BEST-OF-DEPTH: LEARNING EXPLORATORY DYNAMICS IN LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped Transformers increase inference-time computation through repeated latent-state updates with shared parameters. We propose training these updates to increase the utility of candidate solutions along a single trajectory. Our method, Best-of-Depth (BoD), minimizes the loss of the best prediction across depths, encouraging a match to the target anywhere within the depth budget without architectural changes. Experiments on N-Queens completion and graph coloring show improved solution discovery under greedy decoding relative to training that supervises only the prediction at the maximum recurrent depth. On N-Queens, new solutions continue to emerge beyond the training depth before discovery saturates. Code-generation experiments also show higher success across evaluated depths than maximum-depth training. Fixed recurrent-step-budget comparisons favor combining depth with multiple trajectories. These findings show how training a recurrent trajectory as a candidate set enables depth-wise solution discovery.

1 INTRODUCTION

Inference-time computation has become an important means of improving the capabilities of large language models (Snell et al., 2025; Brown et al., 2024). Best-of-N sampling and verifier-guided candidate generation convert additional computation into better performance by generating and evaluating multiple candidate solutions (Cobbe et al., 2021; Lightman et al., 2024; Yao et al., 2023). In parallel, looped Transformers increase inference-time computation without adding parameters by repeatedly applying a shared transformation to a latent state (Dehghani et al., 2019; Giannou et al., 2023; Yang et al., 2024). Recent work shows that increasing recurrent depth can improve reasoning (Saunshi et al., 2025; Geiping et al., 2025). Research in this area also addresses adaptive compute allocation (Graves, 2016; Banino et al., 2021), depth extrapolation (Schwarzschild et al., 2021; Bansal et al., 2022), and stable recurrent computation (Kuo et al., 2026; Li et al., 2026).

The training objective can influence how effectively a model uses additional inference-time computation. Inference-aware Best-of-N training incorporates the inference procedure into learning (Chow et al., 2025), while explorative modeling trains on the candidate that best matches the target (Gladstone et al., 2026). Both approaches shape candidate generation through the learning objective. This perspective suggests a new role for recurrent depth. For tasks with multiple valid solutions or access to an external verifier, exploring alternative candidates can be valuable alongside refining a single prediction. We therefore ask: *Can we train each recurrent model so that as much of its trajectory as possible consists of useful prospective solutions?*

Simply increasing recurrent computation does not guarantee that a trajectory will visit distinct useful candidates. Recent work promotes convergence toward fixed points to stabilize recurrent dynamics (Yang et al., 2026) or uses convergence to determine when computation should stop (Movahedi et al., 2026). Stable convergence alone does not determine which useful alternatives are encountered along the trajectory. A common approach to training looped Transformers is to supervise only the prediction at the maximum recurrent depth (Dehghani et al., 2019; Saunshi et al., 2025). We refer to this approach as Last-Depth training. This objective provides no direct incentive for intermediate predictions to form useful alternatives. Unlike independently sampled candidates, predictions at different depths are also coupled: each latent state is generated from the preceding one. This motivates a learning objective that evaluates these coupled predictions as a candidate set.

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.

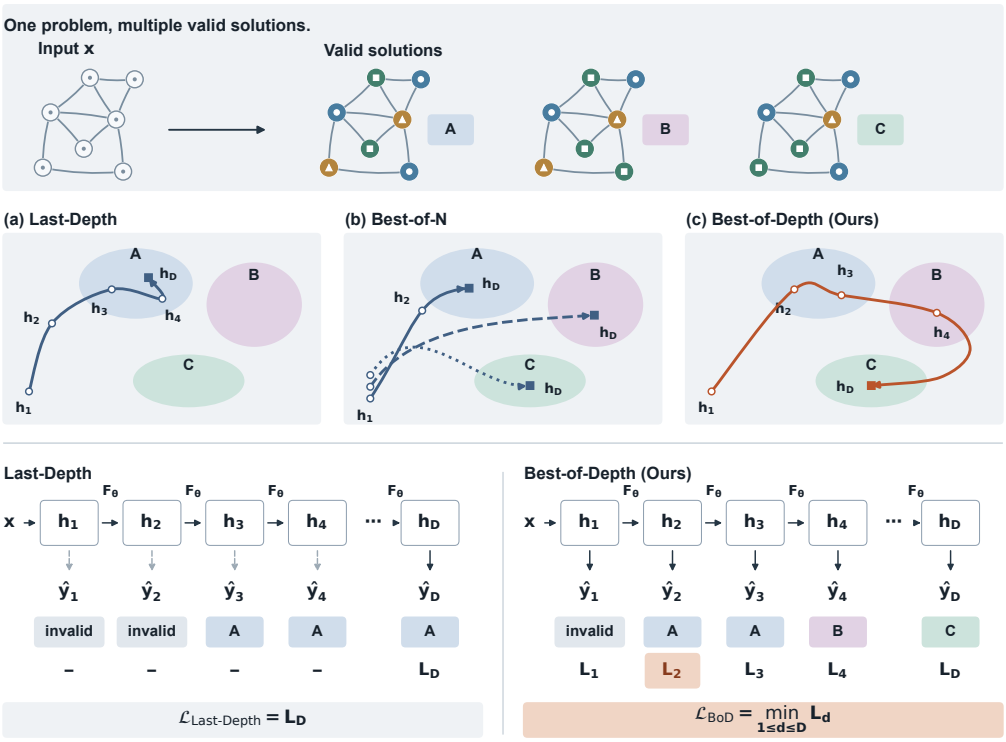

Figure 1: **Learning alternative solutions along one recurrent trajectory. Top:** One input admits multiple valid solutions. **Middle:** Schematic latent trajectories illustrate (a) repeated predictions under Last-Depth, (b) independent candidates from Best-of-N, and (c) distinct solutions along one BoD trajectory; shaded regions decode to A–C. **Bottom:** Both models reuse F_θ across depths. Last-Depth trains on L_D; BoD trains on the lowest loss across depths (highlighted L_2 for target A).

We introduce **Best-of-Depth (BoD)** to train recurrent trajectories as sets of candidate solutions. The objective is defined by the minimum prediction loss across recurrent depths (Figure 1). A shared decoder reads out candidates along the trajectory, and training rewards matching the target at any depth within the budget. Additional depth thus provides further opportunities to discover a useful solution. BoD requires no architectural modification or depth-specific labels. When multiple valid targets are presented for the same input, predictions at different depths can match different solutions. The objective is related to multiple-hypothesis learning (Guzmán-Rivera et al., 2012; Lee et al., 2016; Bhattacharyya et al., 2018). In BoD, the candidates are successive readouts of a shared recurrent computation, coupled through the latent dynamics.

Our contributions are a simple objective for learning exploratory recurrent trajectories and an empirical study of their ability to discover multiple valid solutions. On N-Queens completion and graph coloring, where all valid solutions can be enumerated, BoD improves greedy Success@Depth and Coverage@Depth relative to Last-Depth training. Code-generation experiments with a pre-trained looped language model (Geiping et al., 2025) also show higher success across evaluated depths. Further experiments demonstrate discovery of solutions withheld from the teacher pool and gains beyond the training depth. Comparisons under a fixed recurrent-step budget favor combining depth with multiple trajectories. Analyses of solution discovery, latent-state dynamics, and loss-minimizing depths characterize how these gains relate to the learned trajectories.

2 Related Work

Recurrent Computation and Looped Transformers. Looped Transformers reuse layers across depth; the Universal Transformer also supports adaptive halting (Dehghani et al., 2019). Subsequent work studies algorithmic capabilities (Giannou et al., 2023; Yang et al., 2024; Gatmiry et al., 2024) and latent reasoning (Saunshi et al., 2025). At language-model scale, Geiping et al. (2025) study

reasoning gains from increased recurrent depth. Flexible depth allocation uses learned exits (Zhu et al., 2026), token-level routing (Bae et al., 2025b), or shortcut consistency (Jeddi et al., 2026). Stable depth extrapolation motivates constraints on recurrent dynamics and depth sampling (Prairie et al., 2026), or depth-dependent residual scaling (Li et al., 2026).

Training supervises final outputs at fixed (Dehghani et al., 2019; Saunshi et al., 2025) or sampled (Geiping et al., 2025) depths, recurrent windows (Yang et al., 2024), or intermediate outputs for early exiting (Bae et al., 2025a); other objectives weight losses by exit probabilities (Zhu et al., 2026). Unlike intermediate reasoning-token supervision (Fan et al., 2026), BoD needs only output targets and prescribes no solution order.

Test-Time Candidate Generation and Compute Scaling. Additional inference compute supports candidate generation and evaluation. Repeated sampling raises the chance of at least one correct candidate, while identifying it remains a separate problem (Brown et al., 2024). Best-of-N selects candidates with a verifier (Cobbe et al., 2021); self-consistency aggregates answers across reasoning paths (Wang et al., 2023). Process supervision assesses intermediate steps (Lightman et al., 2024), while tree-structured methods evaluate partial solutions to guide further expansion (Yao et al., 2023; Snell et al., 2025).

Snell et al. (2025) compare parallel sampling with sequential revisions, finding that the preferred allocation depends on difficulty and budget. Their revisions condition on previous textual answers. Our fixed recurrent-step comparison instead allocates compute between recurrent latent updates and multiple trajectories (Section 4.5.1), with external candidate evaluation.

Learning to Generate Multiple Hypotheses. Multiple Choice Learning scores a prediction set through its best member's loss against the target (Guzmán-Rivera et al., 2012). Stochastic MCL trains only the lowest-loss predictor per example, enabling deep ensembles to specialize without a predefined data partition (Lee et al., 2016). Best-of-Many uses the best reconstruction among latent samples in a variational objective for sequence prediction (Bhattacharyya et al., 2018). The forward formulation of Explorative Modeling learns from the candidate closest to a target, studying exploration as a scaling axis across generative tasks (Gladstone et al., 2026).

Chow et al. (2025) align imitation and reinforcement learning with verifier-based Best-of-N selection. BoD instead minimizes supervised prediction loss over recurrent readouts: later hypotheses depend on earlier states, and the selected loss trains the shared update and decoder.

3 Best-of-Depth

3.1 Looped Transformers

A looped Transformer encodes an input x and repeatedly applies a shared transformation:

$$h_0 = \mathrm{Enc}_\theta(x), \qquad h_d = F_\theta(h_{d-1}; x), \qquad p_d = \mathrm{Dec}_\theta(h_d), \quad d = 1, \ldots, D. \tag{1}$$

One application of F_θ is a *loop*, and d denotes *recurrent depth*. A shared decoder reads the prediction distribution p_d from each state. Let $y \in \mathcal{Y}^\star(x)$ be a valid target, $\hat{y}_d(x)$ the greedy prediction, and $L_d(x, y) = \ell(p_d, y)$ the per-example loss. Input and parameter dependence are omitted when unambiguous.

