SCALABLE: SCALING CAPABILITIES BEYOND THE BEST-OF-k LIMIT WITH ADAPTIVE BOUNDARY LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) often improves the single-sample reasoning performance of large language models (LLMs), yet whether the long-run capability growth can surpass the frontier reachable by the initial policy through best-of-k sampling remains debated. In this paper, we investigate the conditions under which RLVR can sustain capability growth beyond this apparent ceiling. We study problems with a controllable difficulty scale and fine-grained, verifiable solution-quality evaluation, with a focus on combinatorial optimization (CO). We observe a sharp performance decline as task difficulty increases, which we characterize as an empirical *capability boundary*. Under RL with uniform sampling, this boundary shifts toward harder instances, and its movement can be tracked using intrinsic or extrinsic signals. Based on these observations, we propose *Adaptive Boundary Reinforcement Learning* (ABRL), a novel learning strategy that dynamically concentrates training near the model's moving capability boundary. Across three CO tasks, ABRL sustains capability improvement, while experiments in other reasoning domains provide supporting evidence of its effectiveness beyond CO. At the largest problem sizes in our best-of-k evaluation across the CO tasks, ABRL achieves average normalized best-of-1 performance of 0.5548, approximately $\mathbf{8.93}\times$ the initial policy's best-of-4096 performance. In addition, we find that the number of training steps required to reach target performance exhibits an empirical power-law relationship with problem size across a broad range of problem settings. Code is available at `https://anonymous.4open.science/r/ABRL-7D3C/`.

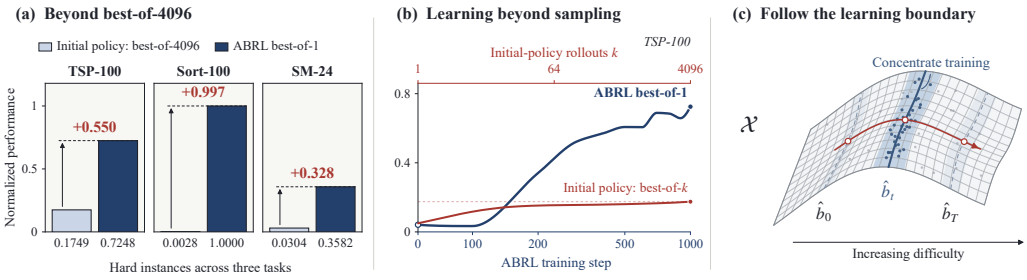

Figure 1: **Learning beyond initial-policy sampling.** (a) ABRL best-of-1 versus the initial policy's best-of-4096 performance; red labels indicate absolute performance gains. (b) Interpolated TSP-100 trajectories on independent horizontal axes: logarithmic in k; linear through step 100, then logarithmic in training steps. (c) Training follows the moving capability boundary toward harder instances.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves the single-sample reasoning performance of large language models (LLMs) (Guo et al., 2025). Yet Yue et al. (2025) find that initial policies can outperform RL-trained models in pass@k at large k. ProRL, however, reports capability expansion through prolonged RL training (Liu et al., 2025). Together, these findings suggest

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

that the effect of RL on pass@k may depend on the training mechanism. This leaves open a central question: under what conditions can RL sustain capability growth beyond the initial policy's finite-budget best-of-k frontier?

We hypothesize that sustained capability growth requires training instances that remain informative as the policy improves. Instances that support learning at one stage may become less informative as the policy becomes proficient on them (Jiang et al., 2021; Yu et al., 2025). This suggests that the training distribution should adapt to the policy's evolving capabilities.

To study this question in a controlled setting, we use combinatorial optimization (CO) tasks, which provide a controllable difficulty axis and verifiable measures of solution quality (Bello et al., 2017; Kool et al., 2019). Varying problem size lets us measure performance across difficulty levels, while fine-grained graded scores capture improvements even when solutions remain unaccepted by optimality standard. Procedural generators supply instances across sizes for controlled training and evaluation of CO problems including Traveling Salesman Problem (TSP), Sorting, and Stable Matching (SM).

To begin, we use TSP as a diagnostic setting to test whether performance exhibits a localized transition as problem size increases. We observe a sharp decline in normalized performance over a localized range of problem sizes. We characterize this transition as an empirical capability boundary. During uniform RL, this boundary shifts toward harder instances. We further find that the estimated turning points of mean thinking length and reward variance track this movement. Mean thinking length provides comparable localization without task scoring, motivating its use for adaptive boundary estimation (Section 3).

Based on these experimental observations, we propose *Adaptive Boundary Reinforcement Learning* (ABRL). ABRL is a training strategy that dynamically allocates training data around the estimated capability boundary for more sustained and steady performance growth (Figure 1). Across three CO tasks, ABRL increases the average normalized performance from 0.2149 under uniform RL to 0.3533, a relative improvement of 64.4% (Table 4). At the largest problem sizes in our best-of-k evaluation, ABRL achieves average normalized performance of 0.5548, compared with 0.0621 for the initial policy's best-of-4096 sampling (Section 4.3). We assess scaling across a broad set of settings spanning three tasks, seven performance thresholds, and three persistence criteria. Training steps to target performance exhibit empirical power-law scaling with problem size, with average log-space $R^2 > 0.95$ for each task (Section 4.4). ABRL also outperforms uniform RL in three adapted non-CO reasoning settings, providing further evidence of its applicability (Section 4.6).

Our contributions are threefold:

- **An intrinsic signal for capability boundaries.** We identify mean thinking length as an intrinsic signal that helps characterize the model's capability boundary and track its evolution during training.
- **Adaptive Boundary Reinforcement Learning (ABRL).** We introduce ABRL, an adaptive training strategy that dynamically concentrates training near the policy's evolving capability boundary.
- **Empirical frontier expansion and scaling.** Across three CO tasks, ABRL substantially improves average normalized performance over uniform RL. At harder problem sizes, its average performance exceeds the initial policy's best-of-4096 performance. We observe empirical power-law scaling between problem size and training steps to target performance. Additional experiments support ABRL's effectiveness beyond CO tasks.

2 Related Work

Capability growth under RL and repeated sampling. Repeated sampling can substantially increase the fraction of problems solved by a fixed language model (Brown et al., 2024), motivating comparisons between RL-trained policies and their initial policies at larger sampling budgets. In their evaluated settings, Yue et al. (2025) find that initial policies can match or exceed RL-trained models at large k, despite lower single-sample performance. By contrast, Liu et al. (2025) report that prolonged RL can improve performance beyond extensive sampling from the initial policy. These findings motivate studying the training conditions that support such gains. We investigate adaptive difficulty allocation, asking whether training near an evolving capability boundary can improve performance beyond the initial policy's finite-budget best-of-k reference.

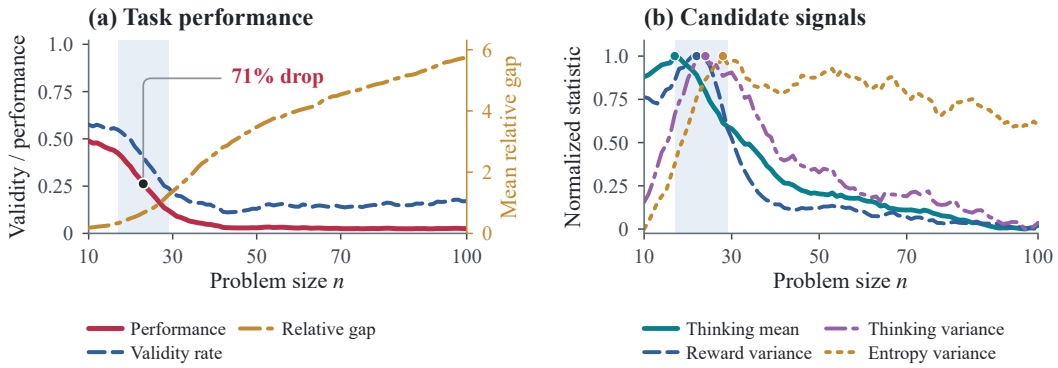

Figure 2: Initial performance transition and candidate signals. (a) Performance, validity, and relative gap. (b) Signals rescaled to $[0, 1]$. Shading marks the rapid-decline region; the black dot marks its steepest decline, and colored dots mark estimated turning points. Details in Appendix A.

Adaptive curricula and learning opportunities. Curriculum learning schedules training instances to support learning as the policy develops (Bengio et al., 2009; Klink et al., 2020; Jiang et al., 2021). In LLM reinforcement learning, DAPO filters uniformly correct or incorrect rollout groups and resamples to maintain informative training batches (Yu et al., 2025). More directly related to our work, Cai et al. (2026) use pass@k to locate capability boundaries and combine targeted teacher guidance with RL. ABRL tracks the evolving boundary through rollout statistics and concentrates training near it, using mean thinking length as a verifier-free signal for curriculum selection. Further comparisons with related curricula and learning-based CO methods are provided in Appendix B.

Intrinsic statistics of reasoning behavior. Prior work studies the relationship between reasoning length and correctness (Su et al., 2025) and the evolution of token entropy during RL (Wang et al., 2025). Other work controls response length to balance accuracy and inference cost (Aggarwal & Welleck, 2025; Luo et al., 2026). We investigate whether rollout statistics track the moving capability boundary across problem sizes and can guide adaptive training allocation.

3 Intrinsic Statistics Reveal a Moving Capability Boundary

To understand how a policy's capability changes with task difficulty, we first ask whether performance exhibits a localized transition and whether that transition moves as training proceeds. We study these questions on TSP, where problem size provides a controlled difficulty axis. We first locate the transition for the initial policy, then follow it during uniform RL and compare its movement with candidate rollout statistics.

3.1 Zero-shot performance transitions and candidate signals

We evaluate the initial Qwen3-8B policy at integer TSP sizes $n = 10, \ldots, 100$. At training step t ($t = 0$ for the initial policy), normalized performance $S_t(n)$ is the expected response score s:

$$S_t(n) = \mathbb{E}[s \mid t, n], \qquad s = \begin{cases} \dfrac{1}{1+g}, & \text{valid response,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Here g is the relative tour-length gap to a reference solution; a valid tour visits every city exactly once. Figure 2a shows that the decline in $S_0(n)$ accompanies both lower validity and larger relative gaps among valid tours. Measurement details are provided in Appendix A.1.

We characterize the empirical capability boundary by the *rapid-decline region*, the interval where performance falls sharply with problem size. Initially, smoothed performance drops by approximately 71% over $n = 17$–29. At step t, c_t denotes the problem size with the steepest decline, and a_t denotes the lower endpoint of the rapid-decline region

We then compare four rollout statistics as candidate indicators of this transition: mean thinking length, thinking-length variance, entropy variance, and reward variance. Thinking length counts

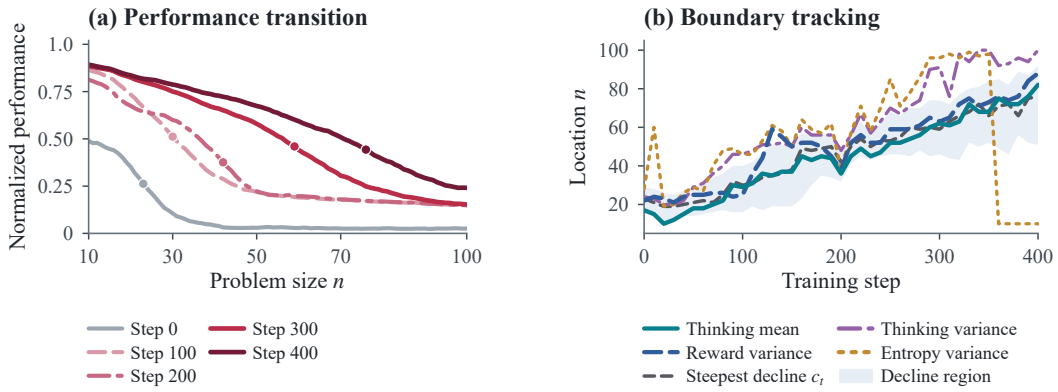

Figure 3: Boundary tracking under uniform RL. Profiles at five steps and signal turning points over training; details in Appendix A.2.

tokens in a complete reasoning span. Entropy variance measures variation in response-average token entropy across rollouts, while reward variance measures within-prompt reward variation (Appendix A.1).

