LONG-HORIZON LANGUAGE MODEL REINFORCEMENT LEARNING VIA PROGRESSIVE POINT MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Current paradigms for training language models via reinforcement learning rely heavily on sparse outcome rewards. However, as we pursue tasks that require longer and more complicated trajectories, such strategies result in slow learning. Prior work has attempted to address this problem by rewarding partial progress; however, naive formulations are often biased and converge to suboptimal policies. We show that a simple and *unbiased* dense reward formulation, which we term **progressive point matching**, scales *exponentially* more efficiently to long-horizon tasks by rewarding partial progress on a segment level, both theoretically and empirically via synthetic environments. We then show how progressive point matching can be practically instantiated using a single reference trajectory per task. On extremely hard math reasoning problems, sparse outcome rewards cannot make any progress, whereas segment-level rewards enable improvements at larger test-time token budgets when measured by success rate or pass@k.

1 INTRODUCTION

Reinforcement learning (RL) has proven a powerful tool for improving the reasoning and agentic capabilities of large language models (LLMs). The typical paradigm is to utilize large-scale policy gradients, with a sparse outcome reward generated at the end of a full trajectory. While straightforward to implement and verify, sparse rewards often result in weak signal per trajectory (Schulman & Lab, 2025) and brings a range of potential difficulties such as slow training and poor exploration (Ladosz et al., 2022; Andrychowicz et al., 2018).

Such issues are especially prominent in long-horizon tasks, which require solving many subproblems before ultimately reaching the goal. As we show in Figure 1, sparse outcome reward methods require exponentially more trajectories to reach a desired pass rate as the task horizon increases, due to exponentially degrading signal-to-noise ratio of the policy gradient (Theorem 4.1). Indeed, the standard approach to tackle difficult problems wastes compute by generating many more trajectories than needed, then discarding a large chunk of those trajectories altogether (Yu et al., 2025; Cheng* et al., 2026). In the worst case, the policy may be unable to sample a single success, resulting in no reward signal nor RL improvement.

How can we learn efficiently as we scale to longer and more difficult tasks? One natural answer is to utilize sources of reward signal other than the typical sparse outcome reward, which does not carry discriminative power beyond the final answer – if a trajectory makes progress on dozens of subproblems but fails at the final stretch, it nevertheless is assigned a reward of zero. By properly assigning partial credit to each trajectory, we may alleviate this trend.

Our key insight is that reasoning problems can be regarded as discovering paths through a Markovian state space. One may decompose a long-horizon task into *reasoning points*, which are sufficient statistics for subsequent reasoning. Thus, different prefixes of the trajectory that reach the same reasoning point are interchangeable with respect to future reasoning. For example, two trajectory prefixes that provide different proofs to the same lemma reach the same reasoning point, as subsequent reasoning can simply condition on the lemma without its proof. The *reasoning state* is the *set* of reached reasoning points, and the task becomes reaching a state that contains the goal. Unlike process rewards (Lightman et al., 2023), which may not correlate with true task progress (Setlur et al., 2024; Zhang et al., 2025), our method rewards reaching reasoning states that are closer to the goal state.

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.

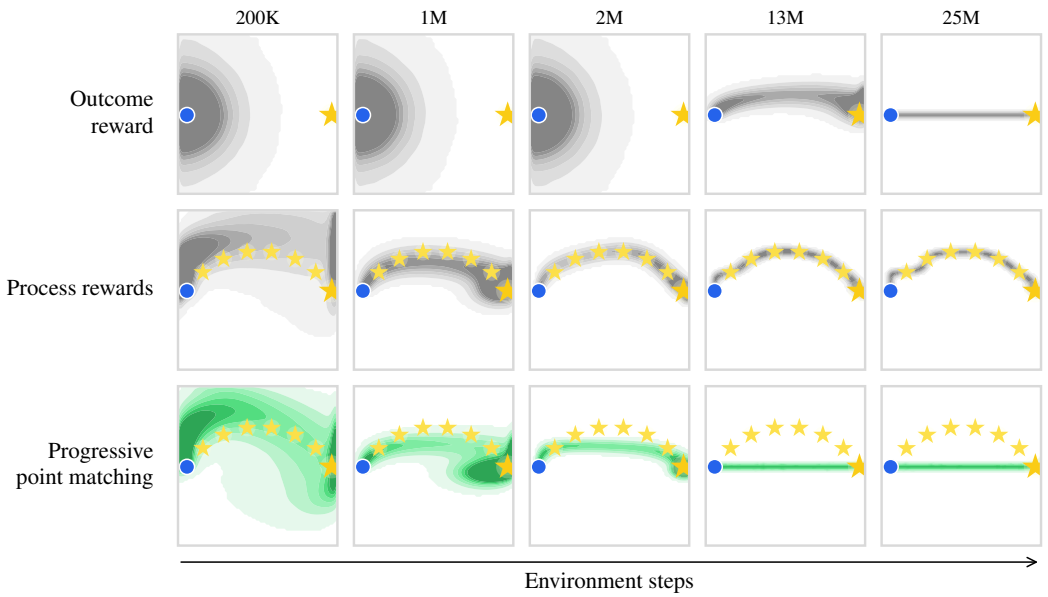


Figure 1: As an analogy, consider a long-horizon problem in which an agent is rewarded for the correctness of its final answer. In this figure, we apply a discount factor to demonstrate that the goal can be reached suboptimally. **Top:** Sparse outcome rewards are unbiased, but successful trajectories become exponentially rare as the task horizon grows – in our simulated task, over 12M samples are required to observe a single success. **Middle:** One may combat this by rewarding the agent for reaching each reasoning point, but this biases the agent toward suboptimality when the hints themselves are suboptimal. **Bottom:** Progressive point matching can efficiently learn the optimal policy under outcome rewards.