Supervising the final recurrent output is an established approach in fixed-depth looped models (Dehghani et al., 2019; Saunshi et al., 2025). We call this *Last-Depth training*: only the prediction at the maximum training depth contributes to the loss,

$$\mathcal{L}_{\text{Last-Depth}} = L_D. \tag{2}$$

3.2 Best-of-Depth Objective

At training depth $D = D_{\text{train}}$, let $\mathcal{D} \subseteq \{1, \ldots, D\}$ contain the supervised readouts, including D. BoD minimizes

$$\mathcal{L}_{\text{BoD}}(x, y) = \min_{d \in \mathcal{D}} L_d(x, y). \tag{3}$$

Each L_d averages losses over predicted positions, so BoD selects one depth for each entire target and trains on its loss, breaking exact ties at random.

For an autoregressive target, the teacher-forced loss is

$$L_d(x, y) = -\frac{1}{T} \sum_{t \in \mathcal{T}} \log p_d(y_t \mid x, y_{<t}), \qquad T = |\mathcal{T}|, \tag{4}$$

where $\mathcal{T}$ contains target-token positions and p_d is the depth-d next-token distribution.

For the same readouts, Mean-Depth supervises all depths and Random-Depth samples one per example:

$$\mathcal{L}_{\text{Mean-Depth}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} L_d, \qquad \mathcal{L}_{\text{Random-Depth}} = L_{\tilde{d}}, \quad \tilde{d} \sim \text{Uniform}(\mathcal{D}). \tag{5}$$

Soft-Depth smoothly relaxes the minimum:

$$\mathcal{L}_{\text{Soft-Depth}}(x, y) = -\tau \log \left(\frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \exp[-L_d(x, y)/\tau] \right), \qquad \tau > 0. \tag{6}$$

Lower-loss depths receive greater gradient weight; the limits $\tau \to 0$ and $\tau \to \infty$ recover BoD and Mean-Depth, respectively. All methods support the same inference readouts.

3.3 Interpreting the Best-of-Depth Objective

For exact-match loss, write $\mathcal{L}^{0\text{–}1}_{\text{BoD}} = \min_{d \in \mathcal{D}} \mathbf{1}[\hat{y}_d \neq y]$. The first two results connect this objective to target discovery and solution coverage.

Proposition 1 (First-hit interpretation)**.** *Let $\tau_y = \min\{d \in \mathcal{D} : \hat{y}_d = y\}$, with $\tau_y = \infty$ if no readout matches y. Then*

$$\mathbb{E}[\mathcal{L}^{0\text{–}1}_{\text{BoD}}] = \Pr(\tau_y > D). \tag{7}$$

BoD thus evaluates whether the target is reached anywhere within the depth budget.

Proposition 2 (Connection to solution coverage)**.** *Fix x and $S_D(x) = \{\hat{y}_d(x) : d \in \mathcal{D}\}$ independently of the target $Y \sim \text{Uniform}(\mathcal{Y}^\star(x))$. Then*

$$\mathbb{E}_Y[\mathcal{L}^{0\text{–}1}_{\text{BoD}} \mid x] = 1 - \frac{|S_D(x) \cap \mathcal{Y}^\star(x)|}{|\mathcal{Y}^\star(x)|} = 1 - \text{Coverage@Depth}(D; x). \tag{8}$$

Minimizing this risk maximizes coverage; repeated candidates add none.

For cross-entropy, the following construction separates the objectives under shared recurrent computation.

Proposition 3 (Separation in solution discovery)**.** *There exist a multi-solution task, a shared recurrent model with depth budget D and factorized readouts, and a common symmetric initialization and SGD schedule such that, under cross-entropy training with uniformly resampled valid targets, greedy coverage satisfies*

$$\lim_{n \to \infty} \text{Coverage@Depth}(D; \theta^{(n)}_A) = \begin{cases} 0, & A \in \{\textit{Last, Mean, Random, Soft}\}, \\ 1, & A = \textit{BoD}, \end{cases} \tag{9}$$

almost surely, where $\theta^{(n)}_A$ denotes the parameters after n updates with objective A. The Soft-Depth result holds for every fixed $\tau > 0$; BoD selects one minimizing depth at ties.

In this prescribed cyclic recurrence, the other objectives retain or recover a symmetric predictor with invalid greedy outputs. BoD breaks symmetry by selecting one depth, allowing different targets to reinforce the same latent solution through different recurrent phases.

Proofs are in Appendices D.1 and D.3, including a numerical illustration of Proposition 3. Cross-entropy coverage bounds are given in Appendices D.2 and D.4.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Compared methods and candidate generation. We compare BoD, Last-Depth, Mean-Depth, Random-Depth, and Soft-Depth ($\tau = 0.25$; Section 3). Within each controlled task, all use the same output head and a width-128 Transformer with two shared layers per loop and eight attention heads. Unless stated otherwise, training uses uniformly resampled valid targets at $D_{\text{train}} = 16$; evaluation uses greedy readouts through $D_{\text{test}} = 64$, averaged across three seeds. Multiple trajectories pool all intermediate predictions. Section 4.4 specifies autoregressive generation; Appendix A gives implementation details.

Evaluation metrics. For the distinct candidate set $S_D(x)$ (Section 3.3), with $D = D_{\text{test}}$, we report

$$\text{Success@Depth}(D; x) = \mathbf{1}[S_D(x) \cap \mathcal{Y}^\star(x) \neq \varnothing], \tag{10}$$

$$\text{Coverage@Depth}(D; x) = \frac{|S_D(x) \cap \mathcal{Y}^\star(x)|}{|\mathcal{Y}^\star(x)|}. \tag{11}$$

Both are problem-averaged percentages. Coverage is our primary controlled-task metric; Success requires only a validity verifier. Single-depth accuracy is the fraction of problems solved by the prediction at depth D. We abbreviate the cumulative metrics as Success@D and Coverage@D, or Success and Coverage for full candidate sets without a common depth budget. Appendix C details seed aggregation.

4.2 N-QUEENS COMPLETION

Settings. Each input fixes five queens on a 10×10 board; the model completes the board so that no two queens share a row, column, or diagonal. We represent the board with 100 cell tokens and train on 24,320 problems. Evaluation uses 1,000 multi-solution test problems with all valid completions enumerated. Distinct completed boards count as different solutions, while symmetry-equivalent inputs are kept within the same split (Appendix A.1).

Results. BoD outperforms all four objectives at depth 64, with 91.03% coverage and 99.03% success (Figure 2, left). Its largest gains occur near the training depth, followed by further discoveries beyond it and saturation by depth 64. The other objectives also benefit from deeper evaluation, but their trajectories recover fewer of each problem's valid completions. Thus, BoD improves both the chance of finding a solution and the fraction of alternatives recovered along one greedy trajectory.

4.3 GRAPH COLORING

Settings. The model assigns one of three colors to each vertex so that adjacent vertices have different colors. We train on 4,096 connected 10-vertex graphs and test on 512, with adjacency rows encoded as vertex tokens. Each graph has 2–128 enumerated solution families; colorings that differ only in color names count as one solution. Isomorphic graphs do not cross splits (Appendix A.2).

Results. BoD achieves 72.16% coverage at depth 64 while solving nearly all test graphs (Figure 2, right). Success is already close to saturation at the training depth, whereas coverage continues to increase beyond it: additional depth mainly recovers alternatives for graphs already solved. Soft-Depth also has high success but only 44.39% coverage, showing that similar success can conceal substantial differences in solution discovery. Development-selected Soft-Depth also has lower coverage than BoD (Appendix C.4).

4.4 CODE GENERATION

Settings. We fine-tune Huginn, a pretrained 3B-parameter looped language model (Geiping et al., 2025), on 20,000 Python examples from CodeFeedback (Zheng et al., 2024), with 1,000 validation examples. Each of the five objectives uses training depth 32; we also evaluate the pretrained model. On all 164 HumanEval and 500 MBPP problems (Chen et al., 2021; Austin et al., 2021), each model generates a complete program separately at each fixed depth in $\{4, 8, 12, 16, 24, 32\}$ using greedy decoding. Success@32 pools these six candidates. Tests score completed programs without

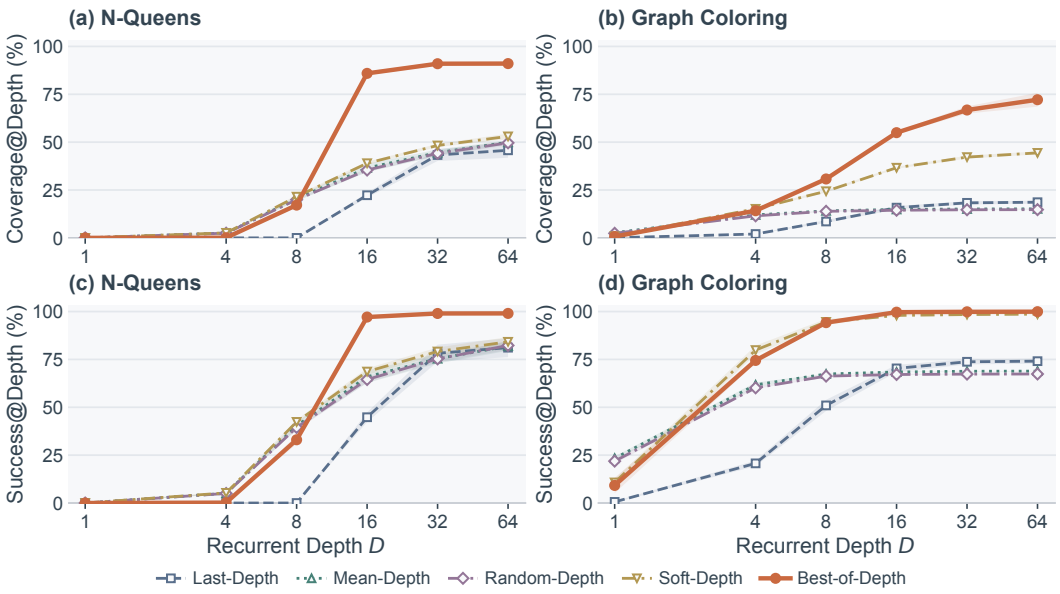

Figure 2: **Depth-wise solution discovery.** Coverage@Depth (top) and Success@Depth (bottom) for greedy trajectories trained at depth 16. Lines and bands show means and ± 1 sample SD across three training seeds.

guiding generation or repair; MBPP uses the prompt-visible public tests. Appendix A.3 gives the full protocol.

Results. BoD achieves the highest Success@32 on both benchmarks (Table 1), improving over Last-Depth from 34.8% to 37.8% on HumanEval and from 39.2% to 44.2% on MBPP. It also has the highest accuracy at depth 32, although Mean-Depth, Random-Depth, and Soft-Depth perform better at shallow depths. Pooling BoD's depth-specific programs solves problems missed at depth 32 alone, showing that shallower generation depths contribute useful candidates. These findings extend the benefit of BoD to autoregressive candidate generation across depths.

Table 1: Code-generation results (%). Base is the pretrained model; other rows correspond to fine-tuning objectives. Depth columns report accuracy for one greedy program per problem. Success@Depth at 32 combines all six evaluated depths $\{4, 8, 12, 16, 24, 32\}$. Bold values indicate the best result in each column.

