \mathbb{R}$. We want to show that $a^2 + b^2 \leq 2$ and $A^2 + B^2 \leq 1$.
>
> Consider specific values of x.
> For $x = 0$, $f(0) = 1 - a - A \geq 0$, so $a + A \leq 1$.
> For $x = \pi$, $f(\pi) = 1 + a - A \geq 0$, so $A - a \leq 1$.
> For $x = \pi/2$, $f(\pi/2) = 1 - b + A \geq 0$, so $b - A \leq 1$.
>
> Consider the integral of $f(x)$ over a period.
> $\int_0^{2\pi} f(x)\,dx = 2\pi \geq 0$. This doesn't give any information.
>
> Multiply $f(x)$ by $\cos x$ and integrate.
> $\int_0^{2\pi} f(x)\cos x\,dx = -a\pi \geq 0$, so $a \leq 0$. This is not necessarily true.
>
> Consider $f(x) \geq 0$.
> Let $a = r\cos\alpha$, $b = r\sin\alpha$. Then $a\cos x + b\sin x = r\cos(x - \alpha)$.
> Let $A = R\cos\beta$, $B = R\sin\beta$. Then $A\cos 2x + B\sin 2x = R\cos(2x - \beta)$.
>
> Consider the case where $b = B = 0$.
> $f(x) = 1 - a\cos x - A\cos 2x \geq 0$.
> Let $y = \cos x$ with $-1 \leq y \leq 1$.
> $g(y) = -2Ay^2 - ay + A + 1 \geq 0$ for $y \in [-1, 1]$.
>
> Consider integrals again.
> $\int_0^{2\pi} f(x)\cos x\,dx \geq 0 \implies -a\pi \geq 0 \implies a \leq 0$.
> This is not true.
>
> Consider the vector $v = (1, \cos x, \sin x, \cos 2x, \sin 2x)$.
> The condition is $v \cdot c \geq 0$, where $c = (1, -a, -b, -A, -B)$.

Segmentation and tagging. The macro layer partitions the trace into nine strategic blocks corresponding to distinct solution attempts: substituting specific values of x, integrating $f(x)$ over a period, multiplying by $\cos x$ before integrating, a polar substitution, the special case $b{=}B{=}0$, and a vector formulation. The micro layer further decomposes these blocks into 18 tactical segments. The functional-tagging step then assigns each segment

one of the five functional labels, yielding a distribution of DERIVATION (8), ACTION (5), CONCLUSION (2), DEAD_END (2), and HYPOTHESIS (1)— a mix consistent with the trace's multi-strategy and back-tracking nature.

Tree construction. The stack-based construction step assembles the 18 tagged segments into the rooted tree shown in Figure 7. Segment 0 becomes the root; each ACTION or HYPOTHESIS segment $(1, 3, 10, 12, 14, 16)$ opens a new branch, each DERIVATION or CONCLUSION segment continues the active branch, and each DEAD_END segment (6 and 9) closes it. Notably, segment 10—an ACTION immediately following the DEAD_END at segment 9—attaches directly to the root rather than to the abandoned branch, explicitly representing the abandonment of the integral-based strategies and the start of a fresh one as sibling subtrees.

Structural complexity. The resulting tree contains 18 nodes with maximum depth 5 and maximum width 7, with six branch points and eleven leaves. The six exploratory nodes (ACTION or HYPOTHESIS) yield an exploration diversity of 0.33 and a branching balance of 1.00. Applying the graph-entropy formula in Equation (1) to the normalized node-degree distribution gives $Q_C(T) = 3.86$ bits, placing this trace toward the higher-complexity end of the score range and confirming that the parser correctly surfaces the multi-strategy structure that distinguishes branched, exploratory traces from linear, procedural ones.

D LoRA Rank Sensitivity

To assess how the LoRA rank r affects the two branches of GROVE, we sweep $r \in \{4, 8, 16, 32, 64\}$ independently for the SFT-only and RL-only baselines on the s1K training corpus, holding all other hyperparameters fixed at the values reported in Appendix B.1. The production GROVE configuration uses $r = 32$ for both branches; this sweep characterises how robust each branch is to that choice. Figures 8a and 8b report pass@1 per benchmark.