Figure 2b shows that mean thinking length increases with problem size before turning downward near the onset a_0. Thinking-length variance and reward variance have estimated turning points near the steepest-decline location c_0, whereas entropy variance exhibits a broader high-value region. All four estimated turning points lie within the rapid-decline region. We next test which signals continue to track this transition during training.

3.2 Tracking the moving boundary under uniform RL

To test whether the transition moves with the policy rather than reflecting a fixed property of the task, we follow the same problem-size profile during uniform RL. The rapid-decline region remains identifiable and shifts overall toward harder instances, while the performance transition becomes more gradual (Figure 3a; evaluation details in Appendix A.1).

The estimated turning points of mean thinking length and reward variance follow this movement, whereas those of thinking-length variance and entropy variance become less aligned later in training (Figure 3b). Let p_t denote a signal's estimated turning-point location at step t. Table 1 reports median absolute distances to c_t and a_t across training evaluations. Mean thinking length and reward variance track c_t most closely, both with a median distance of 4 cities. Full profiles and robustness checks appear in Appendices A.3 and A.4.

Table 1: **Signal alignment.** Median localization errors (↓); bold marks minima.

	Localization error ↓		
Signal	$\lvert p_t - c_t \rvert$	$\lvert p_t - a_t \rvert$	**Task scoring**
Mean thinking length	**4**	**12**	No
Reward variance	**4**	14	Yes
Thinking length variance	14	26	No
Entropy variance	18	33	No

Takeaway. On TSP, mean thinking length tracks the moving capability boundary with localization comparable to reward variance, providing a practical signal for adaptive training allocation.

4 Adaptive Boundary Reinforcement Learning

We propose Adaptive Boundary Reinforcement Learning (ABRL), which dynamically concentrates training near the policy's evolving capability boundary. We evaluate this training strategy across combinatorial tasks and alternative curricula, test performance gains against the initial policy's best-of-k performance, and examine scaling with problem size and transfer to other reasoning domains.

4.1 BOUNDARY-GUIDED LEARNING

Let π_{θ_t} be the policy at training step t, and $\mathcal{D}_d$ the instance distribution at difficulty d in a finite support $\mathcal{S}$. Its performance profile is

$$P_t(d) = \mathrm{Perf}(\pi_{\theta_t}, \mathcal{D}_d)\,, \tag{2}$$

where larger values indicate better performance. A capability boundary b_t represents a location in the rapid-decline region

$$b_t \in \mathcal{B}_t, \qquad \mathcal{B}_t = \left\{ d : \frac{P_t(d-\delta) - P_t(d+\delta)}{2\delta} \geq \tau_t \right\}, \tag{3}$$

where $\delta > 0$ is the neighborhood radius and $\tau_t > 0$ the decline-rate threshold; only neighborhoods within the difficulty support are considered.

Boundary estimation. Following the boundary-tracking results in Section 3, we maintain the mean thinking-length profile $L_t(d)$ with an exponential moving average of training rollouts and use its estimated turning point as the boundary estimate $\hat{b}_t \in \mathcal{S}$ (Appendices C.2 and C.3).

Boundary-centered sampling. Easier instances near the boundary can retain quality differences, while much harder ones may yield uniformly infeasible responses. We define a scaled distance using easier- and harder-side widths $\sigma_{-,t}$ and $\sigma_{+,t}$ (Appendix C.4):

$$r_t(d) = \begin{cases} \dfrac{\hat{b}_t - d}{\sigma_{-,t}}, & d \leq \hat{b}_t, \\ \dfrac{d - \hat{b}_t}{\sigma_{+,t}}, & d > \hat{b}_t, \end{cases} \qquad \sigma_{-,t} \geq \sigma_{+,t} > 0. \tag{4}$$

For a non-increasing decay function $\kappa : [0, \infty) \to [0, 1]$, the normalized boundary kernel is

$$K_t(d) = \frac{\kappa(r_t(d))}{\sum_{u \in \mathcal{S}} \kappa(r_t(u))}, \qquad \kappa(0) = 1, \qquad \lim_{r \to \infty} \kappa(r) = 0. \tag{5}$$

Policy learning. Mixing K_t with uniform exploration of weight ϵ gives

$$\rho_t(d) = (1 - \epsilon) K_t(d) + \frac{\epsilon}{|\mathcal{S}|}, \qquad 0 \leq \epsilon < 1. \tag{6}$$

We sample difficulties $d \sim \rho_t$ and instances $x \sim \mathcal{D}_d$, then train with GRPO using graded task rewards (Appendix C.1).

4.2 EXPERIMENTAL SETUP AND MAIN RESULTS

Table 2 compares Qwen3-8B before and after task-specific ABRL with five additional models on the same three CO tasks. ABRL improves Qwen3-8B on all three tasks, increasing its unweighted average performance from 0.1107 to 0.6441. After training, the 8B policy reaches performance broadly comparable to that of several larger-scale models, although the relative differences remain task-dependent. Metric definitions and evaluation settings are provided in Appendices C.1 and C.5.

Table 2: **Performance across CO tasks** (↑). Bold marks column maxima.

Model	TSP	Sort	Stable Matching	Overall
Qwen3-8B	0.1002	0.0205	0.2116	0.1107
Qwen3-8B + ABRL (Ours)	0.8217	**0.5169**	0.5936	**0.6441**
Qwen3.5-35B-A3B	0.4031	0.1730	0.5175	0.3645
Qwen3-Max	0.5207	0.1510	0.7847	0.4854
DeepSeek-V4-Flash	0.6531	0.0309	**0.9372**	0.5404
MiniMax-M3	0.4829	0.5044	0.6079	0.5317
GPT-5.6-Luna	**0.8846**	0.2760	0.5316	0.5641

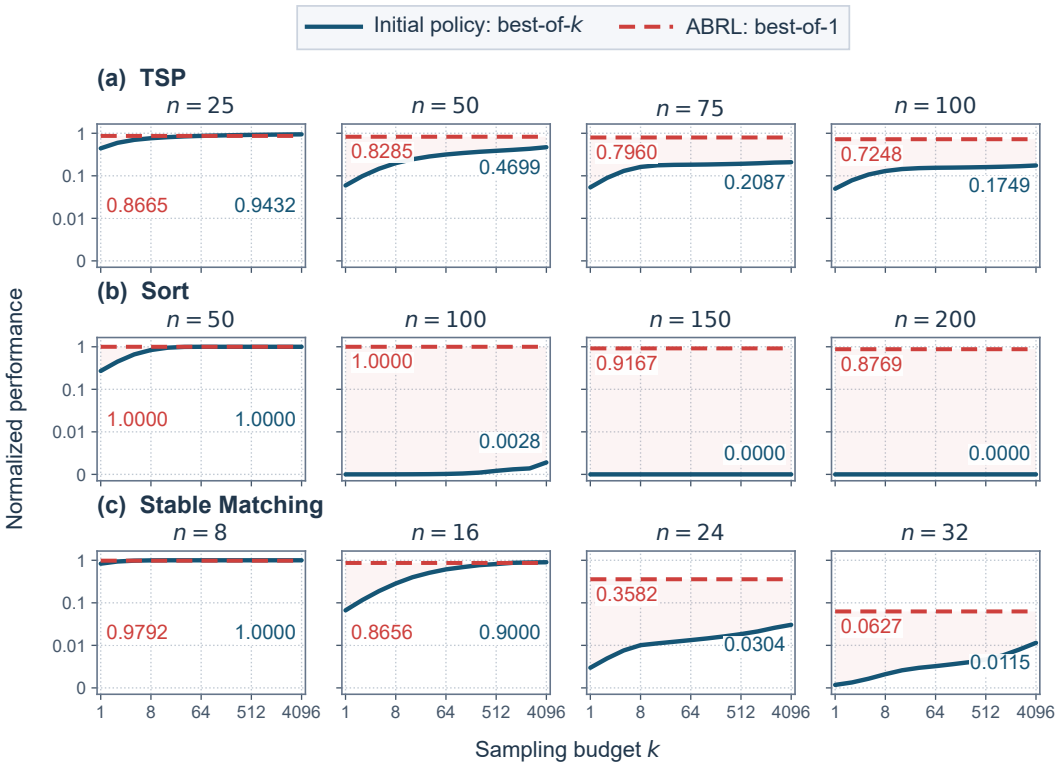

Figure 4: **ABRL best-of-1 surpasses the initial policy's best-of-4096 performance at harder problem sizes.** The horizontal axis is logarithmic; the vertical axis is logarithmic above 0.01 and linear near zero.

4.3 Beyond the initial policy's sampling performance

To assess whether ABRL's gains can be recovered through inference-time sampling, we compare its best-of-1 performance with the initial Qwen3-8B policy's best-of-k performance across four problem sizes per task, increasing k to 4096 (Figure 4; Appendix C.6).

Extensive sampling matches or exceeds ABRL at smaller problem sizes. Averaged over the largest evaluated size in each task, normalized performance is 0.5548 for ABRL versus 0.0621 for the initial policy's best-of-4096 sampling.

Takeaway. On harder instances across three CO tasks, ABRL's best-of-1 performance exceeds the initial policy's best-of-4096 performance.

4.4 Empirical scaling across combinatorial tasks

During ABRL training, we observe an approximate power-law relationship between problem size and the steps needed to reach a target performance. For task j, let $P_j(t, n)$ be normalized performance at training step t and problem size n. To reduce the influence of early-training fluctuations, we retain only threshold-attainment points observed within the analysis window, from training step 50 to the final checkpoint used for each task. Attainment times are reported as absolute training steps. Let $T_{j,\tau}(n)$ be the first observed step in the analysis

Table 3: **Fit quality across performance thresholds.** Values report the average log-space R^2 over $K = 1, 2, 3$, after limited outlier trimming per setting.

Task	$\tau = 0.30$	$\tau = 0.40$	$\tau = 0.50$
TSP	0.9602	0.9710	0.9569
Sort	0.9754	0.9745	0.9717
Stable Matching	0.9816	0.9745	0.9740

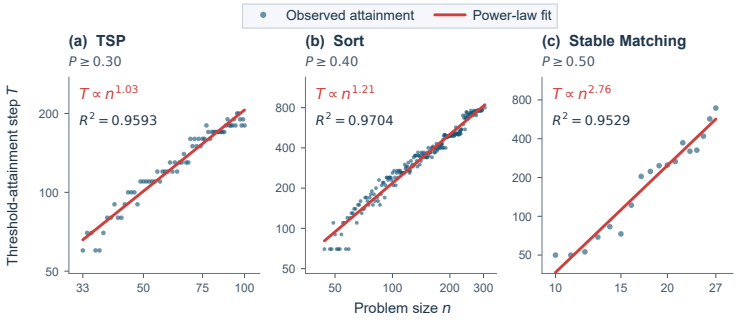

Figure 5: **Empirical scaling of training steps with problem size.** Both axes are logarithmic; R^2 is computed in log space.

window with $P_j(t, n) \geq \tau$, where τ is the performance threshold. We fit

$$T_{j,\tau}(n) \approx A_{j,\tau}\, n^{p_{j,\tau}}, \tag{7}$$

where $A_{j,\tau}$ is a fitted scale factor and $p_{j,\tau}$ is the scaling exponent. Figure 5 shows representative fits, each with log-space $R^2 > 0.95$.

To check sensitivity to the attainment criterion, we vary the performance threshold and require $K \in \{1, 2, 3\}$ consecutive observations at or above it. The average log-space R^2 over K exceeds 0.95 for every task and threshold in Table 3. Full results and analysis details are in Appendix C.7.