	HumanEval					**MBPP**				
	Depth				Success@Depth	Depth				Success@Depth
Method	4	8	16	32	32	4	8	16	32	32
Base	0.0	12.8	24.4	25.6	30.5	1.2	17.0	29.0	29.8	39.8
Last-Depth	0.0	13.4	27.4	26.8	34.8	1.8	21.2	30.2	31.4	39.2
Mean-Depth	**6.7**	20.1	27.4	27.4	31.7	**10.4**	**26.0**	28.0	29.0	37.2
Random-Depth	4.9	20.1	26.8	28.7	30.5	8.8	25.8	29.6	29.6	37.4
Soft-Depth	6.1	**20.7**	26.2	26.8	32.9	9.4	25.2	29.2	30.8	39.4
Best-of-Depth	0.6	14.6	**29.3**	**29.9**	**37.8**	0.6	21.0	**30.6**	**33.2**	**44.2**

4.5 Ablation Studies

4.5.1 Width vs. Depth under a Fixed Recurrent-Step Budget

Figure 3(a, b) compares seven allocations of $B = ND = 64$ recurrent-block applications, with $D \in \{1, 2, 4, 8, 16, 32, 64\}$ and $N = 64/D$. Each method pools and deduplicates all intermediate

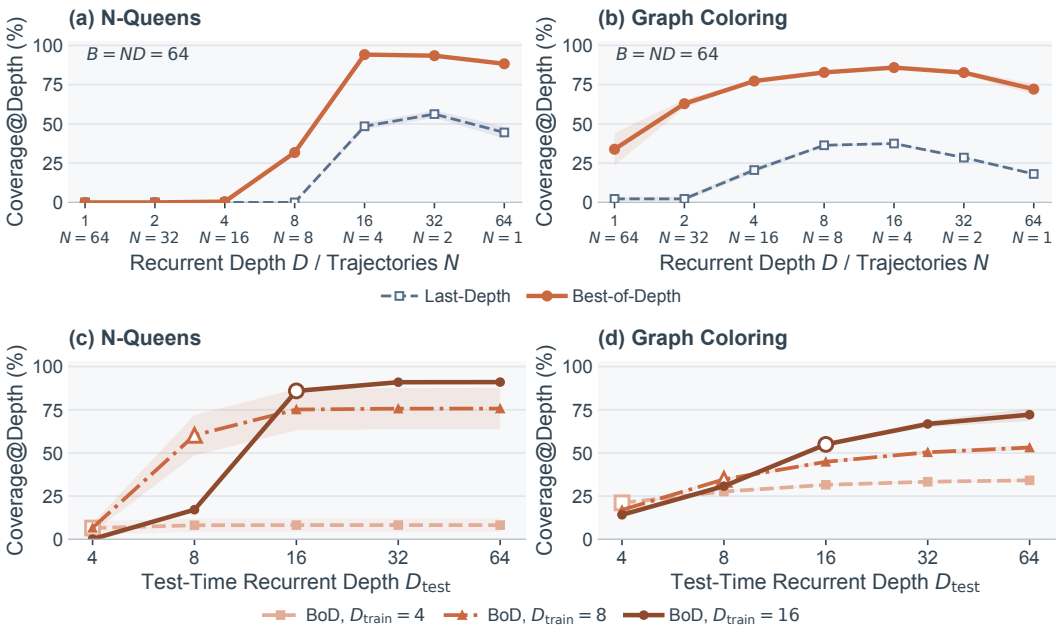

Figure 3: **Depth allocation and extrapolation.** (a, b) Coverage pooling intermediate predictions from randomized trajectories under $ND = 64$. (c, d) BoD with unperturbed greedy readouts; open markers denote training depth. Lines and bands show means and ± 1 sample SD across three seeds, averaging inference repeats within seeds in (a, b).

predictions from N trajectories. These use initial-state Gaussian perturbations for N-Queens and vertex permutations for graph coloring, including at $N = 1$. Appendix C gives the protocols.

BoD achieves its highest mean coverage at $(N, D) = (4, 16)$ on both tasks; Last-Depth peaks at $(2, 32)$ on N-Queens and $(4, 16)$ on graph coloring. These results favor combining depth with multiple trajectories.

4.5.2 Training Depth and Test-Time Extrapolation

We vary training depth to examine whether solution discovery continues beyond the supervised trajectory. Figure 3(c, d) evaluates BoD trained at $D_{\text{train}} \in \{4, 8, 16\}$ using greedy readouts through $D_{\text{test}} \in \{4, 8, 16, 32, 64\}$. All settings show further discoveries beyond the training depth, with diminishing gains. Models trained deeper reach higher coverage by depth 64, but can lag at shallower depths. Training depth thus shapes both the breadth of the candidate set and the depth at which alternatives become available. Success curves and numerical results are in Appendix C.

4.5.3 Discovering Solutions Beyond the Training Targets

We test discovery beyond supplied targets by restricting each graph's teacher pool to one, two, three, or four valid solutions, with an all-solutions control. The pools are nested and sampled uniformly; all conditions use the same training budget and 3,034 training graphs with at least five solutions (Appendix C.5). Evaluation on these same graphs measures *Seen Coverage*, the fraction of teacher-pool solutions recovered, and *Unseen Coverage*, the fraction of withheld valid solutions recovered, averaged per graph. Table 2 compares Last-Depth, Last-Depth with multiple sampled trajectories (Last-Depth + BoN), and BoD under a recurrent-step budget of 64, collecting intermediate predictions for all three. BoD has the highest unseen coverage in every restricted-pool condition, increasing from 29.4% with one training solution to 46.2% with four. These are familiar training inputs, not unseen graphs.

Table 2: Discovery beyond the training targets on graph coloring. Seen Coverage and Unseen Coverage (%) on 3,034 training graphs under $ND = 64$, reported as mean ± sample SD over three training seeds. All uses every valid solution as a training target candidate; Unseen Coverage is then undefined (N/A).

Training solutions	Last-Depth		Last-Depth + BoN		Best-of-Depth	
	Seen Coverage	Unseen Coverage	Seen Coverage	Unseen Coverage	Seen Coverage	Unseen Coverage
1	**99.8**±0.0	13.5±2.4	25.4±0.3	25.4±0.4	98.8±0.4	**29.4**±0.4
2	46.6±0.6	12.9±1.2	22.8±0.7	22.9±0.9	**94.2**±1.1	**37.0**±0.8
3	37.1±1.4	13.8±1.3	23.6±0.2	23.7±0.3	**91.3**±1.3	**42.2**±0.5
4	28.4±0.3	13.8±0.4	24.4±0.6	24.4±0.6	**87.7**±1.6	**46.2**±1.2
All	11.8±0.2	N/A	25.0±0.4	N/A	**67.4**±4.4	N/A

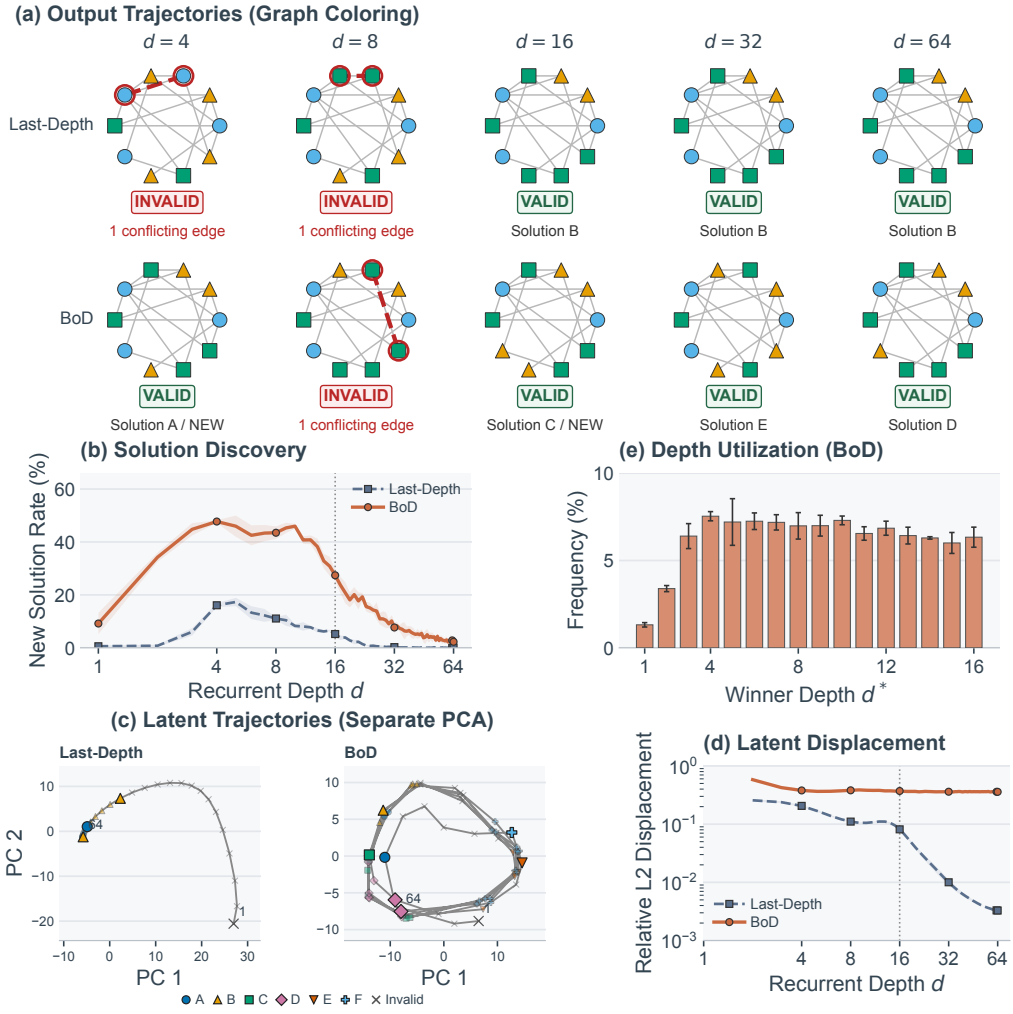

Figure 4: **Exploratory recurrent dynamics on graph coloring.** (a) Example outputs; NEW marks first appearances. (b) Fraction of graphs with a new valid solution. (c) Separate PCA projections for the same example and seed as (a); markers identify solutions or invalid outputs. (d) Relative state displacement. (e) Loss-minimizing depths. Bands (b, d) and error bars (e) show ±1 sample SD across three seeds, each averaging 512 graphs. Dotted lines mark training depth 16.

4.5.4 Comparison with Other Baselines

Table 15 in Appendix C.6 compares BoD with unshared and autoregressive Transformers, masked diffusion (Sahoo et al., 2024), HRM (Wang et al., 2025), TRM (Jolicoeur-Martineau, 2025), and an instance-level adaptation of Ouro's objective (Zhu et al., 2026). Task splits and update counts are shared, but architectures and candidate counts vary. BoD achieves the highest mean success and coverage on both tasks. HRM has similar N-Queens success but substantially lower coverage, illustrating the distinction between finding one solution and many. Ouro's adapted objective also yields lower coverage on the same looped backbone.

5 Understanding Exploratory Recurrent Dynamics

Figure 4 examines greedy graph-coloring trajectories under uniformly resampled teachers. Aggregate statistics use all 512 test graphs and three training seeds; Appendix B details the protocol.