Both branches are relatively stable across the rank sweep. The five-benchmark average of the SFT-only baseline ranges from 49.77% at r=8 to 52.55% at r=16, a 2.78-point band; the RL-only baseline ranges from 50.01% at r=32 to 53.56% at r=64, a 3.55-point band. No rank produces a catastrophic drop. Within these narrow bands the two paradigms peak at different ranks—SFT at r=16 and RL at r=64— and the per-benchmark patterns are not aligned: r=64 gives the strongest GPQA score for both adapters (40.40% SFT, 38.38% RL), while smaller ranks ($r \leq 16$) remain within roughly two points of the largest rank on MATH500. The production rank r=32 is not the global maximum for either branch but lies safely within the stable band on both.

A natural reading of this stability is that the reasoning updates required from each adapter are directionally low-rank— concentrated in a small number of dominant subspaces of the base model rather than spread across the full d-dimensional weight space. Once r exceeds the effective dimensionality of these task-relevant subspaces, additional capacity contributes diminishing returns and the score plateaus; below it, the adapter still recovers most of the useful update by absorbing the most informative directions, so degradation is graceful rather than abrupt. This reading is consistent with the singular-value analysis of Section 4.4.1: both adapters concentrate the bulk of their update energy in only the leading few singular modes, implying that a moderate rank (r=32) is already sufficient to capture the paradigm-specific structure on which GROVE relies.

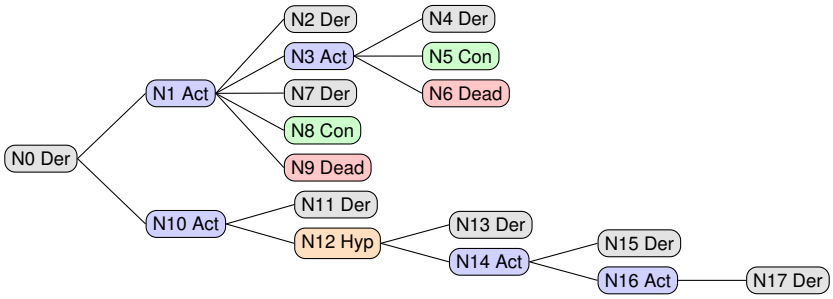
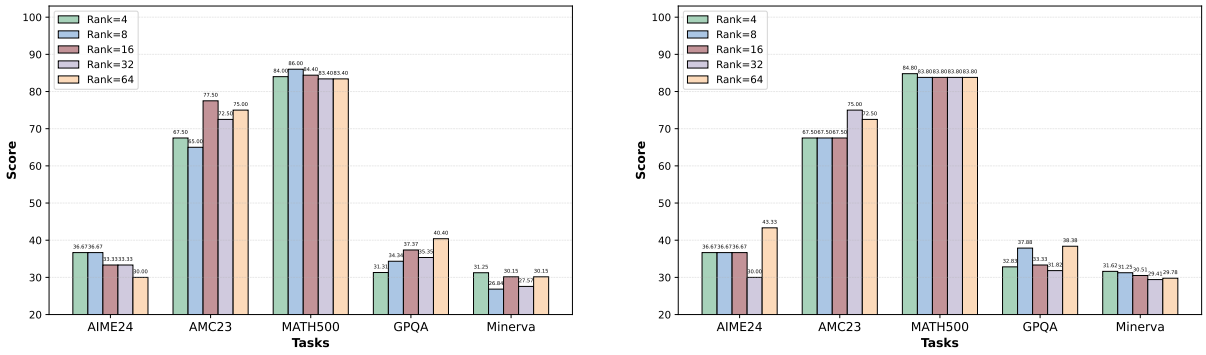


Figure 7: Reasoning Tree constructed for the worked example, with nodes colour-coded by functional tag: Der , Act , Hyp , Con , Dead . Nodes 1 ("consider specific values") and 10 ("consider $f(x) \geq 0$") are the two top-level strategies; the DEAD_END nodes 6 and 9 are leaves, marking sub-strategies after which the parser backtracks.



(a) SFT-only baseline.

(b) RL-only baseline.

Figure 8: Pass@1 across the five reasoning benchmarks as a function of LoRA rank $r \in \{4, 8, 16, 32, 64\}$ for models trained on s1K: (a) the SFT-only baseline and (b) the RL-only baseline.