4.5 Training strategy comparisons and ablations

Training strategy comparisons. Table 4 compares Uniform RL, ADR (OpenAI et al., 2019), and ABRL at step 200, with policies trained separately for each task. Uniform RL samples the full difficulty range uniformly; ADR's main component samples uniformly up to an estimated boundary; ABRL concentrates near it. Both mix in 10% uniform exploration over the full range. ABRL outperforms Uniform RL on all three tasks and performs best on TSP and Stable Matching. ADR leads on Sort at step 200, but ABRL overtakes it with continued training (Figure 13; Appendix C.9).

Further discussion of the incremental curriculum and the approach of Cai et al. (2026) on TSP is provided in Appendix C.9. An additional compute-matched comparison for TSP is provided in Appendix G.

Design ablations. On TSP, we compare mean thinking length and reward variance under two sampling rules (Table 5). Direct weighting favors bins with larger signal values (Appendix C.9); boundary-centered sampling uses the signal to locate the boundary and samples nearby. Boundary-centered sampling performs better with both signals. Mean thinking length achieves the highest score, while reward variance leads under direct weighting. We examine ABRL's sensitivity to random seeds and report its validation-response truncation rates on TSP in Appendix C.9.

Table 4: **Training strategies.** Step 200.

Strategy	TSP	Sort	Stable Matching
Uniform RL	0.3473	0.0305	0.2670
ADR	0.4483	**0.1223**	0.2753
ABRL	**0.5560**	0.1043	**0.3997**

Table 5: **TSP ablations.** Step 200.

Signal	Direct	Boundary
Mean thinking length	0.4274	**0.5560**
Reward variance	0.4480	0.5307

4.6 Extension to other reasoning domains

We further evaluate ABRL in three non-CO reasoning settings: a modified GSM-Infinite task (Zhou et al., 2025), a procedurally generated integer-program execution task inspired by CoRE (Gao et al., 2026), and a modified CogniLoad task (Kaiser et al., 2026). Table 6 compares the initial policy, uniform RL, and ABRL. ABRL outperforms both the initial policy and uniform RL in all three

settings, supporting the applicability of boundary-guided training beyond combinatorial tasks. Additional evaluation details appear in Appendix C.10.

Table 6: **Performance on non-CO reasoning tasks.**

Setting	**Initial policy** Step 0	**Uniform RL** Step 100	**ABRL** Step 100
GSM-Infinite	0.4645	0.8113	**0.8525**
Integer Program Execution	0.1572	0.1766	**0.2863**
CogniLoad	0.3455	0.7957	**0.8997**

Blue highlights the best score in each setting.

5 Understanding Long-run Capability Gains

ABRL achieves better performance than test-time sampling with sustained capability growth on harder instances. We investigate the reasons for this unusual gain, including cross-difficulty performance transfer, fine-grained feedback, and training allocation.

5.1 Transfer across difficulty levels

A first possibility is that learning on easier instances transfers to harder ones. Training Qwen3-8B only on TSP sizes $n = 10$–30 raises performance at the untrained size $n = 50$ from 0.0310 to 0.2520 by step 150 (Figure 6a).

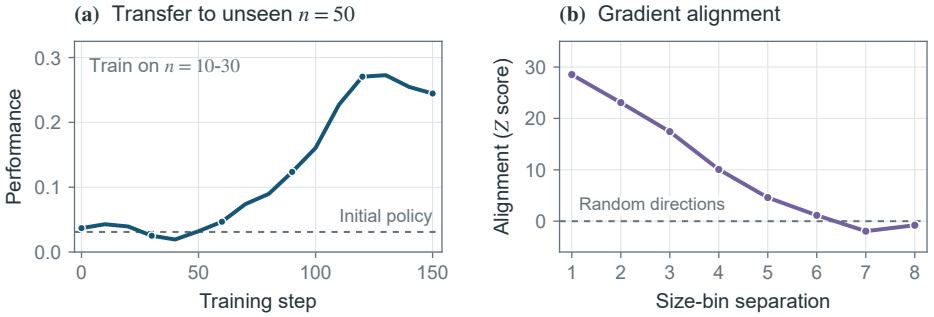

Figure 6: **Cross-size transfer** (a) and **gradient alignment** at a separate ABRL step-100 checkpoint (b). Both curves use a centered three-point moving average; text and table values are unsmoothed. Dashed lines mark initial performance and random-direction alignment, respectively.

We also compare policy gradients across size bins using Z scores relative to a random-direction baseline. Nearby bins show positive alignment, which weakens or becomes negative with greater separation (Figure 6b), consistent with shared update directions at similar difficulties.

Together, these observations suggest that training on easier instances can improve imperfect attempts at harder, untrained sizes, providing a starting point for further learning. Details appear in Appendix D.

5.2 Iterative improvement through graded feedback

For a TSP instance, a valid tour can still be graded by its length when the binary optimality criterion gives negative binary feedback. The fine-grained reward distinguishes attempts with the same binary optimality scores.

The updated policy generates new responses whose quality differences may support further learning. Iterative refinement of imperfect attempts thus offers a possible route beyond the initial policy's best-of-k sampling performance.

The two TSP runs improve similarly early on, but graded rewards are associated with higher and less variable performance later, ending at 0.9509 versus 0.6777 with binary rewards (Figure 7). Details appear in Appendix E.

5.3 Training near the capability boundary

Where should further training be concentrated? From the same ABRL step-100 model weights, whose curriculum center is $n = 46$, we train three branches for 30 additional steps at $n = 20$, $n = 46$, or $n = 100$: an easier size, the current center, and a harder size. Details appear in Appendix F.

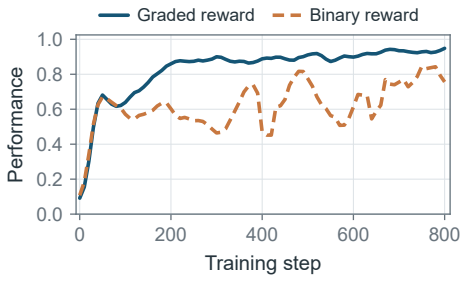

Figure 7: **Graded versus binary rewards.** Curves are smoothed; text reports raw endpoints.

Table 7: **Training location.** Performance over $n = 10$–100; Δ: change from step 100.

Training size	Performance	Δ
Start (step 100)	0.3211	—
Easier ($n = 20$)	0.3341	$+0.0130$
Boundary ($n = 46$)	**0.4863**	**+0.1652**
Harder ($n = 100$)	0.2608	-0.0603

Training at $n = 46$ gives the largest gain ($+0.1652$); across the remaining sizes, performance rises from 0.3178 to 0.4870. Easier training helps little, whereas harder training lowers overall performance. These results support following the moving boundary.

Takeaway. Cross-difficulty transfer, graded feedback, and training near the capability boundary offer a plausible account of gains beyond the initial policy's sampling performance.

6 Conclusion

Our results demonstrate that, in structured and verifiable reasoning settings, focusing learning around the current capability boundary helps RL sustain capability growth in LLMs. For certain task families, this approach yields performance on harder problems that surpasses the initial policy's finite-budget best-of-k performance. ABRL tracks the learner's evolving capability boundary effectively with rollout statistics during training, improving the reasoning capability with a higher average normalized performance compared to uniform RL. Under this strategy, best-of-1 performance on selected hard problem settings exceeds the initial policy's best-of-4096 sampling under the same evaluation protocol.

We also observe an empirical power-law relationship between training steps and the problem sizes the model can solve at a given performance level across a range of evaluated settings. Experiments in three adapted non-CO reasoning settings provide additional evidence that boundary-guided training is effective across a broader range of similar structured and verifiable reasoning tasks. Our analyses are consistent with three contributors to the observed gains: cross-difficulty transfer, iterative improvement through graded feedback, and adaptive allocation near the moving capability boundary.

Together, these findings suggest that RL can do more than reinforce behaviors already accessible to the initial policy. With adaptive training allocation over problem difficulties, RL can improve performance on harder instances beyond the initial policy's finite-budget best-of-k frontier.

Limitations and Scope Our findings are validated through empirical evaluations on structured and verifiable tasks under the training regimes considered here. Whether boundary-guided RL and the observed scaling behavior extend beyond these settings remains open, with further questions concerning extrapolation, evaluation protocols, and boundary estimation. Further discussion is provided in Appendix H.

AI USE STATEMENT

In this work, generative AI tools were used to assist with drafting and polishing the manuscript, and retrieving related literature. We also used these tools to support experimental execution through code implementation and run monitoring. The authors take responsibility for the final content of this work, including AI-assisted text, code, claims, and results.

REPRODUCIBILITY STATEMENT

The final version will specify instance generators, prompts, verifiers, reward definitions, model checkpoints, rollout budgets, curriculum update intervals, smoothing choices, and all hyperparameters in Equation 6. We will release evaluation instances and scripts sufficient to reproduce every performance frontier.

ETHICS STATEMENT

This work studies synthetic combinatorial problems and does not use personal or sensitive data. Its main risk is over-generalizing results from verifiable algorithmic tasks to open-ended domains; we therefore state this limitation and separate observations from broader hypotheses.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=4jdIxXBNve.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2017. URL https://arxiv.org/abs/1611.09940.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 41–48, 2009.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. URL https://arxiv.org/abs/2407.21787.

Pengxiang Cai, Tianchen Fang, Xiaohan Li, Qingyuan Zeng, Guocong Li, and Jintai Chen. Curriculum reinforcement learning can incentivize reasoning capacity in LLMs beyond the base model. *arXiv preprint arXiv:2606.22317*, 2026. URL https://arxiv.org/abs/2606.22317.

Tony Cai, Jianqing Fan, and Tiefeng Jiang. Distributions of angles in random packing on spheres. *Journal of Machine Learning Research*, 14(57):1837–1864, 2013. URL https://jmlr.org/papers/v14/cai13a.html.

Jun Gao, Yun Peng, Qian Qiao, Changhai Zhou, Yuhua Zhou, Shiyang Zhang, Shichao Weng, Zhenchang Xing, and Xiaoxue Ren. CoRE: A fine-grained code reasoning benchmark beyond output prediction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9446–9459. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.460. URL https://aclanthology.org/2026.findings-acl.460/.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017. URL https://forskning.ruc.dk/en/publications/an-extension-of-the-lin-kernighan-helsgaun-tsp-solver-for-constra/.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4940–4950, 2021.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. In *Advances in Neural Information Processing Systems*, volume 38, pp. 164787–164826, 2025. doi: 10.52202/085713-5495. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f0c6c93e1dbff84daf6082fef7e1a094-Paper-Conference.pdf.

Daniel Kaiser, Arnoldo Frigessi, Ali Ramezani-Kebrya, and Benjamin Ricaud. CogniLoad: A synthetic natural language reasoning benchmark with tunable length, intrinsic difficulty, and distractor density. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0Sex2H5Jnn.

Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://iclr.cc/virtual/2019/poster/1049.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17998–18031, 2025. doi: 10.52202/085713-0608. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1a22b912945fb7c0bdd079e792b31b6f-Paper-Conference.pdf.

Haotian Luo, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Xiaochun Cao, Dacheng Tao, Naiqiang Tan, and Li Shen. O1-Pruner: Length-harmonizing fine-tuning for O1-like reasoning pruning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14242–14257, 2026. doi: 10.18653/v1/2026.findings-acl.697. URL https://aclanthology.org/2026.findings-acl.697/.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in LLMs. *arXiv preprint arXiv:2505.00127*, 2025.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3850. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a797c2d2e0c1fdabf4d1ab8cd0b465c6-Abstract-Conference.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, volume 38, pp. 57654–57689, 2025. doi: 10.52202/085713-1933. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/537d5aa768c2d534016a4d06f87bc8fb-Paper-Conference.pdf.

Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9136–9144, 2022. doi: 10.1609/aaai.v36i8.20899.

Runlong Zhou, Zelin He, Yuandong Tian, Yi Wu, and Simon S. Du. Understanding curriculum learning in policy optimization for online combinatorial optimization. *arXiv preprint arXiv:2202.05423*, 2022. URL https://arxiv.org/abs/2202.05423.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. GSM-∞: How do your LLMs behave over infinitely increasing reasoning complexity and context length? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78933–78983. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25m.html.