BoD continues to discover new valid solutions around and initially beyond the training depth, where Last-Depth yields few new discoveries (Figure 4b). This metric counts both first solutions and subsequent alternatives. Discoveries diminish at larger depths, consistent with Figure 2. In the selected example (panel (a)), Last-Depth remains invalid through depth 14 and visits two solutions by depth 64, whereas BoD visits all six.

In the example's separate PCA projections (Figure 4c), Last-Depth approaches a smaller region, while BoD continues moving among states decoding to different valid solutions. Across the test set, BoD also retains larger relative state displacement around and beyond the training depth, whereas Last-Depth's declines markedly (panel (d)). Displacement is measured before projection relative to the preceding state's norm. Latent movement alone does not establish solution discovery.

BoD's loss-minimizing depths span all 16 supervised depths (Figure 4e), so intermediate depths supply the best predictions for different input–target pairs. This distribution weights graphs equally and valid targets uniformly within each graph. It does not imply that multiple depths specialize to distinct solutions within every graph; panels (a, b) measure solution discovery directly.

6 Discussion and Limitations

Although BoD constrains neither solution distances nor visitation order, the mean distance between consecutive discoveries on graph coloring is lower than under random orderings of the same solutions (Appendix B.1). This also occurs under Last-Depth, suggesting that recurrent dynamics may shape visitation order. Its relationship to BoD's broader discovery remains to be understood.

For autoregressive generation, we generate each sequence separately at a fixed depth. Future work could sample or branch across loops' next-token distributions, combining this with stopping, restarts, and width–depth allocation.

Our study covers small combinatorial problems and one pretrained language model, and discovery saturates with depth. Larger tasks and different models are needed to establish how broadly training and inference depth affect candidate discovery.

7 Conclusion

Best-of-Depth trains on the lowest-loss prediction along a looped Transformer's trajectory, allowing different depths to match different valid targets without architectural changes. On N-Queens completion and graph coloring, BoD improves greedy success and solution coverage, with further discoveries beyond the training depth. On familiar graph-coloring inputs, it also recovers solutions withheld from the teacher pool. Trajectory analyses show broader solution discovery alongside sustained latent movement and use of multiple intermediate depths. Fixed recurrent-step-budget comparisons favor combining depth with multiple trajectories, and code-generation experiments show higher success across programs generated at different fixed depths. These findings support training and evaluating recurrent trajectories as candidate sets, so that shared computation can produce complementary valid solutions across depth.

Reproducibility Statement

Appendix A describes the datasets, models, training objectives, and evaluation protocols. Appendix B specifies the trajectory analyses and example-selection criteria, Appendix C reports detailed results and aggregation procedures, and Appendix D provides the proofs. Code-generation hyperparameters in Table 4 describe all five fine-tuning objectives.

AI Use Statement

AI tools assisted with manuscript editing, mathematical derivations and proof review, LaTeX formatting, and compilation.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=WwpYSOkkCt.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=QuqsEIVWIG.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021. URL https://arxiv.org/abs/2107.05407.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2202.05826.

Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL https://arxiv.org/abs/1806.07772.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-N sampling in large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.15287.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. arXiv preprint arXiv:2606.31779, 2026. URL https://arxiv.org/abs/2606.31779.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15130–15152. PMLR, 2024. URL https://proceedings.mlr.press/v235/gatmiry24b.html.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Alexi Gladstone, Heng Ji, and Yilun Du. Explorative modeling: Unlocking a third pretraining axis and end-to-end generation. arXiv preprint arXiv:2607.27372, 2026. URL https://arxiv.org/abs/2607.27372.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. URL https://arxiv.org/abs/1603.08983.

Abner Guzmán-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems*, 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/hash/cfbce4c1d7c425baf21d6b6f2babe6be-Abstract.html.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RzYXb5YWBs.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. arXiv preprint arXiv:2510.04871, 2025. URL https://arxiv.org/abs/2510.04871.

Hsun-Yu Kuo, El Mahdi Chayti, Patrik Reizinger, Wieland Brendel, and Martin Jaggi. Stabilizing extrapolation in looped transformers via learned stochastic stopping. arXiv preprint arXiv:2606.29983, 2026. URL https://arxiv.org/abs/2606.29983.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1606.07839.

Shuzhen Li, Yifan Zhang, Jiacheng Guo, Quanquan Gu, and Mengdi Wang. DeepLoop: Depth scaling for looped transformers. arXiv preprint arXiv:2607.13491, 2026. URL https://arxiv.org/abs/2607.13491.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers. arXiv preprint arXiv:2606.18206, 2026. URL https://arxiv.org/abs/2606.18206.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models. arXiv preprint arXiv:2604.12946, 2026. URL https://arxiv.org/abs/2604.12946.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.07524.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.17416.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2106.04537.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. arXiv preprint arXiv:2506.21734, 2025. URL https://arxiv.org/abs/2506.21734.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2311.12424.

Xiao-Wen Yang, Ziyu Han, Xi-Hua Zhang, Wen-Da Wei, Jie-Jing Shao, Lan-Zhe Guo, and Yu-Feng Li. Stabilizing recurrent dynamics for test-time scalable latent reasoning in looped language models. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=tbdSfBMlbq.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.10601.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. OpenCodeInterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12834–12859, 2024. doi: 10.18653/v1/2024.findings-acl.762. URL https://aclanthology.org/2024.findings-acl.762/.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. arXiv preprint arXiv:2510.25741, 2026. URL https://arxiv.org/abs/2510.25741.

A Implementation Details

For N-Queens and graph coloring, Last-Depth, Mean-Depth, Random-Depth, Soft-Depth, and BoD use the same architecture and output head within each task. The readout set for Mean-Depth, Random-Depth, Soft-Depth, and BoD is $\mathcal{D} = \{1, \ldots, D_{\text{train}}\}$. Soft-Depth uses $\tau = 0.25$ except in the temperature sweep. Each presentation samples a teacher uniformly from the input's complete valid solution set, except in the withheld-solution ablation.

Table 3: Training settings for the N-Queens and graph-coloring objective comparisons.

Setting	Value
Recurrent transformation	Two shared Transformer layers per loop
Width / heads / FFN width	128 / 8 / 384
Output normalization	StableMax
Training depth D_{train}	16
Optimizer	AdamAtan2
Learning rate / batch size	10^{-4} / 768
Schedule	2,000 warmup updates, then constant learning rate
Weight decay / gradient clipping	1 / norm 1
EMA decay	0.999
Training updates	N-Queens: 20,000; graph coloring: 10,000
Training seeds	0, 1, 2

A.1 N-Queens Completion

Dataset. Inputs fix five queens on a 10×10 board. The training, development, and test splits contain 24,320, 400, and 2,000 inputs, respectively, with equal numbers of single- and multi-solution inputs in each split. All valid completions are enumerated, and D_4-equivalent inputs do not cross splits. Reported coverage and success use the 1,000 multi-solution test inputs. Distinct completed boards count as different solutions; outputs are not merged under D_4 symmetry.

Model and training. The model represents the board with 100 cell tokens and uses the shared recurrent architecture in Table 3. The objective comparisons train for 20,000 updates and evaluate EMA weights at that update count. Computation uses BF16 with FP32 state. Evaluation collects the intermediate predictions of one trajectory through depth 64.

A.2 Graph Coloring

Dataset. Inputs are connected graphs with 10 vertices, at least 10 edges, and 2–128 valid 3-coloring families. The training, development, and test splits contain 4,096, 512, and 512 graphs, respectively, with graph-isomorphic inputs separated across splits. All canonical solution families are enumerated; global color-name permutations identify the same family. Every test graph has multiple valid solutions. Coverage is computed using each graph's number of valid families and then averaged over graphs. The verifier checks edge constraints and color range independently of the sampled teacher.

Model and training. A projection of adjacency rows produces 10 vertex tokens. Only the 10 vertex colors contribute to the loss; trailing padding positions in the serialized representation are excluded. The model uses the shared configuration in Table 3, trains for 10,000 updates, and is evaluated using the EMA weights at that fixed update. The BoD objective minimizes graph-level mean cross-entropy over depths, after averaging over the valid output positions. Evaluation collects all greedy predictions from one trajectory through depth 64.

Temperature sweep. Figure 6 varies $\tau = 0.1, 0.2, \ldots, 1.0$ while retaining the graph-coloring configuration above. Within each training seed, the runs have matching recorded initialization and teacher-sampling hashes. Each evaluation collects all outputs from one 64-step trajectory. Metrics are averaged over three training seeds, with sample SD. Temperature is selected by mean development Coverage@64, with ties broken by development Success@64 and then smaller temperature.

Test scores do not select the temperature or checkpoint. BoD and Mean-Depth reference results use the same configuration.

A.3 Code Generation

Fine-tuning. We use the 3.565B-parameter Huginn model and Python examples from CodeFeedback (Geiping et al., 2025; Zheng et al., 2024). The training and validation sets contain 20,000 and 1,000 examples, respectively. Examples whose combined prompt and target exceed 1,024 tokens are excluded. Last-Depth and BoD use the same examples, teacher responses, data order, and number of updates. Table 4 summarizes the training settings.

Table 4: Shared fine-tuning settings for the five code-generation objectives.

Setting	Value
Model	Huginn 3.565B, shared recurrent parameters
Data	CodeFeedback Python examples; 20,000 train / 1,000 validation
Maximum sequence length	1,024 tokens, including prompt and target
Training duration	1 epoch, 625 updates
Effective batch size	32
Maximum training depth	$D_{\text{train}} = 32$
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0
Peak learning rate	5×10^{-7}
Schedule	Warmup–Stable–Decay; first 63 updates warm up, final 63 decay

The loss at each depth is the mean teacher-forced cross-entropy over the entire response, excluding prompt and padding tokens, as in Eq. 4. Last-Depth uses the loss at depth 32. For BoD, all depths from 1 to 32 are scored along one recurrent trajectory, and a single depth with the lowest response-level loss is selected for each example. This depth is then recomputed for backpropagation through its last at most eight recurrent updates; exact loss ties are broken randomly. The selected depth is shared by all response tokens in the example, not chosen separately for each token.

Mean-Depth averages response-level losses over depths 1 through 32; Random-Depth samples one depth uniformly per example, shared by all response tokens. Soft-Depth uses Eq. 6 with $\tau = 0.25$ and $\mathcal{D} = \{1, \ldots, 32\}$, applied to response-level losses. The loss is computed per example and then averaged over the batch. All five objectives use the dataset and optimization settings in Table 4, with the final checkpoint at update 625.

Generation and scoring. We evaluate all 164 HumanEval problems and all 500 MBPP problems. Each model generates one program greedily at each fixed depth in $\{4, 8, 12, 16, 24, 32\}$, giving six candidates per problem. There is no stochastic sampling, test-guided repair, or regeneration. Prompt and generated output together are limited to 1,024 tokens. Inference uses BF16 with KV caching and batches of at most 64, subdivided as needed. Each fixed-depth policy generates its own complete autoregressive sequence; candidates with diverging prefixes require separate states and caches.