A MEASUREMENT AND ROBUSTNESS OF THE MOVING CAPABILITY BOUNDARY

This appendix specifies the measurements behind Section 3.

A.1 EVALUATION COHORT AND STATISTIC DEFINITIONS

During uniform RL, we evaluate the policy every ten steps from step 0 to 400 on a fixed set of TSP instances covering integer sizes 10–100. Uniform denotes the fixed size proposal distribution.

For checkpoint t and size n, let $N_{t,n}$ be the total number of responses and $\mathcal{V}_{t,n}$ the indices of valid responses. For a valid tour i, define $g_i = (C_i - C_i^{\mathrm{ref}})/C_i^{\mathrm{ref}}$, where C_i is its length and C_i^{ref} is the LKH-3 reference length for the same instance. Response i receives $s_i = C_i^{\mathrm{ref}}/C_i = (1+g_i)^{-1}$ if valid and $s_i = 0$ otherwise. Normalized performance, validity rate, and mean relative gap are

$$S_t(n) = \frac{1}{N_{t,n}} \sum_i s_i, \quad V_t(n) = \frac{|\mathcal{V}_{t,n}|}{N_{t,n}}, \quad G_t(n) = \frac{1}{|\mathcal{V}_{t,n}|} \sum_{i \in \mathcal{V}_{t,n}} g_i. \tag{8}$$

The performance definition is identical to Equations 1 and 11. Thus S_t is validity times the conditional mean of $(1+g)^{-1}$, not $V_t/(1+G_t)$: scoring precedes averaging. Equal rollout counts also give equal prompt weights. Only G_t excludes invalid answers, so changes in validity can change its represented population. The LKH-3 reference (Helsgaun, 2017) is heuristic, not a certified optimum. Gap retains its original units on a separate axis in Figure 2a; intersections with validity or performance curves have no numerical meaning.

Thinking length counts tokens between a unique opening `<think>` tag and its matching `</think>` tag. Mean thinking length is averaged within each prompt and then equally across prompts.

Thinking-length variance is the sample variance across responses at a size. Entropy variance is the sample variance across response-average token entropies. The saved entropy is the sum of actor token entropies divided by response length, not cumulative entropy or within-response entropy variance. Reward variance is computed within each prompt's eight responses and then averaged equally over prompts. All sample variances use $\mathrm{ddof} = 1$.

A.2 A TRANSITION LOCATION AND ITS SURROUNDING REGION

Each node-level aggregate is smoothed with an unweighted centered window of radius $r = 5$, shortened at the observed range endpoints. Scoring and aggregation precede smoothing; there is no smoothing across training steps. A least-squares line fitted to the smoothed performance within another radius-r window gives slope $s_t(n)$. With decline rate $d_t(n) = -s_t(n)$, define

$$c_t = \arg\max_n d_t(n), \quad \mathcal{S}_t(\alpha) = \{n : d_t(n) \geq \alpha d_t(c_t)\}, \quad a_t = \min \mathcal{B}_t(\alpha), \tag{9}$$

where $\mathcal{B}_t(\alpha)$ is the connected component of $\mathcal{S}_t(\alpha)$ containing c_t. Here α is the retained fraction of the maximum decline rate, $\mathcal{S}_t(\alpha)$ contains all sizes meeting this threshold, and $\mathcal{B}_t(\alpha)$ is the rapid-decline region around c_t. The maximum decline is positive at every checkpoint. The main analysis uses $\alpha = 0.60$; at initialization, $c_0 = 23$ and $a_0 = 17$. The initial region spans $n = 17$–29, over which smoothed performance falls from 0.424 to 0.122, a relative decrease of approximately 71%. The point c_t locates the strongest decline; it is not the interval midpoint. The onset a_t describes its leading edge and depends on the threshold, whereas changing α does not change c_t.

For a candidate statistic, $M_t(n)$ denotes its value at training step t and size n, and $\widetilde{M}_t(n)$ its spatially smoothed profile. Following the local-slope formulation in Appendix C.3, let $s_{\ell,t}(n)$ and $s_{r,t}(n)$ denote the local slopes immediately below and above n, respectively. An interior turning-point candidate satisfies

$$s_{\ell,t}(n) > 0, \qquad s_{r,t}(n) < 0.$$

The statistic therefore rises on the easier side and falls on the harder side. Let p_t be the estimated location. We select the interior candidate with the highest smoothed value, breaking exact ties toward smaller sizes. If the local slope remains strictly negative throughout the observed range, we

set $p_t = 10$; if strictly positive throughout, we set $p_t = 100$. We report median absolute distances to both c_t and a_t, taking absolute values before medians. Median signed offsets $p_t - c_t$ for mean thinking length, reward variance, thinking-length variance, and entropy variance are -3, $+4$, $+14$, and $+16$ cities.

A.3 Profiles and trajectories across training

Figure 8 shows validity, conditional gap, normalized performance and its decline rate, and the four candidate statistics at five regularly spaced checkpoints. The estimated turning point of mean thinking length moves with the strongest performance decline, while thinking variance becomes broader and entropy variance's overall amplitude decreases. At steps 0, 200, and 400, the estimated turning points of mean thinking length are 17, 36, and 82, compared with steepest-decline locations 23, 42, and 76. Figure 3a retains these five performance profiles, with separate line styles for steps 100 and 200. Panel (b) compares all estimated turning points against the steepest-decline trajectory and the measured 60% region. The native-unit profiles below additionally expose changes in each candidate's amplitude and shape.

Figure 9 includes every checkpoint. Rows are rescaled independently to $[0, 1]$ only for display; locations are computed from the spatially smoothed profiles before display normalization. The estimated location of entropy variance lies near the initial transition but later reaches the lower endpoint, $n = 10$. Mean thinking length and reward variance remain closer to its strongest decline.

A.4 Parameter sensitivity

Table 8 varies the spatial radius r and retained-decline threshold α, recomputing performance references and estimated turning points at each setting. The radius applies to both profile smoothing and local slope fitting; α affects only the region and its onset a_t. Mean thinking length has lower error than the two reward-free variance alternatives for both targets at every tested setting. Its error to the strongest decline is three or four cities across radii, whereas onset error varies substantially with radius and threshold.

Table 8: Sensitivity of transition localization. Each entry gives median absolute distances *to the steepest point / to the onset*, in cities across 41 checkpoints. Radius applies to spatial smoothing and slope fitting; α defines the connected rapid-decline region. Only the onset depends on α.

Radius	α	Length mean	Reward var.	Length var.	Entropy var.
3	0.50	4/10	3/14	15/27	19/32
3	0.60	4/7	3/11	15/23	19/29
3	0.75	4/2	3/7	15/20	19/26
3	0.90	4/3	3/5	15/17	19/23
5	0.50	4/16	4/20	14/32	18/33
5	0.60	4/12	4/14	14/26	18/33
5	0.75	4/5	4/10	14/22	18/29
5	0.90	4/3	4/7	14/18	18/22
10	0.50	3/19	4/27	16/37	19/37
10	0.60	3/15	4/22	16/32	19/34
10	0.75	3/9	4/16	16/27	19/31
10	0.90	3/6	4/11	16/22	19/26

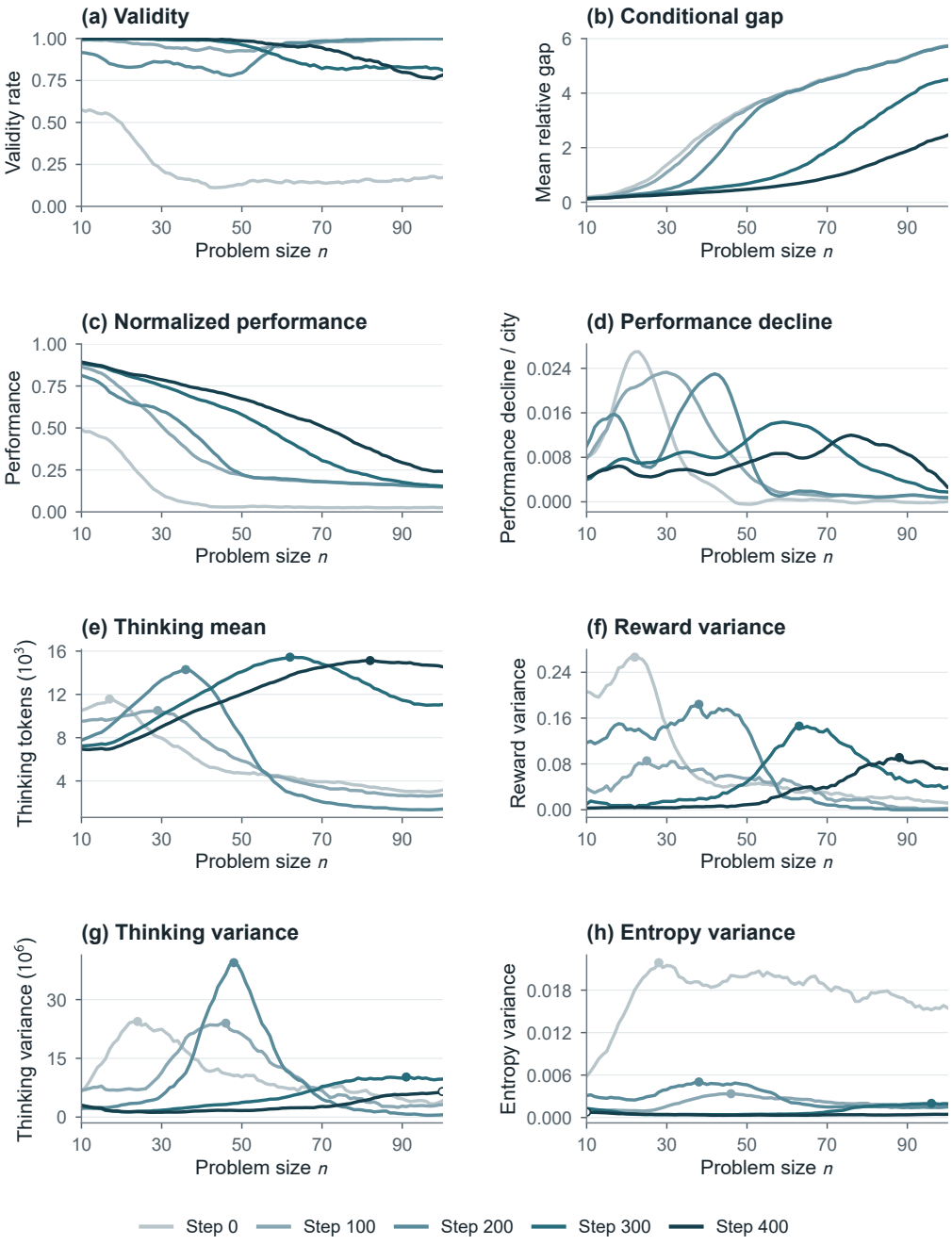

Figure 8: Native-unit profiles at regularly spaced checkpoints. All panels use the same spatial smoothing and light-to-dark checkpoint colors. Panels (a–d) separate validity, conditional gap, normalized performance, and its decline rate. Markers in (e–h) show estimated locations; hollow markers indicate endpoint estimates under the convention in Appendix A.2. Performance decline is measured along problem size, not training time.

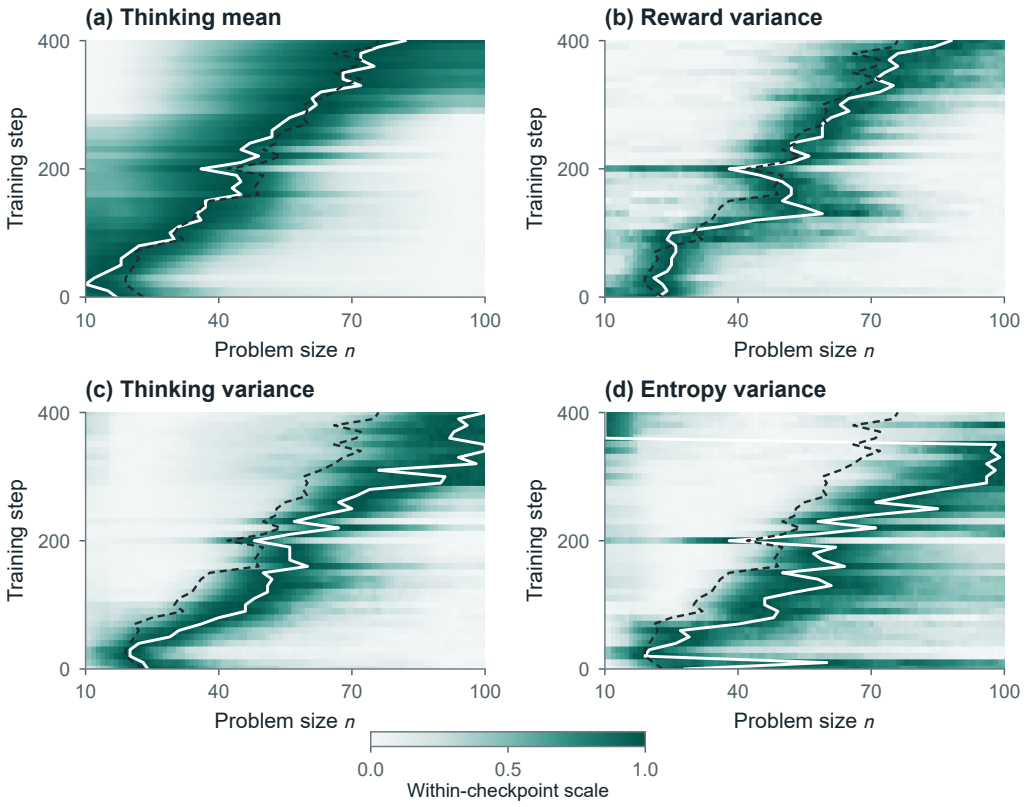

Figure 9: All 41 checkpoints, showing shape and location together. Color gives the within-checkpoint normalized statistic. The white curve traces the estimated location p_t; the dark dashed curve marks the steepest performance decline. Each row is an observed checkpoint; no temporal smoothing is applied. Row-wise normalization compares shape and location, not amplitude.

B Comparison with Related Approaches

Table 9 distinguishes the signal used to allocate training from the feedback used to optimize the policy. The term *verifier-free* describes ABRL's boundary estimation and curriculum selection, not its RL objective.

Table 9: Representative mechanisms for selecting or shaping learning opportunities. Descriptions refer to the cited implementations.

Approach	Signal or feedback	Mechanism and relation to ABRL
Self-paced RL (Klink et al., 2020)	Agent performance and target task distribution	Learns a task distribution progressing toward a target; ABRL follows a local length transition along a size axis.
Prioritized level replay (Jiang et al., 2021)	Estimated learning potential, including TD errors	Prioritizes revisiting levels; ABRL generates instances at sizes selected using the maintained thinking-length profile.
DAPO (Yu et al., 2025)	Outcomes within each prompt's rollout group	Filters groups with uniformly correct or incorrect answers; ABRL changes the distribution of problem sizes before observing their rewards.
Boundary-aware Curriculum RL (Cai et al., 2026)	Pass@k boundary evaluation and teacher guidance	Because Pass@k collapses evaluation to verifier-based success, it does not exploit continuous scores for reward shaping; ABRL instead estimates and tracks the boundary from intrinsic response lengths without teacher guidance, making it better suited to reward-shaping scenarios with continuous quality feedback.
Hardness-adaptive TSP (Zhang et al., 2022)	Solver-dependent hardness evaluation	Generates challenging instances and trains a neural TSP solver; ABRL tracks LLM reasoning length to reweight a fixed size support.
End-to-end LLM CO (Jiang et al., 2025)	Solver demonstrations, feasibility and quality rewards	Combines SFT and RL for solution generation; ABRL studies sustained capability growth through adaptive curriculum selection.
ABRL (ours)	Complete-think length for selection; graded rewards for GRPO	Uses EMA and local slope transitions to estimate a boundary and sample sizes through an asymmetric exponential kernel.

What existing curriculum theory supports. Zhou et al. (2022) analyze policy optimization for online CO through latent Markov decision processes. Their theory connects curriculum benefits to reduced distribution shift, and establishes a specific result for the Best Choice Problem. This provides a theoretical precedent for examining how task allocation affects optimization. Its assumptions and guarantees do not directly establish convergence or continued performance growth for GRPO-trained language models. We therefore investigate possible mechanisms behind ABRL's gains and test its boundary and scaling behavior empirically.

Signal use and scope of novelty. The length and entropy studies discussed in Section 2 motivate intrinsic diagnostics, but do not substitute for validating ABRL's estimator. We test whether local length transitions align with independently evaluated performance and whether acting on them improves training. Likewise, reward-variance and response-length bucket baselines test alternative allocation mechanisms; they should not be conflated with reproductions of the full published approaches in Table 9. Our comparisons concern LLM training strategies under common task settings, not a ranking of specialized CO solvers or a general comparison of SFT and RL.

C ABRL Implementation and Task Configurations

Across the three combinatorial tasks, ABRL maintains thinking-length statistics from training rollouts using EMA. Boundary estimation uses local smoothing and local slopes, followed by asymmet-

ric sampling with uniform exploration. Table 10 summarizes the configurations for TSP, Sort, and Stable Matching.

Table 10: ABRL configurations. Radii are in problem-size units; the EMA history window is in training steps.

Parameter	TSP	Sort	Stable matching
Size support	10–100	30–500	5–32
Smoothing radius	5	5	3
Slope-fit radius	5	10	3
Slope ratio q	-0.5	0.1	-0.5
Left/right coefficients a_-/a_+	$3/1.5$	$6/1.5$	$2/1$
Exploration fraction ϵ	0.1	0.1	0.1
EMA history window W	10	10	10
EMA base weight a_0	0.2	0.2	0.2
EMA weight amplitude a_1	0.7	0.7	0.7
Exposure decay scale λ	2	2	2
Reference rollout count m_0	8	8	8

C.1 Task rewards and evaluation metrics

We use GRPO (Shao et al., 2024) through the veRL framework (Sheng et al., 2025) without a learned value critic.

Training rewards. Infeasible responses receive zero task reward. For feasible responses y to instance x, we use

$$f_x^{\mathrm{TSP}}(y) = \frac{1}{1 + g_{\mathrm{ref}}(x, y)}, \qquad f_x^{\mathrm{Sort}}(y) = \frac{1}{1 + I(y)}, \qquad f_x^{\mathrm{SM}}(y) = \frac{1}{1 + (B(y)/2)^2}. \tag{10}$$

For TSP, $g_{\mathrm{ref}}(x, y) = [C(y) - C_{\mathrm{LKH}}(x)]/C_{\mathrm{LKH}}(x)$, where $C(y)$ is the tour length and $C_{\mathrm{LKH}}(x)$ is the LKH-3 reference length (Helsgaun, 2017). The reference is a heuristic solution rather than a certified optimum. Sort requires a complete permutation of the input, and $I(y)$ counts inversions. Stable Matching requires a complete one-to-one matching, and $B(y)$ counts blocking pairs. Any matching with $B = 0$ receives maximal reward; agreement with a canonical matching is not required.

Evaluation metrics. Within each task, let N be the total number of evaluated responses and $\mathcal{V}$ the indices of valid responses. We report

$$\mathrm{Performance}_{\mathrm{TSP}} = \frac{1}{N} \sum_{i \in \mathcal{V}} \frac{1}{1 + g_i}, \tag{11}$$

$$\mathrm{Performance}_{\mathrm{Sort}} = \frac{1}{N} \sum_{i \in \mathcal{V}} \frac{1}{1 + I_i}, \tag{12}$$

$$\mathrm{Performance}_{\mathrm{SM}} = \frac{1}{N} \sum_{i \in \mathcal{V}} \frac{1}{1 + B_i}, \tag{13}$$

where $g_i = g_{\mathrm{ref}}(x_i, y_i) = (C_i - C_i^{\mathrm{ref}})/C_i^{\mathrm{ref}}$, with $C_i = C(y_i)$ and $C_i^{\mathrm{ref}} = C_{\mathrm{LKH}}(x_i)$; $I_i = I(y_i)$ counts inversions, and $B_i = B(y_i)$ counts blocking pairs. Invalid responses contribute zero to all three metrics. At a fixed training step t and size n, TSP performance is $S_t(n)$ in Equation 1; Appendix A.1 uses the same definition.

C.2 Maintaining length statistics with EMA

We follow the notation of Section 4.1: difficulty d corresponds to problem size n in our CO experiments. At training step t, $L_t(d)$ denotes the maintained mean thinking length and $\widehat{L}_t(d)$ the current observed mean. The exponential moving average (EMA) update is

$$L_t(d) \leftarrow \big(1 - \alpha_t(d)\big) L_t(d) + \alpha_t(d) \widehat{L}_t(d), \tag{14}$$

where $\alpha_t(d)$ is the update weight. Let $C_t(d)$ count training prompt groups sampled at difficulty d during the preceding W training steps, and let $m_t(d)$ denote the effective rollout count contributing to the current observation. The dynamic coefficient is

$$a(C_t(d)) = a_0 + a_1 \exp[-C_t(d)/\lambda], \tag{15}$$

$$\tau(C_t(d)) = -\frac{m_0}{\ln[1 - a(C_t(d))]}, \tag{16}$$

$$\alpha_t(d) = 1 - \exp[-m_t(d)/\tau(C_t(d))]. \tag{17}$$

Here $a(C_t(d))$ is the update weight for the reference rollout count m_0, and $\tau(C_t(d))$ is the corresponding decay scale. The base weight a_0, amplitude a_1, exposure decay scale λ, history window W, and reference count m_0 are given in Table 10.

C.3 Local-slope boundary estimation

After local smoothing, we identify candidate transition locations from the slopes on either side. At a candidate difficulty d, let $s_t^-(d)$ and $s_t^+(d)$ be the fitted slopes immediately below and above it. We require

$$s_t^-(d) > 0, \qquad s_t^+(d) \le q\, s_t^-(d), \tag{18}$$

where q controls the required change in slope. These conditions identify locations where thinking length rises on the easier side and its growth weakens or reverses on the harder side. Candidate locations guide updates to the estimated boundary $\hat{b}_t$, which sets the sampling center. The smoothing radius, slope-fit radius, and slope ratio are listed in Table 10.

C.4 Kernel and width schedules

We instantiate the kernel in Equation 5 with exponential decay and boundary-dependent widths:

$$\kappa(r) = e^{-r}, \qquad \sigma_{-,t} = a_- \ln \hat{b}_t, \qquad \sigma_{+,t} = a_+ \ln \hat{b}_t. \tag{19}$$

Table 10 gives the coefficients a_- and a_+, the uniform exploration fraction ϵ, and the difficulty support for each task. The coefficients remain fixed within a run, while the widths vary with the estimated boundary $\hat{b}_t$.

C.5 Model evaluation settings

The evaluation size ranges are 10–100 for TSP, 30–500 for Sort, and 5–32 for Stable Matching. Table 11 lists the generation settings used for the model comparisons in Table 2. Both Qwen3-8B variants use the default thinking mode with a 20K-token generation limit (20,480 tokens).

Table 11: Generation settings for model evaluation. Token limits are per response.