HumanEval prompts contain the function signature and docstring. Scoring retains the generated function, helper functions, and imports, then executes the HumanEval tests. MBPP prompts contain the problem description and public tests, and scoring executes those same public tests. Consequently, the reported MBPP success measures public-test correctness, not correctness on an additional hidden test suite. One MBPP prompt exceeds the input limit; it is counted as incorrect and retained in the denominator of 500 for the completed evaluations.

Accuracy at a given depth is the fraction of problems whose single generated program passes all tests. The individual-depth columns in Table 1 report this fraction. Success@32 is the fraction solved by at least one of the six evaluated depths, which is oracle candidate-set success rather than an i.i.d. pass@k estimate. Evaluation tests are used only for scoring, with no feedback during generation. We do not report Coverage@Depth because valid programs cannot be exhaustively enumerated. The reported counts are re-aggregated from archived candidate-level test outcomes.

B Graph-Coloring Trajectory Analysis

Figure 4 uses the uniform-teacher Last-Depth and BoD checkpoints at update 10,000, trained at depth 16 and evaluated greedily through depth 64 without input permutations or initial-state noise. Re-inferred predictions match the archived predictions for all six runs. Solution IDs identify canonical color families, removing global color-label permutations. The new-solution indicator at depth d is

$$\mathbf{1}\big[\hat{y}_d(x) \in \mathcal{Y}^\star(x) \text{ and } \hat{y}_d(x) \notin S_{d-1}(x)\big], \tag{12}$$

where $S_0(x) = \varnothing$ and membership is evaluated using canonical color families. Discovery indicators are checked against the preceding decoded outputs and enumerated valid families. For each population curve, we first average over all 512 graphs within each seed and then report the mean and sample standard deviation across seeds 0, 1, and 2.

Panels (a) and (c) use test graph 4727 (local index 119), with six valid solution families, at training seed 1. This example was selected post hoc from archived candidates for an extended invalid prefix under Last-Depth, subsequent valid predictions, and multiple discoveries under BoD. Last-Depth is invalid through depth 14 and visits two distinct valid families by depth 64; BoD visits all six. Panel (a) shows depths 4, 8, 16, 32, and 64 with a common vertex layout. The example illustrates possible behavior, not its prevalence.

Panel (c) uses the actual recurrent core states immediately before the vocabulary head, with no additional output normalization. Each PCA flattens the ten vertex states, centers the resulting 64 vectors over depth, and fits two components separately for each model. All 64 states and transitions are retained; larger markers emphasize first appearances and the final depth. Cross-model projected distances and axis directions are not compared.

Relative displacement is computed before PCA as

$$\Delta_d = \frac{\|\tilde{h}_d - \tilde{h}_{d-1}\|_2}{\|\tilde{h}_{d-1}\|_2}, \qquad d = 2, \ldots, 64, \tag{13}$$

where $\tilde{h}_d$ concatenates the ten recurrent core states at depth d. Winner histograms minimize the graph-level mean loss over the 16 training depths, evaluating all valid teachers with equal weight within each graph, rather than weighting graphs by their number of solutions. Winner depths in this analysis use the first minimum for exact ties. The measured tie rate is zero in all six runs, so this convention gives the same histogram as the random tie-breaking rule used during training.

B.1 Locality of Solution Discovery

We conduct a post-hoc analysis of the order in which distinct valid solutions first appear in the same greedy trajectories. For each graph, we retain only first discoveries through depth 64, omitting invalid predictions and repeated solutions. Consecutive discoveries may therefore be separated by several recurrent steps. We include graphs with at least three discoveries, since reordering two solutions cannot change their distance.

The distance between two colorings is the number of vertices assigned different colors, minimized over all six global color-label permutations. We average this distance between consecutive first discoveries and compare it with the exact expectation under a uniform random permutation of the same discovered set. This expectation equals the mean distance over all unordered pairs of distinct solutions in that set. We then average each quantity equally over eligible graphs within each seed.

Table 5 shows shorter mean distances for the observed order in all three seeds for both objectives. Thus, locality is present in the order of discoveries and is not unique to BoD. The eligible graphs and discovered sets differ across methods, so these values do not isolate a difference in locality between the objectives. The analysis concerns average distances and does not imply that each discovery is the nearest unvisited solution.

C Additional Results

Unless stated otherwise, values are means and sample standard deviations over training seeds 0, 1, and 2. Each seed first averages the metric equally over evaluation problems. For randomized

Table 5: **Locality of first-discovery order on graph coloring.** Distances measure the number of changed vertices between consecutive first-discovered solutions, minimized over color-label permutations. Rows average over graphs with at least three discoveries. Random order gives the expected distance under a uniform permutation of the same discovered set on each graph.

Method	Seed	Graphs	Observed	Random order
Last-Depth	0	111	1.296	1.620
Last-Depth	1	115	1.370	1.671
Last-Depth	2	110	1.285	1.614
Best-of-Depth	0	458	2.369	2.630
Best-of-Depth	1	456	2.248	2.583
Best-of-Depth	2	463	2.369	2.602

restarts, the 32 inference repeats are averaged within each training seed before computing variation across training seeds; repeats are not treated as additional training runs.

C.1 Greedy Discovery across Depth

Tables 6–9 give the numerical results for Figure 2. The evaluation populations and checkpoint budgets are specified in Appendices A.1 and A.2. All models are trained at depth 16 and read out greedily at every depth up to the stated budget.

Table 6: N-Queens Coverage@Depth (%) on 1,000 multi-solution test problems. Models are trained at $D_{\text{train}} = 16$. Mean ± sample SD across three training seeds.

Method	@1	@4	@8	@16	@32	@64
Last-Depth	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	22.27 ±1.27	43.29 ±3.12	45.75 ±3.93
Mean-Depth	0.00 ±0.00	2.51 ±0.18	20.42 ±1.20	36.52 ±2.21	44.52 ±1.66	49.94 ±1.05
Random-Depth	0.00 ±0.00	2.46 ±0.11	19.93 ±0.53	35.42 ±0.91	44.15 ±1.24	49.63 ±1.34
Soft-Depth	0.00 ±0.00	2.63 ±0.37	21.53 ±0.82	38.99 ±0.79	48.26 ±0.83	53.01 ±1.47
Best-of-Depth	0.00 ±0.00	0.10 ±0.09	17.11 ±1.25	85.92 ±0.93	90.97 ±0.98	91.03 ±1.05

Table 7: N-Queens Success@Depth (%) on 1,000 multi-solution test problems. Models are trained at $D_{\text{train}} = 16$. Mean ± sample SD across three training seeds.

Method	@1	@4	@8	@16	@32	@64
Last-Depth	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	44.80 ±2.52	78.23 ±4.48	81.13 ±4.94
Mean-Depth	0.00 ±0.00	5.10 ±0.40	40.37 ±1.83	65.97 ±3.25	75.37 ±2.50	81.97 ±2.46
Random-Depth	0.00 ±0.00	5.00 ±0.20	39.43 ±1.34	64.50 ±1.61	75.27 ±1.72	82.30 ±2.35
Soft-Depth	0.00 ±0.00	5.30 ±0.70	42.37 ±1.11	68.67 ±1.69	79.13 ±2.10	84.20 ±2.49
Best-of-Depth	0.00 ±0.00	0.20 ±0.17	33.03 ±2.20	97.13 ±0.55	99.00 ±0.10	99.03 ±0.12

Table 8: Graph-coloring Coverage@Depth (%) on 512 test graphs. Models are trained at $D_{\text{train}} = 16$. Mean ± sample SD across three training seeds.

Method	@1	@4	@8	@16	@32	@64
Last-Depth	0.05 ±0.01	2.01 ±0.10	8.55 ±0.77	15.79 ±0.88	18.30 ±0.88	18.62 ±0.76
Mean-Depth	2.67 ±0.09	11.99 ±0.25	14.11 ±0.05	14.84 ±0.03	15.09 ±0.07	15.15 ±0.07
Random-Depth	2.38 ±0.20	11.51 ±0.15	13.87 ±0.29	14.42 ±0.25	14.68 ±0.15	14.75 ±0.16
Soft-Depth	0.88 ±0.25	15.17 ±0.59	24.36 ±0.66	36.64 ±0.53	42.18 ±0.77	44.39 ±0.80
Best-of-Depth	0.71 ±0.42	14.18 ±0.53	30.76 ±0.70	54.95 ±0.80	66.79 ±1.89	72.16 ±3.62

At depth 64, graph-coloring BoD achieves $99.93 \pm 0.11\%$ success and $72.16 \pm 3.62\%$ coverage. On the multi-solution N-Queens test population, the corresponding values are $99.03 \pm 0.12\%$ and $91.03 \pm 1.05\%$.

Table 9: Graph-coloring Success@Depth (%) on 512 test graphs. Models are trained at $D_{\text{train}} = 16$. Mean ± sample SD across three training seeds.

Method	@1	@4	@8	@16	@32	@64
Last-Depth	0.59 ±0.34	20.70 ±1.47	50.98 ±2.92	70.31 ±1.86	73.76 ±1.39	74.09 ±1.30
Mean-Depth	23.37 ±0.96	61.72 ±1.22	67.38 ±0.20	68.42 ±0.11	68.82 ±0.11	68.88 ±0.11
Random-Depth	21.88 ±1.28	60.16 ±0.52	66.28 ±0.63	67.12 ±0.30	67.32 ±0.11	67.38 ±0.00
Soft-Depth	10.74 ±1.79	79.88 ±2.34	94.60 ±1.58	97.98 ±0.11	98.44 ±0.20	98.63 ±0.20
Best-of-Depth	9.18 ±4.04	74.48 ±1.47	94.21 ±0.74	99.67 ±0.30	99.87 ±0.23	99.93 ±0.11

C.2 Width and Depth under a Fixed Recurrent-Step Budget

Tables 10 and 11 give the values plotted in Figure 3(a, b). All seven allocations use $ND = 64$, pooling valid predictions from every intermediate depth of all trajectories before computing per-problem coverage.

For N-Queens, each trajectory initializes the recurrent state with independent Gaussian noise, scaled by the encoded input:

$$h_0^{(j)} = 0.1\,\mathrm{RMS}(\mathrm{Enc}_\theta(x))\,\epsilon_j, \qquad \epsilon_j \sim \mathcal{N}(0, I), \tag{14}$$

where RMS is the root mean square over entries of the encoded input, which remains unchanged during initialization. The scale is selected from $\{0, 0.03, 0.1, 0.3\}$ by mean development coverage at $(N, D) = (4, 16)$ across both methods and all three seeds. The selected value, 0.1, is fixed before test evaluation. Graph-coloring trajectories instead independently permute vertices, map predictions back to the original vertex order, and canonicalize color names before pooling solutions. Both procedures also apply at $N = 1$.

Table 10: N-Queens Coverage@Depth (%) at $ND = 64$, pooling all N trajectories through depth D. Mean ± sample SD over three training seeds after averaging 32 inference repeats within each seed.

Depth D	Trajectories N	Last-Depth	Best-of-Depth
1	64	0.00 ±0.00	0.00 ±0.00
2	32	0.00 ±0.00	0.00 ±0.00
4	16	0.00 ±0.00	0.50 ±0.30
8	8	0.00 ±0.00	31.78 ±1.81
16	4	48.49 ±2.06	94.13 ±0.56
32	2	56.25 ±2.40	93.45 ±0.92
64	1	44.55 ±3.85	88.28 ±1.32

Table 11: Graph-coloring Coverage@Depth (%) at $ND = 64$, pooling all N trajectories through depth D. Mean ± sample SD over three training seeds after averaging 32 inference repeats within each seed.

Depth D	Trajectories N	Last-Depth	Best-of-Depth
1	64	2.25 ±1.25	33.85 ±10.34
2	32	2.26 ±1.35	62.86 ±2.63
4	16	20.61 ±1.82	77.28 ±0.67
8	8	36.39 ±0.65	82.80 ±0.20
16	4	37.50 ±0.51	85.86 ±0.49
32	2	28.54 ±0.52	82.67 ±1.44
64	1	18.12 ±0.41	72.14 ±3.47

C.3 Training Depth and Test-Time Extrapolation

The training-depth ablation uses BoD models trained at depths 4, 8, and 16, each with three seeds. The protocol holds the evaluation population and training-update budget fixed within each task, as specified in Appendix A. Depth-16 models are the same as in Figure 2. Evaluation uses unperturbed greedy trajectories. Tables 12 and 13 give coverage values for Figure 3(c, d).

Table 12: N-Queens BoD Coverage@Depth (%) across training and test depths. Mean ± sample SD across three training seeds.

Training depth	@4	@8	@16	@32	@64
4	6.48 ±3.88	8.11 ±3.96	8.21 ±4.01	8.21 ±4.01	8.21 ±4.01
8	6.61 ±2.73	60.21 ±11.72	75.15 ±11.99	75.66 ±11.90	75.70 ±11.95
16	0.10 ±0.09	17.11 ±1.25	85.92 ±0.93	90.97 ±0.98	91.03 ±1.05

Table 13: Graph-coloring BoD Coverage@Depth (%) across training and test depths. Mean ± sample SD across three training seeds.

Training depth	@4	@8	@16	@32	@64
4	21.09 ±0.50	27.64 ±0.83	31.53 ±0.54	33.32 ±0.64	34.13 ±0.69
8	17.02 ±0.55	34.69 ±0.92	44.86 ±0.90	50.38 ±1.10	53.11 ±1.57
16	14.18 ±0.53	30.76 ±0.70	54.95 ±0.80	66.79 ±1.89	72.16 ±3.62

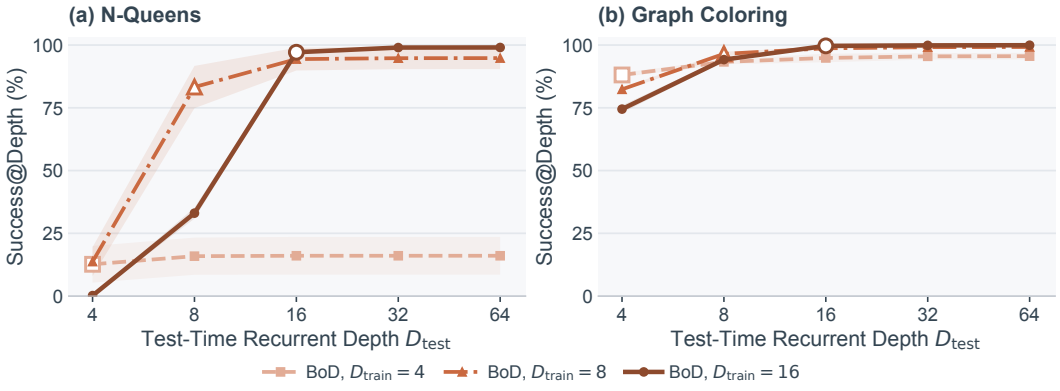

Figure 5: **Success across training and evaluation depths.** Success@Depth for BoD trained at depths 4, 8, and 16, using the same checkpoints and evaluation problems as Figure 3(c, d). Open markers indicate evaluation at the training depth. Curves show means and shaded bands indicate ±1 sample SD across three training seeds.

Figure 5 and Table 14 show the corresponding success results. Success improves beyond training for the depth-4 and depth-8 models on both tasks. For graph coloring, success saturates earlier than coverage, separating initial solution finding from continued discovery of alternatives.

Table 14: BoD Success@Depth (%) for Figure 5. Mean ± sample SD across three training seeds.

Training depth	@4	@8	@16	@32	@64
			N-Queens		
4	12.67 ±7.31	15.90 ±7.41	16.07 ±7.51	16.07 ±7.51	16.07 ±7.51
8	13.73 ±5.78	83.27 ±8.42	94.40 ±4.54	94.77 ±4.37	94.80 ±4.39
16	0.20 ±0.17	33.03 ±2.20	97.13 ±0.55	99.00 ±0.10	99.03 ±0.12
			Graph Coloring		
4	88.09 ±0.85	93.29 ±0.92	94.86 ±1.71	95.51 ±1.28	95.57 ±1.30
8	82.36 ±0.56	96.48 ±0.90	98.70 ±0.45	99.15 ±0.30	99.22 ±0.39
16	74.48 ±1.47	94.21 ±0.74	99.67 ±0.30	99.87 ±0.23	99.93 ±0.11

C.4 Sensitivity to the Soft-Depth Temperature

The temperature in Soft-Depth controls how strongly learning concentrates on the lowest-loss readouts. We vary $\tau \in \{0.1, 0.2, \ldots, 1.0\}$ on graph coloring and evaluate cumulative success and coverage through depth 64, keeping all other settings fixed (Appendix A.2). Coverage decreases as temperature increases, approaching Mean-Depth at larger temperatures (Figure 6). The development-selected $\tau = 0.1$ achieves 62.41% test coverage, compared with 72.16% for BoD, while both achieve success above 99%. Smaller temperatures are not evaluated, so this comparison does not establish the optimal Soft-Depth temperature beyond the tested range.

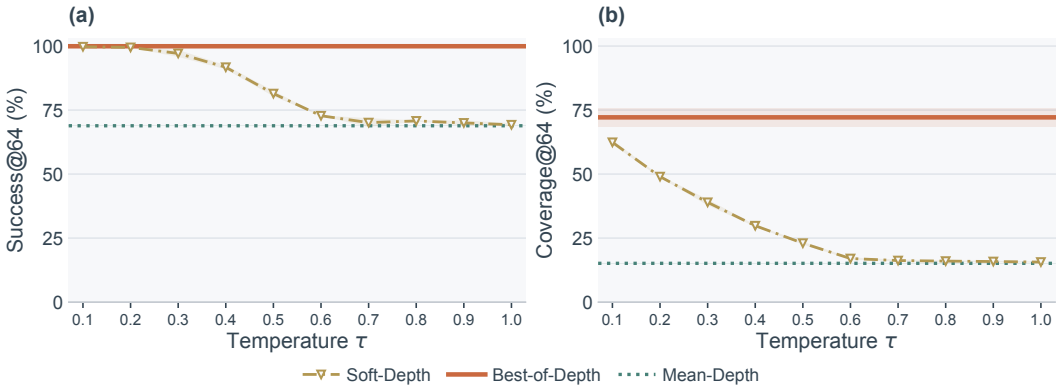

Figure 6: **Sensitivity to the Soft-Depth temperature on graph coloring.** Test success and coverage use all greedy readouts through depth 64. The curve varies $\tau = 0.1, 0.2, \ldots, 1.0$; horizontal lines show BoD and Mean-Depth under the same configuration. Lines show means and shaded bands indicate ±1 sample SD across three training seeds.

C.5 Withheld-Solution Discovery Protocol

The ablation in Section 4.5.3 uses 3,034 graphs from the graph-coloring training split with at least five canonical valid solutions per graph. The cohort contains 79,713 solutions in total, with 5–128 per graph (mean 26.27). This restriction leaves withheld solutions in every one- through four-target condition; it changes the evaluation cohort relative to the full graph-coloring experiment. All coloring families are identified up to a global permutation of color names, as in Appendix A.2.

For each graph and experimental seed, one random ordering of all valid solutions defines nested teacher pools of sizes one, two, three, and four. The all-solutions condition uses the complete pool. Pools remain fixed throughout each run; only the selected teacher is resampled at each presentation. Last-Depth and BoD share teacher pools, target draws, graph order, and initialization within each seed. The protocol uses the architecture and optimizer in Appendix A.2, training depth 16, 10,000

updates, and three paired seeds. Teacher-exposure logs verify that every selected teacher has been presented and no excluded solution has been used as a target.

Last-Depth and BoD each run one unperturbed trajectory through depth 64. Last-Depth + BoN reuses the Last-Depth checkpoint and pools predictions from independently vertex-permuted trajectories under $ND = 64$. The allocation is selected from the seven pairs in Section 4.5.1 using development-set coverage averaged across teacher conditions and seeds. This selects $N = 4$ trajectories of depth $D = 16$, fixed before evaluating withheld-solution discovery. Outputs are mapped back to the original vertex order before verification and canonicalization. Every method collects all intermediate readouts; Last-Depth + BoN therefore denotes sampling with all-depth readout, rather than terminal-only sampling. Randomized evaluation uses 32 repeats per training seed, averaged before computing the mean and sample SD across training seeds.

Seen and unseen coverage are macro-averaged over the same fixed graph cohort, using the teacher pool and its valid-solution complement as their respective denominators. Since increasing the teacher pool reduces the number of withheld solutions, supporting measurements include total coverage, the number of distinct unseen solutions recovered and the fraction of graphs with at least one such solution. The all-solutions condition has no withheld solutions, so its unseen coverage is undefined, while seen coverage equals total coverage. Results on disjoint test graphs, measured separately using ordinary coverage and success, assess generalization to new inputs rather than discovery beyond a known input's targets.

C.6 Additional Baseline Protocols

Table 15 uses the same datasets, teacher sampling, optimizer recipe, width 128, eight attention heads, and FFN width 384 as the main experiments. Each method uses three training seeds and a fixed final EMA checkpoint after 20,000 updates for N-Queens or 10,000 updates for graph coloring. These are independent implementations under a common training recipe. Parameter counts and training and inference FLOPs are not matched.

Table 15: Baseline comparison on N-Queens and graph coloring. Success and Coverage (%) summarize each method's candidate set, with mean ± sample SD over three training seeds. Ouro uses an instance-level adaptation of its training objective. Compute and candidate counts differ across architectures; Appendix C.6 gives the protocol.