Model	Max. generated tokens	Thinking configuration
Qwen3-8B	20,480	Default thinking mode
Qwen3-8B + ABRL	20,480	Default thinking mode
Qwen3.5-35B-A3B	65,536	Default settings
Qwen3-Max	32,768	Default settings
DeepSeek-V4-Flash	65,536	Default settings
MiniMax-M3	65,536	Default settings
GPT-5.6-Luna	32,768	High reasoning effort

C.6 Best-of-k evaluation and supporting results

Validity, exact success, and solution errors. For TSP, Sort, and Stable Matching, validity requires a complete feasible tour, an exact permutation of the input, or a complete one-to-one matching, respectively. Strict success requires zero inversions ($I = 0$) for Sort and zero blocking pairs ($B = 0$) for Stable Matching. Validity alone is insufficient in either task. For best-of-k evaluation, Valid@k

and Pass@k report the rates of obtaining at least one valid or strictly successful response within a sampling budget of k. TSP reference gaps are measured against LKH-3 heuristic tours (Helsgaun, 2017) and do not define certified optimality or exact success. Mean reference gaps and blocking-pair counts are computed only where a valid answer is obtained. Invalid answers contribute zero to performance. ABRL reference lines are shown for all metrics. Raw-error panels use size-specific linear vertical ranges; fraction and performance panels use 0–1. Figures 10, 11, and 12 show all 36 supplied metric curves across tasks and problem sizes, with linear coordinates throughout.

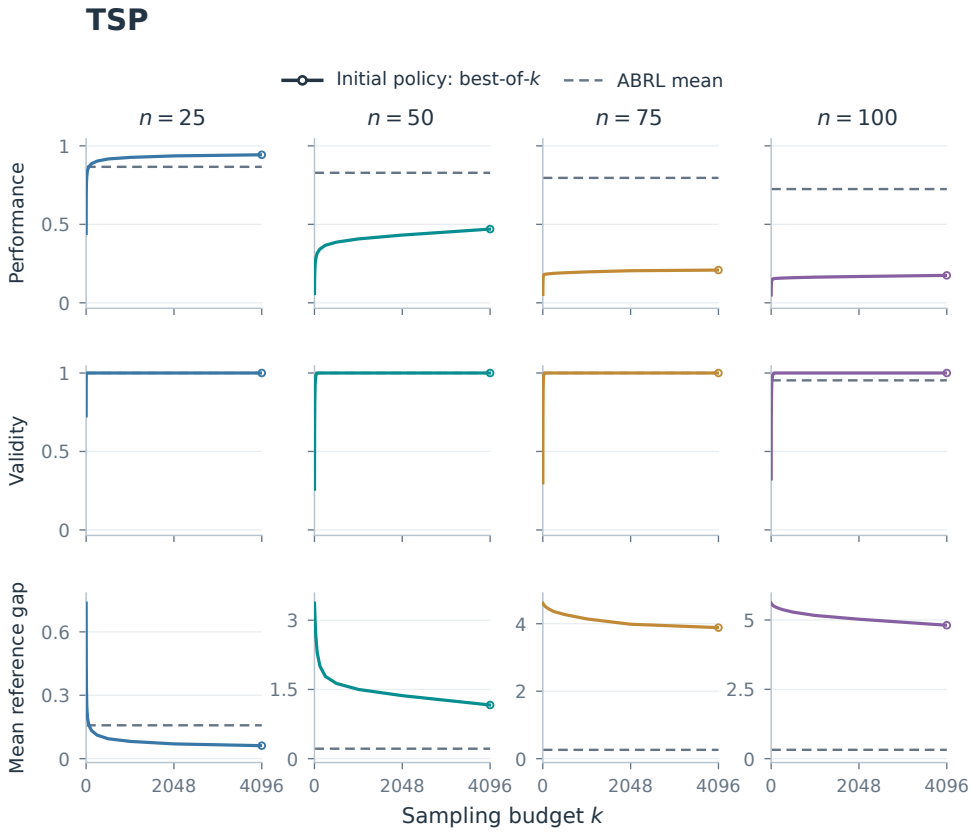

Figure 10: **TSP: all sizes and metrics on linear axes.** Rows show performance, validity, and mean reference gap among valid answers; columns show the four problem sizes. Colored curves connect $k = 1, 2, 4, \ldots, 4096$, and circles mark their endpoints. Gray dashed lines give ABRL means for all three metrics. All four initial-policy pools are complete. Gap panels use separate linear vertical ranges.

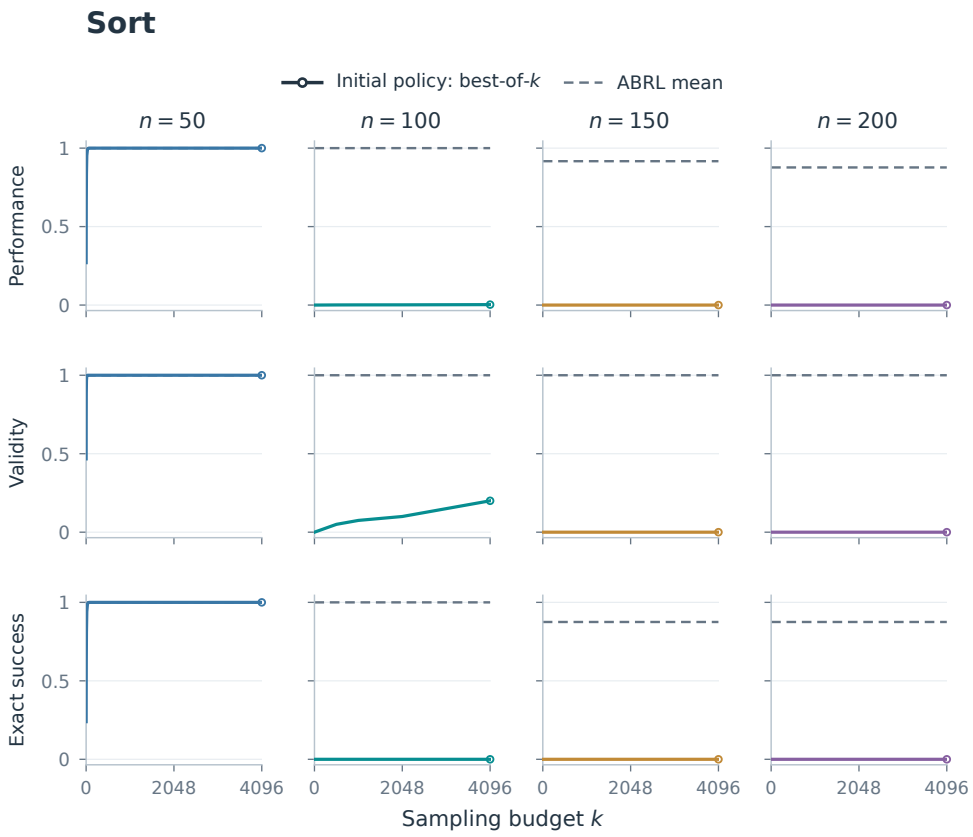

Figure 11: **Sort: all sizes and metrics on linear axes.** Rows show performance, validity, and exact success ($I = 0$). Gray dashed lines show the corresponding ABRL means. Curves connect $k = 1, 2, 4, \ldots, 4096$. At $n = 150, 200$, every initial-policy metric is zero.

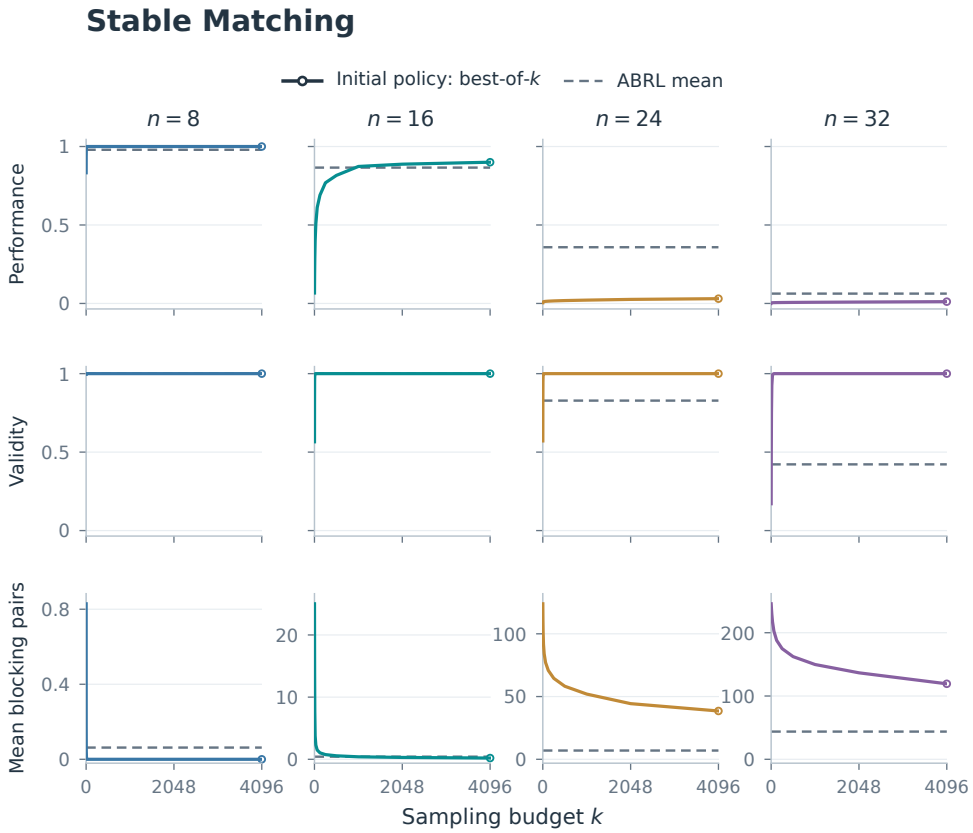

Figure 12: **Stable Matching: all sizes and metrics on linear axes.** Rows show performance, validity, and mean blocking-pair count among valid answers. Gray dashed lines give ABRL means for all three metrics. Curves connect $k = 1, 2, 4, \ldots, 4096$, and raw-error panels use separate linear vertical ranges.

C.7 THRESHOLD AND PERSISTENCE SWEEP

K denotes the number of consecutive scheduled observations at which normalized performance reaches or exceeds a threshold τ for the same problem size. We evaluate $K \in \{1, 2, 3\}$ and seven thresholds $\tau \in \{0.30, 0.35, \ldots, 0.60\}$. For $K > 1$, attainment is dated to the first observation of the first qualifying sequence for all three tasks.

TSP and Sort use validation probes; Stable Matching uses training rollouts. Figure 5 shows $K = 1$ results, with power-law fits to all displayed points. Table 12 reports log-space R^2.

Table 12: **Complete threshold and persistence sweep.** Cells report in-sample log-space R^2 from fits with limited outlier trimming per setting. The Mean column averages $K = 1, 2, 3$; Mean rows average the seven thresholds equally.

τ	$K = 1$	$K = 2$	$K = 3$	Mean
TSP				
0.30	0.9593	0.9608	0.9605	0.9602
0.35	0.9571	0.9716	0.9735	0.9674
0.40	0.9566	0.9726	0.9838	0.9710
0.45	0.9376	0.9762	0.9762	0.9633
0.50	0.9035	0.9830	0.9843	0.9569
0.55	0.8535	0.9881	0.9845	0.9420
0.60	0.8626	0.9887	0.9890	0.9468
Mean	0.9186	0.9773	0.9788	0.9582
Sort				
0.30	0.9729	0.9754	0.9777	0.9754
0.35	0.9721	0.9763	0.9779	0.9755
0.40	0.9704	0.9768	0.9762	0.9745
0.45	0.9716	0.9734	0.9726	0.9725
0.50	0.9699	0.9729	0.9722	0.9717
0.55	0.9687	0.9704	0.9668	0.9686
0.60	0.9689	0.9684	0.9663	0.9678
Mean	0.9706	0.9734	0.9728	0.9723
Stable Matching				
0.30	0.9607	0.9933	0.9908	0.9816
0.35	0.9680	0.9884	0.9910	0.9825
0.40	0.9638	0.9858	0.9737	0.9745
0.45	0.9515	0.9704	0.9812	0.9677
0.50	0.9529	0.9868	0.9824	0.9740
0.55	0.9617	0.9912	0.9908	0.9812
0.60	0.9564	0.9893	0.9939	0.9799
Mean	0.9593	0.9865	0.9862	0.9773

Across the three tasks, these results support an empirical power-law relation between problem size and attainment steps under different performance thresholds and persistence requirements.