	N-Queens		Graph Coloring	
Method	Success	Coverage	Success	Coverage
Transformer (8 layers)	9.8 ± 1.3	4.8 ± 0.6	39.6 ± 1.8	5.2 ± 0.5
Transformer (32 layers)	10.9 ± 1.1	5.4 ± 0.6	44.5 ± 1.1	6.3 ± 0.3
Autoregressive Transformer	57.4 ± 3.0	27.4 ± 1.4	67.9 ± 1.5	8.2 ± 0.4
Masked Diffusion	34.2 ± 4.1	16.3 ± 2.0	76.1 ± 1.5	10.3 ± 0.3
HRM	98.6 ± 0.5	48.0 ± 0.6	67.4 ± 0.9	13.7 ± 0.2
TRM	49.4 ± 23.6	23.8 ± 11.4	67.4 ± 1.5	14.3 ± 0.3
Ouro	77.9 ± 4.5	46.8 ± 3.7	73.3 ± 1.3	18.6 ± 0.1
Best-of-Depth	**99.0** ± 0.1	**91.0** ± 1.0	**99.9** ± 0.1	**72.2** ± 3.6

The unshared Transformers have 8 or 32 layers and produce one candidate per input. The autoregressive baseline uses a four-layer decoder and produces one complete candidate. Masked diffusion uses four layers with a SUBS parameterization and a completed 64-step denoising schedule. HRM uses four-layer low- and high-level modules with two cycles at each level; TRM uses two shared layers, four inner iterations, three cycles, and at most 16 outer training segments. HRM and TRM are evaluated with a fixed 64-step expansion rather than their native adaptive stopping rule. BoD and Ouro also collect all readouts through depth 64. Success and coverage are computed on the resulting candidate sets, using the definitions in Section 4.1. For the autoregressive and diffusion baselines, metrics are averaged across three inference seeds within each training seed, without pooling candidates across repeats. The table then reports the mean and sample SD across training seeds.

The Ouro baseline applies its Stage I expected-loss and entropy-regularization objective to whole instances, using the same looped backbone as BoD. A shared linear gate pools the recurrent states

over unknown cells for N-Queens and over vertices for graph coloring, then predicts a sigmoid exit hazard at each depth. The first 15 hazards define an exit distribution by survival products, with the remaining mass assigned to depth 16. The gate adds 129 parameters; its weights start at zero and its bias at $-\log 15$. Training minimizes the expected per-instance loss plus β times the KL divergence of the exit distribution from the uniform distribution over the 16 training depths. Evaluation collects all greedy readouts through depth 64; the gate neither stops the trajectory nor filters candidates. This comparison adapts the learning objective, rather than evaluating a pretrained Ouro language model or reproducing its second training stage.

We select one positive coefficient per task by mean development Coverage@Depth at depth 64 over three training seeds, using the grid $\{0.001, 0.003, 0.01, 0.03, 0.05, 0.1, 0.25, 1\}$. A separate $\beta = 0$ control is excluded from selection; exact ties favor the smaller coefficient. The selected values are $\beta = 0.03$ for N-Queens and $\beta = 0.001$ for graph coloring, using 200 multi-solution development puzzles and 512 development graphs, respectively.

C.7 Code Generation

Tables 16 and 17 report accuracy at each of the six evaluated depths and Success@32 over their union, on the full evaluation sets for Base and all five fine-tuning objectives.

Table 16: HumanEval accuracy (%) at each fixed depth, using one greedy program per problem. The final row reports Success@32 across the six evaluated depths.

Depth	Base	Last-Depth	Mean-Depth	Random-Depth	Soft-Depth	Best-of-Depth
4	0.0	0.0	6.7	4.9	6.1	0.6
8	12.8	13.4	20.1	20.1	20.7	14.6
12	23.2	25.0	25.6	25.6	26.2	26.8
16	24.4	27.4	27.4	26.8	26.2	29.3
24	25.0	27.4	26.8	28.0	26.8	27.4
32	25.6	26.8	27.4	28.7	26.8	29.9
Success@32	30.5	34.8	31.7	30.5	32.9	37.8

Table 17: MBPP accuracy (%) at each fixed depth, using one greedy program per problem. MBPP uses the public tests shown in the prompt. The final row reports Success@32 across the six evaluated depths.

Depth	Base	Last-Depth	Mean-Depth	Random-Depth	Soft-Depth	Best-of-Depth
4	1.2	1.8	10.4	8.8	9.4	0.6
8	17.0	21.2	26.0	25.8	25.2	21.0
12	26.8	28.2	28.4	28.2	29.2	29.4
16	29.0	30.2	28.0	29.6	29.2	30.6
24	29.6	32.0	29.2	29.0	30.4	29.4
32	29.8	31.4	29.0	29.6	30.8	33.2
Success@32	39.8	39.2	37.2	37.4	39.4	44.2

On HumanEval, BoD solves 62 problems across the six depths, compared with 57 for Last-Depth, 54 for Soft-Depth, 52 for Mean-Depth, and 50 for both Random-Depth and Base. Relative to depth 32 alone, additional depths solve 13, 13, 10, 7, 3, and 8 new problems, respectively. On MBPP, BoD solves 221 problems across depths, compared with 199 for Base, 196 for Last-Depth, 197 for Soft-Depth, 186 for Mean-Depth, and 187 for Random-Depth. Additional depths contribute 55 solutions beyond depth 32 alone for BoD, compared with 50, 39, 43, 41, and 39, respectively. The advantage is not uniform across individual depths: Mean-Depth, Random-Depth, and Soft-Depth perform better than BoD at depths 4 and 8 on both benchmarks, and Last-Depth performs better at MBPP depths 4, 8, and 24. These aggregate results establish candidate-set gains under this protocol; they do not measure variation across independent fine-tuning runs.

D THEORETICAL ANALYSIS

Throughout this appendix, fix a non-autoregressive input x and a deterministic recurrent trajectory. Targets are drawn from the finite valid solution set $\mathcal{Y}^\star(x)$, with cardinality $M \geq 1$, uniformly unless otherwise specified. For the cross-entropy results, all depths $1, \ldots, D$ are read out. The joint readout distribution on the complete output space Ω, with $p_{d,j}$ the categorical distribution at predicted position j, is

$$P_d(y \mid x) = \prod_{j=1}^{T(x)} p_{d,j}(y_j \mid x). \tag{15}$$

It is normalized over Ω, which includes invalid assignments. All logarithms are natural. For cross-entropy, define the expected whole-example loss

$$s_D(x) = T(x)\, \mathbb{E}_Y[\mathcal{L}_{\text{BoD}}(x, Y)] = \mathbb{E}_Y\left[-\log \max_{d \leq D} P_d(Y \mid x)\right]. \tag{16}$$

Here $T(x)$ is the number of predicted positions; the factor $T(x)$ reverses the position averaging in the training loss. We suppress x where the conditioning is unambiguous.

D.1 PROOFS OF PROPOSITIONS 1 AND 2

For Proposition 1, the minimum of the binary losses is zero exactly when at least one readout equals the target, which is equivalent to $\tau_y \leq D$. Hence the loss equals $\mathbf{1}[\tau_y > D]$ pointwise. Taking an expectation over any distribution of target–trajectory pairs gives Eq. 7.

For Proposition 2, fix the candidate set independently of the sampled target. For any distribution q supported on $\mathcal{Y}^\star(x)$,

$$\mathbb{E}_Y[\mathcal{L}^{0\text{–}1}_{\text{BoD}} \mid x] = \sum_{y \in \mathcal{Y}^\star(x)} q(y \mid x)\, \mathbf{1}[y \notin S_D(x)] \tag{17}$$

$$= 1 - \sum_{y \in S_D(x) \cap \mathcal{Y}^\star(x)} q(y \mid x). \tag{18}$$

When $q(y \mid x) = 1/M$, the second term is exactly the fraction of valid solutions in the candidate set, proving Eq. 8. For a random candidate set, the same identity holds conditional on that set, provided it is independent of the target given x; averaging then yields expected coverage.

For graph-coloring family coverage, both the candidates in $S_D(x)$ and the predictions in the exact-match loss are canonicalized in the same way. The target set contains one canonical representative per valid family. Using raw color-label equality on the left and family coverage on the right would not give this identity. The cross-entropy results below instead use the raw joint distribution in Eq. 15: generating a canonical target is sufficient to discover its family, so their coverage lower bounds also hold after canonicalization.

D.2 DEPTH-WISE SAMPLING AND COVERAGE

Lemma 1 (Distinct valid solutions from depth-wise sampling)**.** *Draw $\tilde{Y}_d \sim P_d(\cdot \mid x)$ at each depth and set $S_D(x) = \{\tilde{Y}_d : d \leq D\}$. Let K_D count the distinct valid solutions in this random candidate set. Then*

$$\mathbb{E}[K_D \mid x] \geq M e^{-s_D(x)}, \qquad \mathbb{E}[\text{Coverage@Depth}(D; x) \mid x] \geq e^{-s_D(x)}. \tag{19}$$

The expectations are over candidate sampling; independence between candidates is not required.

Proof. For each valid solution y, define $A_y = \{\tilde{Y}_d = y \text{ for some } d \leq D\}$. The union contains each individual event, hence

$$\Pr(A_y \mid x) \geq \max_{d \leq D} \Pr(\tilde{Y}_d = y \mid x) = \max_{d \leq D} P_d(y \mid x). \tag{20}$$

This uses only the marginal distribution of each candidate. Let

$$a_D(y) = \max_{d \le D} P_d(y \mid x), \qquad Z_D^\star = \sum_{y \in \mathcal{Y}^\star(x)} a_D(y). \tag{21}$$

Linearity of expectation gives $\mathbb{E}[K_D \mid x] \ge Z_D^\star$. Since targets are uniform, Jensen's inequality for the convex function $-\log$ yields

$$s_D = \frac{1}{M} \sum_{y \in \mathcal{Y}^\star(x)} -\log a_D(y) \ge -\log\left(\frac{Z_D^\star}{M}\right). \tag{22}$$

Thus $Z_D^\star \ge M e^{-s_D}$, proving the count bound. Dividing by M gives the coverage bound. If some valid target has $a_D(y) = 0$, then $s_D = \infty$ and the stated lower bound is zero. □

The result also holds when additional candidates are drawn at any depth, since adding candidates cannot remove a discovered solution. The sampling distributions must match those in the loss. For independent draws, the exact expected count for one sample per depth is

$$\mathbb{E}[K_D \mid x] = \sum_{y \in \mathcal{Y}^\star(x)} \left[1 - \prod_{d=1}^{D} (1 - P_d(y \mid x))\right]. \tag{23}$$

Sharpness. Suppose M is divisible by D. Partition the valid solutions into D disjoint subsets of size M/D, and let P_d be uniform on the dth subset. Then $s_D = \log(M/D)$, and one draw from each depth produces exactly $D = M e^{-s_D}$ distinct valid solutions. This establishes sharpness as a bound on normalized readout distributions. It does not require that a particular factorized decoder can represent every such partition.

D.3 A Shared-Recurrent Construction for Proposition 3

Task and valid solutions. There is one input, the labeled complete bipartite graph $K_{3,3}$. Its valid outputs are the six perfect matchings, represented by the 3×3 permutation matrices in $\mathcal{M}$. The output space is $\{0,1\}^{3\times 3}$; a selected edge has value one. A training target Y is drawn independently and uniformly from $\mathcal{M}$ at each update. Every edge occurs in two matchings, so $\mathbb{E}[Y] = J/3$, where J is the all-ones matrix. The model does not receive the target as input, and decoding does not enforce the matching constraints.

One shared recurrence. Let $r = (1\ 2\ 3)$ permute the left vertices and $c = (1\ 2)$ permute the right vertices. Define the fixed linear map $\mathcal{R}$ by $(\mathcal{R}A)_{r(i),c(j)} = A_{ij}$. This is a symmetry of the graph and preserves matchings. It satisfies $\mathcal{R}^6 = I$, and every matching has a full orbit:

$$\{\mathcal{R}^d M : d = 1, \dots, 6\} = \mathcal{M} \qquad (M \in \mathcal{M}). \tag{24}$$

Indeed, identify a matching with a permutation π from left to right vertices. If $\mathcal{R}^d M^\pi = M^\pi$, then $c^d = \pi r^d \pi^{-1}$. When $3 \nmid d$, the right side has order three while the left has order one or two. Thus $3 \mid d$, and the identity then also requires $2 \mid d$. The orbit therefore has length six.

The only trainable parameter is a matrix $\theta \in \mathbb{R}^{3\times 3}$. Let Π denote the permutation matrix representing $\mathcal{R}$ on vectorized matrices. A ten-dimensional hidden state with fixed initialization evolves according to

$$h_0 = \begin{pmatrix} 0_9 \\ 1 \end{pmatrix}, \qquad h_d = S(\theta) h_{d-1}, \qquad S(\theta) = \begin{pmatrix} \Pi & \Pi \operatorname{vec}(\theta) \\ 0_{1\times 9} & 0 \end{pmatrix}. \tag{25}$$

The first nine entries of h_d give a logit matrix H_d; the last entry is zero after the first recurrent step. Consequently, $H_d = \mathcal{R}^d \theta$. The same parameter-dependent matrix $S(\theta)$ is applied at every depth, but the trainable pattern is injected only at the first step. Subsequent steps apply the prescribed permutation. Thus, this construction studies how the objective uses an available recurrent orbit, rather than learning that orbit's rule.

Readout and common training protocol. All methods use $D = 6$ and the shared factorized Bernoulli readout

$$P_{\theta,d}(y) = \prod_{i,j} \sigma(H_{d,ij})^{y_{ij}} [1 - \sigma(H_{d,ij})]^{1-y_{ij}}, \qquad L_d(\theta; Y) = -\frac{1}{9} \log P_{\theta,d}(Y), \tag{26}$$

where σ is the sigmoid. Greedy decoding selects an edge exactly when its logit is positive. Each method starts at $\theta^{(0)} = bJ$, with $b = -\log 2$, and uses single-example SGD with $\eta_n = 9/(n+1)$. This is equivalent to step size $1/(n+1)$ on the summed loss; the factor nine accounts for the position averaging in Eq. 26. Random-Depth samples its depth independently of the target. BoD selects one minimizing depth uniformly when losses tie. Soft-Depth uses any fixed $\tau > 0$ in Eq. 6. All methods may share the same target sequence. For the one input in this construction,

$$\text{Coverage@Depth}(6; \theta) = \frac{\left|\{\mathbf{1}[\mathcal{R}^d \theta > 0] : d = 1, \ldots, 6\} \cap \mathcal{M}\right|}{6}. \tag{27}$$

The indicator acts entrywise. This makes explicit the parameter dependence used in Proposition 3.

Proof of Proposition 3. Write $\Phi(\theta) = \sum_{i,j} \log(1 + e^{\theta_{ij}})$ and use the Frobenius inner product. Because $\mathcal{R}$ permutes entries,

$$L_d(\theta; Y) = \frac{1}{9}\big[\Phi(\theta) - \langle \theta, \mathcal{R}^{-d} Y \rangle\big], \qquad \nabla_\theta L_d(\theta; Y) = \frac{1}{9}\big[\sigma(\theta) - \mathcal{R}^{-d} Y\big]. \tag{28}$$

Baseline minimizers and stationary objectives. Let R_A denote the expected loss for objective A, including any random depth. Equation 24 implies $\frac{1}{6}\sum_{d=1}^{6} \mathcal{R}^{-d} Y = J/3$ for every target. Hence

$$R_{\text{Last}}(\theta) = R_{\text{Mean}}(\theta) = R_{\text{Random}}(\theta) = \frac{1}{9}\big[\Phi(\theta) - \langle \theta, J/3 \rangle\big]. \tag{29}$$

This risk has positive diagonal Hessian entries $\sigma(\theta_{ij})[1 - \sigma(\theta_{ij})]/9$ and the unique minimizer bJ. For Mean-Depth the loss already equals this risk for each individual target, so its gradient is zero at initialization and it remains there.

At bJ, all depthwise losses tie. Soft-Depth therefore assigns weight $1/6$ to each gradient, independently of $\tau > 0$. For every target,

$$\nabla_\theta \mathcal{L}_{\text{Soft-Depth}}\big|_{bJ} = \frac{1}{9}\left[\frac{1}{3} J - \frac{1}{6}\sum_{d=1}^{6} \mathcal{R}^{-d} Y\right] = 0. \tag{30}$$

It too remains exactly at initialization. Every logit is then $b < 0$ at every depth, so Mean-Depth and Soft-Depth always have zero greedy coverage.

Almost-sure convergence for Last-Depth and Random-Depth. Their updates have the form $\theta^{(n+1)} = \theta^{(n)} - a_n[\sigma(\theta^{(n)}) - Z_n]$, where $a_n = 1/(n+1)$ and, conditional on the training history $\mathcal{F}_n$, Z_n is uniform over $\mathcal{M}$. Set $V_n = \|\theta^{(n)} - bJ\|_F^2$ and $A(\theta) = \langle \theta - bJ, \sigma(\theta) - J/3 \rangle$. Strict monotonicity of the sigmoid gives $A(\theta) > 0$ except at bJ; bounded coordinates of the stochastic gradient give

$$\mathbb{E}[V_{n+1} \mid \mathcal{F}_n] \le V_n - 2a_n A(\theta^{(n)}) + 9a_n^2. \tag{31}$$

Thus $V_n + 9\sum_{k=n}^{\infty} a_k^2$ is a nonnegative supermartingale. It converges almost surely, and summing its expected drift yields $\sum_n a_n A(\theta^{(n)}) < \infty$ almost surely. Consequently V_n has a finite limit. If this limit were positive, the iterates would eventually lie in a compact annulus about bJ on which A has a positive minimum. This contradicts $\sum_n a_n = \infty$. Therefore both methods converge almost surely to bJ. Since $b < 0$, all logits eventually remain negative, and greedy coverage is eventually zero.

BoD breaks symmetry and retains one latent matching. The orbit identity gives, for every target,

$$\mathcal{L}_{\text{BoD}}(\theta; Y) = \frac{1}{9}\left[\Phi(\theta) - \max_{M \in \mathcal{M}} \langle \theta, M \rangle\right]. \tag{32}$$

At initialization, let the selected tied depth for Y_0 be $d_0^\star$ and set $M_\star = \mathcal{R}^{-d_0^\star} Y_0$. The first update gives

$$\theta^{(1)} = bJ + M_\star - J/3. \tag{33}$$

Its entries equal $\alpha_1 = b + 2/3$ on $M_\star$ and $\beta_1 = b - 1/3$ elsewhere. For any other matching sharing $t < 3$ edges with $M_\star$, the score gap is $(3-t)(\alpha_1 - \beta_1) > 0$. Thus $M_\star$ is now the unique maximum-score matching. Every later target selects the phase satisfying $\mathcal{R}^{d^\star} M_\star = Y$, so

$$\nabla_\theta \mathcal{L}_{\mathrm{BoD}}(\theta; Y) = \frac{1}{9}[\sigma(\theta) - M_\star]. \tag{34}$$

All targets reinforce the same latent matching. The two logit values satisfy

$$\alpha_{n+1} = \alpha_n + a_n[1 - \sigma(\alpha_n)], \qquad \beta_{n+1} = \beta_n - a_n \sigma(\beta_n), \qquad n \geq 1. \tag{35}$$

Their gap grows, preserving the unique maximizing matching. Both are negative after the first update, but $\alpha_2 > -\log 2 + 2/3 + 1/4 > 0$ while $\beta_2 < 0$. These signs persist. For every $n \geq 2$, greedy decoding therefore yields $\hat{y}_d = \mathcal{R}^d M_\star$, whose six distinct values are all valid matchings. BoD has full coverage from the second update onward. This finite-update count uses the stated loss normalization and step sizes. Together with the baseline results, it proves the proposition. □

The Soft-Depth argument relies on exact symmetry at initialization. It does not imply failure under perturbed initializations. Likewise, the prescribed recurrent orbit makes this an existence result for a shared recurrent model, not a general guarantee for learned Transformer dynamics or a comparison of compute efficiency.

Numerical illustration. We run the specified SGD updates for 4,096 iterations in float64 on 64 independent teacher streams (seeds 0–63), shared across objectives within each run. Independent random streams select Random-Depth readouts and the initial BoD tie. Soft-Depth is evaluated at $\tau \in \{0.1, 0.25, 1\}$. Every reported checkpoint is decoded at all six depths and verified against the complete set of matchings. The Soft-Depth updates are computed from their actual weighted gradients; no special rule holds the parameters at initialization. Table 18 illustrates the proved distinction: BoD reaches full coverage after two updates in every run, whereas all four comparison objectives have zero coverage at the final checkpoint. Last-Depth and Random-Depth can temporarily produce valid solutions, consistent with the theorem's eventual, rather than all-iteration, statement. The experiment is a numerical illustration of this construction, separate from the learned looped Transformer experiments. The implementation and per-run results accompany the paper.

Table 18: Numerical illustration of Proposition 3. Greedy Coverage@Depth at $D = 6$ (%), averaged over 64 independent SGD runs on the same matching problem. Columns count training updates. All objectives use the same teacher stream within each run and the same initialization and step sizes.

Training objective	0	2	16	256	4,096
Last-Depth	0.0	15.6	1.6	0.0	0.0
Mean-Depth	0.0	0.0	0.0	0.0	0.0
Random-Depth	0.0	18.8	1.6	0.0	0.0
Soft-Depth ($\tau = 0.1$)	0.0	0.0	0.0	0.0	0.0
Soft-Depth ($\tau = 0.25$)	0.0	0.0	0.0	0.0	0.0
Soft-Depth ($\tau = 1.0$)	0.0	0.0	0.0	0.0	0.0
Best-of-Depth	0.0	100.0	100.0	100.0	100.0

D.4 A Certificate for Greedy Solution Coverage

Let S_D contain the distinct positionwise-greedy predictions from the same readouts. Then

$$\frac{|S_D \cap \mathcal{Y}^\star(x)|}{M} \geq \left[1 - \frac{s_D(x)}{\log 2}\right]_+. \tag{36}$$

To prove this, consider a valid target $y \notin S_D$. At every depth, at least one position is decoded to a token other than y_j. At that position, $p_{d,j}(y_j \mid x) \leq 1/2$, so $-\log P_d(y \mid x) \geq \log 2$. Consequently,

$$\mathbf{1}[y \notin S_D] \leq \frac{\min_{d \leq D} -\log P_d(y \mid x)}{\log 2}. \tag{37}$$

Averaging over the uniform target distribution proves Eq. 36. The result is nontrivial when the whole-example risk is below $\log 2$; for the position-averaged loss, the corresponding threshold is $\log 2 / T(x)$. In graph coloring, matching a canonical target exactly is sufficient to discover its canonical family, so subsequent canonicalization preserves this lower bound.