C.8 ALTERNATIVE FUNCTIONAL FORMS

We also compare the power-law model with exponential and logarithmic-growth alternatives,

$$T = An^p, \qquad T = Ae^{bn}, \qquad T = a + b \log n. \tag{20}$$

Table 13 reports task-wise average in-sample R^2 over the same seven thresholds and $K \in \{1, 2, 3\}$ settings as Table 12. The power-law fit is strongest in all three tasks. For some Sort threshold/K settings, the logarithmic-growth fit predicts $\widehat{T} \leq 0$ over part of the observed range. Because the subsequent calculation uses $\log(\widehat{T})$, those settings are marked as having no valid result and no R^2 is forced. This reflects instability of the fit, not missing Sort data or a program error. In particular, the form $T = a + b \log n$ is poorly suited to Sort and can produce negative predicted attainment steps within the node range or at its boundary.

Table 13: **Alternative scaling fits.** Task-wise average in-sample R^2 over the same threshold and persistence settings as Table 12.

Model	TSP	Sort	Stable Matching
Power law $T = An^p$	0.9582	0.9723	0.9773
Exponential $T = Ae^{bn}$	0.9032	0.7750	0.9209
Logarithmic growth $T = a + b \log n$	0.8803	unstable	0.3259

C.9 Training-strategy comparison protocol

Tables 4 and 5 report Qwen3-8B runs at step 200, using the task performance definitions in Appendix C.1. The evaluation ranges are $n = 10$–100 for TSP, $n = 30$–500 for Sort, and $n = 5$–32 for Stable Matching.

Training strategies. Uniform RL uses a fixed uniform difficulty proposal. Our task-specific ADR implementations update an upper difficulty from validation statistics and sample uniformly below it. TSP and Stable Matching select the largest evaluated size where at least half of prompt groups contain two valid responses with non-identical rewards. Sort selects sizes with empirical pass@4 of at least 0.5. ABRL uses the boundary estimator and sampling rule described in Appendices C.3 and C.4.

Direct weighting. The TSP baselines group sizes into bins 10–15, 16–20, and successive five-city bins. Each size receives a weight from its bin's statistic, with normalization accounting for the number of sizes per bin. Both direct-weighting baselines use 10% uniform mixing. These weights determine sampling probabilities directly, without locating a boundary.

Incremental curriculum on TSP. The incremental curriculum samples problem sizes uniformly from $n = 10$–30 for steps 1–100, $n = 10$–50 for steps 101–200, and $n = 10$–100 from step 201 onward, then randomly selects an instance at the sampled size. Table 14 compares its performance with Uniform RL and ABRL. Uniform RL performs best at step 100. The incremental curriculum outperforms Uniform RL at steps 200 and 300, showing the benefit of progressively expanding the training difficulty range; ABRL achieves higher performance at both checkpoints. ABRL estimates the moving capability boundary from rollout statistics and concentrates sampling around it. These results suggest that tracking the evolving boundary and adapting the sampling distribution can improve on a fixed incremental schedule.

Table 14: **TSP performance under an incremental curriculum.** Average normalized performance over $n = 10$–100.

Step	Uniform RL	Incremental curriculum	ABRL
100	0.332681	0.293289	0.327307
200	0.347286	0.426547	0.555982
300	0.499688	0.604180	0.673672

Pass@8-guided curriculum on TSP. We adapt the pass@k-based boundary signal of Cai et al. (2026) to TSP within our training framework. A problem is counted as solved if at least one of eight responses is valid and satisfies an approximate success criterion relative to the reference solution. We average this binary indicator equally across problems at each size; invalid responses do not count as successes. Validation estimates update the dynamic EMA described in Appendix C.2, followed by smoothing over $n \pm 5$ and local slope fitting over the same radius. The curriculum retains ABRL's asymmetric sampling kernel and 10% uniform exploration; the training rewards remain graded.

Across validation updates through step 150, the center fluctuates between $n = 11$ and $n = 18$, without sustained movement toward harder sizes (Table 15). This behavior illustrates a limitation of using binary success signals to guide training with graded rewards: solution quality can improve without changing the corresponding binary success indicators. Consequently, this Pass@8-based

curriculum signal can remain concentrated on easy instances despite substantial gains in average performance.

Table 15: **Pass@8-guided curriculum centers on TSP.** Estimated problem-size centers after each validation update.

Step	0	10	20	30	40	50	60	70
Center	14	13	12	12	14	14	11	11
Step	80	90	100	110	120	130	140	150
Center	12	12	12	12	12	18	12	17

Early-training stability across random seeds on TSP. We run ABRL on TSP with seeds 12345 and 67890, varying both training and rollout randomness. Table 16 reports normalized performance and the actual sampling centers during the first 50 training steps. Both runs improve in performance at every evaluated checkpoint. Their sampling centers move toward larger problem sizes without reversals at the recorded curriculum updates, both reaching $n = 22$ at step 50. These results provide preliminary evidence of consistent early learning and curriculum progression across the two seeds.

Table 16: **ABRL on TSP under two random seeds.** Average normalized performance over $n = 10$–100 and the centers used by the sampling distribution.

	Performance		Sampling center	
Step	Seed 12345	Seed 67890	Seed 12345	Seed 67890
0	0.11117	0.10798	14	13
10	0.15641	0.14848	14	13
20	0.19585	0.17739	16	14
30	0.21116	0.19357	18	17
40	0.22218	0.20097	21	19
50	0.22947	0.20810	22	22

Validation-response truncation on TSP. We measure truncation in ABRL's validation responses at initialization and every 100 training steps, counting a response as truncated if it reaches the maximum generation token limit. As shown in Table 17, the truncation rate remains below 10% at all evaluated checkpoints and below 1% from step 200 onward.

Table 17: **ABRL validation-response truncation on TSP.**

Step	Truncation rate
0	3.88%
100	9.86%
200	0.17%
300	0.02%
400	0.00%
500	0.05%
600	0.00%
700	0.50%
800	0.00%
900	0.21%
1000	0.00%

Extended Sort comparison. Over the shared 600-step training interval, ABRL outperforms ADR at every evaluation from steps 300 through 600, scoring 0.4152 versus 0.3223 at step 600.

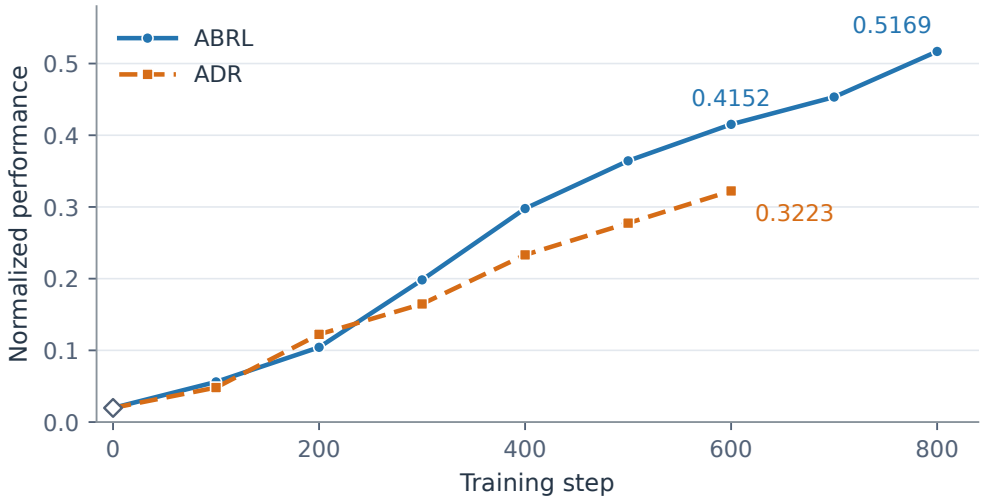

Figure 13: **Sort performance during extended training.** ADR and ABRL are evaluated on the same fixed set every 100 steps, with a shared initial-policy evaluation at step 0. ADR runs to step 600 and ABRL to step 800.

C.10 Non-CO evaluation protocols

GSM-Infinite. We adapt the Hard configuration of GSM-Infinite (Zhou et al., 2025) with graph-depth parameter $d = 3$, using the generator's operation count (OP) as the difficulty variable. We retain its natural-language problems and generated distractors, without additional long-context padding or the Symbolic setting.

To construct an instance at a target OP, we sample a graph skeleton and use dynamic programming to count feasible dependency combinations. We then sample dependencies in proportion to these counts so that the resulting graph has exactly the target OP. This replaces reliance on generating complete graphs and rejecting those with mismatched operation counts. We check that the answer-dependency graph is acyclic, its dependencies are consistent, and every node counted toward OP contributes to the answer.

Initial values and additional additive constants are sampled uniformly from the integers 1–20, and additional multiplicative constants from 2–10. Intermediate values in the answer-dependency graph and final answers must be integers in $[0, 10^7]$, increasing the original upper bound of 10^3 while retaining nonnegativity. Invalid numerical assignments are rejected and resampled, with at most 64 attempts per graph; values are neither clipped nor reduced modulo a constant. The 2–10 range applies only to sampled multiplicative constants, not to previously computed factors in products.

Reference solutions use indexed variables $v_1, v_2, \ldots$ to avoid exhausting a fixed alphabet. Both forward and reverse questions are retained. For reverse questions, we screen candidate unknowns by propagating affine expressions $ax + b$, where x is the unknown input and a, b are coefficients. A nonzero final coefficient ensures a unique solution. Independent exact-arithmetic checks verify consistency among the problem statement, computation trace, answer, and OP.

Prompts include semantic clarifications and require a single final integer enclosed in `<answer>` and `</answer>` tags. Training covers OP 10–100, with 5,000 instances per OP and 455,000 instances in total. Uniform RL and ABRL are evaluated after 100 training steps. The GSM-Infinite performance in Table 6 is the mean exact-answer accuracy over eight responses per evaluation instance (avg@8), reported in $[0, 1]$.

Integer program execution. Motivated by code-execution reasoning benchmarks such as CoRE (Gao et al., 2026), we construct a procedurally generated integer-program execution environment. Each program maintains five integer variables and executes a loop containing K randomly com-

posed blocks for five iterations. Blocks are drawn from 30 predefined templates, giving $5K$ block executions per program. Stored states and block outputs are restricted to integers in $[0, 1000]$, with a modulus between 2 and 1001 for each block.

Two execution implementations cross-check the generated trajectories and final answers. Models are prompted to return the final five integers within `<answer>` and `</answer>` tags. Evaluation performance is the mean exact-match accuracy. Intermediate states are retained for data verification but are not used to assess the model's reasoning.

We use K to control execution size rather than assume that every larger program is harder. Block templates may contain different numbers of primitive operations. Increasing K also increases input length, so this setting does not isolate execution complexity from context length. The evaluation concerns final-state prediction within the generated program family; it does not include CoRE's intermediate-state or equivalent-implementation evaluations.

CogniLoad. CogniLoad evaluates state tracking through a sequence of conditional updates (Kaiser et al., 2026). We construct training and validation sets using the official generation rules for initial states, conditions, state updates, distractors, and reference answers. We fix the generator's intrinsic difficulty parameter at $d = 5$ and the target proportion of statements involving the queried person at $\rho = 0.5$, following the official integer-rounding rule. The curriculum varies N, the number of statements, over every integer from 10 to 100. For each N, we generate 1,000 training instances and 10 validation instances using reproducible seeds determined by the split, N, and instance index. We remove duplicates and independently verify the reference answers by replaying the state transitions.

We retain the generated problem text and replace the introductory instructions to clarify sequential execution and require the queried final attribute value in a single `<answer>...</answer>` block. Instances are neither selected by model performance nor shortened to meet a token limit. Performance is the mean accuracy of the predicted final attribute values. These experiments use our generated data splits and adapted prompting protocol.

D Cross-Size Transfer and Gradient Alignment

D.1 Training on smaller sizes and evaluating at an untrained size

The restricted-range experiment trains Qwen3-8B on TSP instances with $n = 10$–30 and evaluates on size $n = 50$. Validation is recorded every ten training steps. We report results over steps 0–150, matching Figure 6a.

Performance follows Equation 11: invalid responses contribute zero, and valid tours contribute $1/(1 + g_{\text{ref}})$. Validity is the fraction of all responses that encode a valid tour. Mean reference gap is computed only over valid responses, so it measures quality conditionally on validity. Table 18 reports unsmoothed measurements at selected checkpoints. At step 150, both validity and conditional tour quality improve relative to initialization.

Table 18: **Unsmoothed transfer measurements at $n = 50$.** Selected training steps within 0–150 are reported below. Only sizes $n = 10$–30 are used for training.

Training step	Performance ↑	Validity	Mean gap ↓
0	0.0310	12.50%	3.2340
50	0.0376	10.00%	1.8136
100	0.1454	25.00%	0.7334
120	0.3063	51.25%	0.6956
150	0.2520	38.75%	0.5496

D.2 GRADIENT AGGREGATION AND DESCRIPTIVE ALIGNMENT SCORES

The gradient diagnostic uses a separate Qwen3-8B ABRL (CR5) checkpoint at step 100. Per-size gradients are stored through a common-seed CountSketch projection into $d = 65{,}536$ dimensions. The alignment diagnostic is computed from these projected gradients.

Let $\widetilde{g}_n$ denote the projected gradient at size n. We group sizes into nine bins, $\mathcal{B}_1 = \{10, \ldots, 20\}$, $\mathcal{B}_2 = \{21, \ldots, 30\}$, through $\mathcal{B}_9 = \{91, \ldots, 100\}$, and compute

$$\overline{g}_i = \frac{1}{|\mathcal{B}_i|} \sum_{n \in \mathcal{B}_i} \widetilde{g}_n, \qquad c_{ij} = \frac{\overline{g}_i^\top \overline{g}_j}{\|\overline{g}_i\| \, \|\overline{g}_j\|}. \tag{21}$$

Per-size gradients are averaged before normalization, with equal weight for each size. This differs from averaging pairwise cosines between individual sizes. For bin separation ℓ, let $m_\ell = 9 - \ell$ be the number of bin pairs. Their mean alignment is

$$C(\ell) = \frac{1}{m_\ell} \sum_{i=1}^{m_\ell} c_{i,i+\ell}, \qquad \ell = 1, \ldots, 8. \tag{22}$$

One unit on the horizontal axis denotes one bin of separation, roughly ten cities, rather than an exact node difference. The first bin contains eleven sizes and each remaining bin contains ten.

Random-direction baseline and standardization. As a geometric reference, suppose that the nine bin directions are independent and uniformly distributed on the unit sphere in $\mathbb{R}^d$. Under this null, each pairwise cosine has mean zero and variance $1/d$, and distinct pairwise cosines are pairwise independent (Cai et al., 2013). Consequently,

$$\mathbb{E}_0[C(\ell)] = 0, \qquad \mathrm{Var}_0[C(\ell)] = \frac{1}{d\, m_\ell}, \qquad Z_\ell = \sqrt{d\, m_\ell}\, C(\ell). \tag{23}$$

Thus Z_ℓ measures signed mean alignment relative to the stated geometric reference. Figure 6b displays this descriptive sequence after the smoothing described below. Table 19 reports the unsmoothed Z_ℓ values.

Table 19: **Raw Z values by size-bin separation.** The Z scores standardize the mean alignment of m_ℓ bin pairs against the random-direction geometric reference.

Bin separation ℓ	Bin pairs m_ℓ	Raw Z_ℓ
1	8	27.1272
2	7	29.9331
3	6	12.1677
4	5	10.1844
5	4	7.8677
6	3	−4.2537
7	2	−0.1762
8	1	−1.3928

The reference distribution describes random directions, not variability across training runs or rollout samples. The Z_ℓ values are therefore descriptive geometric comparisons rather than statistical significance tests. In addition, larger separations use fewer pairs and different bin combinations, so changes in Z_ℓ cannot be attributed solely to distance. This diagnostic comes from a separate checkpoint and complements the behavioral transfer experiment.

Display smoothing. Both panels in Figure 6 use a centered three-point moving average of their respective plotted sequences. Each endpoint uses the two available points within the displayed range. For the right panel, we first compute each raw Z_ℓ and then smooth the Z sequence. Dashed reference lines and all numbers reported in the text and tables use unsmoothed measurements.

E GRADED AND BINARY FEEDBACK: TRAINING TRAJECTORIES

Task rewards. The two recorded Qwen3-8B runs train and evaluate on TSP sizes $n = 4, \ldots, 20$. We compare steps 0–800, with evaluations every ten steps on the same instances. Both trajectories use the continuous performance metric in Equation 11, with invalid responses contributing zero, regardless of the training reward. Let $C(y)$ be the length of a valid tour and $C^{\mathrm{ref}}(x)$ the reference length stored with the instance, so that $g_{\mathrm{ref}}(x, y) = [C(y) - C^{\mathrm{ref}}(x)]/C^{\mathrm{ref}}(x)$. The graded task reward is $r_{\mathrm{graded}}(x, y) = 1/(1 + g_{\mathrm{ref}}(x, y))$ for a valid tour and zero otherwise. The binary task reward is one for valid tours satisfying the following exact-success criterion and zero otherwise. The criterion includes the evaluator's numerical tolerance.

Early similarity and later fluctuations. Table 20 summarizes the raw validation scores. The first 100 steps have similar average performance in the two runs. Over steps 400–800, the graded-reward run has both higher average performance and smaller changes between successive validations. We quantify the latter descriptively by the mean absolute change between consecutive ten-step evaluations within the indicated window.

Table 20: **Unsmoothed summaries of the reward comparison.** Early and late windows contain 11 and 41 validation checkpoints, respectively. Mean absolute change uses the 40 adjacent validation pairs within steps 400–800. All values use the common performance metric and are reported without display smoothing.

Statistic	Graded reward	Binary reward
Initial performance	0.0706	0.0709
Mean performance, steps 0–100	0.5002	0.5049
Mean performance, steps 400–800	0.9113	0.6782
Mean absolute change, steps 400–800	0.0120	0.1016
Performance at step 800	0.9509	0.6777

Display smoothing. Figure 7 uses a centered three-point moving average of the validation sequence, with two points at each endpoint. Only values within steps 0–800 enter the smoothing windows. The step-800 scores cited in the main text are the raw endpoint measurements, rather than their smoothed display values. Smoothing is used only for presentation; all statistics in Table 20 use the raw trajectories.

F TRAINING LOCATION AT A SHARED CHECKPOINT

Setup. All three continuations start from the model weights of Qwen3-8B after 100 steps of ABRL training, when the curriculum center is $n = 46$. Each continuation trains for 30 additional steps at a fixed TSP size: $n = 20$, $n = 46$, or $n = 100$.

Shared evaluation. We evaluate the starting policy and all three continuations on the same held-out instances over $n = 10$–100. Performance follows Equation 11, averaging $1/(1 + g_{\mathrm{ref}})$ for valid responses and zero for invalid responses. Validity is the fraction of valid responses; mean gap is computed over valid responses.

Table 21: **Additional diagnostics for the fixed-size continuations.**

Policy	Performance ↑	Validity ↑	Mean gap ↓
Shared checkpoint	0.3211	54.76%	1.4337
Train $n = 20$	0.3341	51.63%	1.2052
Train $n = 46$	0.4863	75.00%	0.6344
Train $n = 100$	0.2608	99.69%	3.6100

Performance beyond the continuation sizes. Table 22 separates performance by evaluation region. The final column covers the 88 sizes excluding $n \in \{20, 46, 100\}$ from $10, \ldots, 100$. After training at $n = 46$, performance on it increases from 0.3178 to 0.4870, and 79 of its 88 per-size averages exceed the starting values.

Table 22: **Performance across evaluation regions.** The final column excludes all three continuation sizes from every policy's evaluation. Bold marks the highest value in each column.

Policy	$n = 10$–30	$n = 31$–60	$n = 61$–100	Other sizes
Shared checkpoint	0.7934	0.3572	0.0460	0.3178
Train $n = 20$	**0.8231**	0.3772	0.0450	0.3310
Train $n = 46$	0.8130	**0.6031**	**0.2271**	**0.4870**
Train $n = 100$	0.4717	0.2360	0.1687	0.2613

Validity and solution quality. The $n = 100$ branch raises overall validity to 99.69%, but its mean reference gap rises to 3.6100 and performance falls to 0.2608. At the training size itself, all responses are valid, yet mean gap is 5.8112. This branch nevertheless improves performance over $n = 61$–100.

Training at the center $n = 46$ gives the largest overall improvement among the three locations at this stage, although gains are not uniform across sizes.

G Compute-Matched TSP Comparison

Because the curricula induce different response lengths, equal training steps do not imply equal generation budgets. We therefore compare the TSP training trajectories as a function of cumulative rollout compute, obtained by summing the generated tokens across training rollouts. Figure 14 tests whether ABRL's advantage persists after accounting for differences in response length.

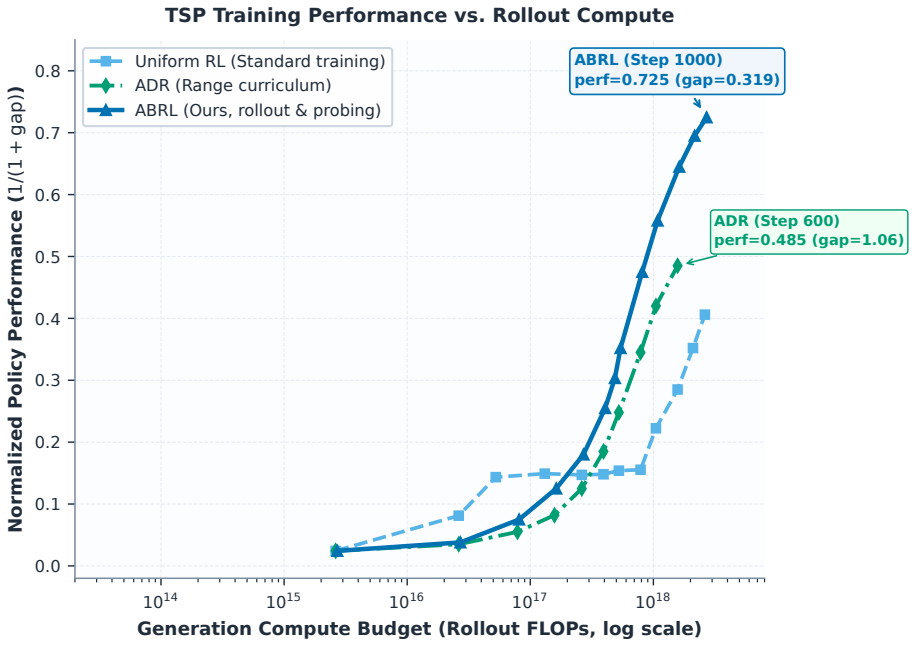

Figure 14: TSP training performance as a function of cumulative rollout compute. Curves show Uniform RL, ADR, and ABRL.

H Additional Limitations and Scope

The initial policy's best-of-k performance is a finite-budget reference, and exceeding it does not characterize the policy's full support. TSP gaps are measured against LKH-3 heuristic tours (Helsgaun, 2017) and do not certify optimality.

Intrinsic statistics depend on the generation protocol. We use complete reasoning spans for length estimation and compare the transition with external task measurements. These checks support the estimator in the studied settings; broader use requires an appropriate difficulty coordinate and validation of the signal. The mechanism analysis concerns the availability and use of relative feedback rather than a guarantee that each boundary instance produces a beneficial update.