By designing the reasoning points, progressive point matching enables interpolation between imitation learning and reinforcement learning. At one extreme, treating every token-level prefix of a reference trajectory as a reasoning point approximates token-level imitation. At another, treating the final answer as the sole reasoning point recovers RL with sparse outcome rewards. Between these extremes, reasoning points can be selected to compress disjoint segments of a reference trajectory. Rewarding reaching reasoning points via on-policy RL is less "strict" than directly imitating reference trajectories, which enables the policy to reach points through potentially more efficient reasoning paths (Figure 1) while avoiding distribution-shift pitfalls of standard imitation learning (Ross et al., 2011; Agarwal et al., 2024). Theoretically, progressive point matching leads to exponentially higher signal-to-noise than sparse outcome rewards (Theorem 4.1) while enabling convergence to the optimal policy under outcome rewards (Proposition 3.1).

We instantiate our method as a modular modification to an off-the-shelf language model RL algorithm, e.g., PPO (Schulman et al., 2017) or GRPO (Shao et al., 2024). Reasoning points are constructed via privileged information, e.g. by using an LLM to extract them from a human or oracle solution. Importantly, we construct segment-level rewards, which allows to assign different advantages to tokens in a trajectory based on their relative contribution.

Our experimental setup isolates the effect of long-horizon scaling by focusing on three synthetic environments: a multi-turn version of Countdown (Gandhi et al., 2024), a multi-turn Matrix Manipulation task (Stojanovski et al., 2025), and synthetic math problems from GSM-Infinite (Zhou et al., 2025b). We show that as reasoning problems become harder, as characterized by requiring reaching more intermediate points to ultimately reach the goal, policies trained with sparse outcome rewards (i.e., standard GRPO) require exponentially more trajectories to converge. In contrast, policies trained via progressive point matching converge faster, with the gap in relative performance increasing with task horizon. We additionally show that progressive point matching allows us to train policies on *near-zero success rate* math problems, where methods using sparse outcome rewards fail to learn completely. When evaluated at larger test-time token budgets, policies trained with progressive point matching extrapolate their ability to achieve a higher success rate and pass@k rate according to outcome rewards.

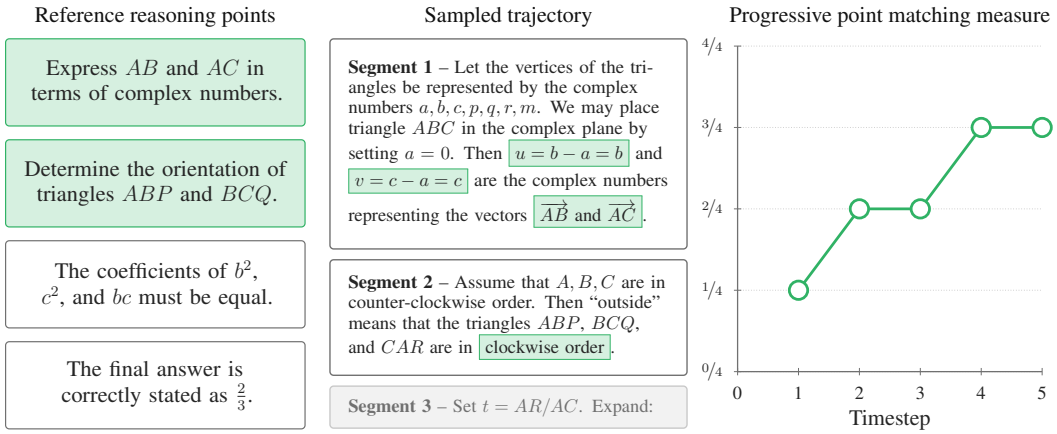


Figure 2: **Progressive point matching rewards enable segment-level credit assignment.** Shown above is an example challenging problem from Omni-MATH. **Left:** Rewards are constructed by measuring the number of reference reasoning points that a trajectory reaches. **Right:** Progressive point matching is a monotonic function over tokens. Successful trajectories will accumulate reward from 0 to 1 throughout the reasoning process, whereas unsuccessful trajectories may plateau when they deviate from reference trajectory.

2 Preliminaries

Markov decision processes. A Markov decision process (MDP) is defined by a state space, an action space $\mathcal{A}$, a transition function $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, a reward function $r : \mathcal{S} \to \mathbb{R}$, and an initial state distribution $p(s_0) \in \Delta(\mathcal{S})$. For our setting, we are concerned only with a subset of MDPs suitable for language reasoning – the reward function is nonzero *only* at the outcome state, transition functions are deterministic, and the state is fully observable. Our policy is defined as a probability distribution over actions $\pi(a \mid s) : \mathcal{S} \to \Delta(A)$, which defines the distribution of state-action trajectories $p(\tau \mid \pi) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t)$, where $\tau = (s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T)$. The standard RL objective is to learn a parameterized policy π_θ that maximizes the expected sum of rewards: $J(\pi_\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \sum_{t=0}^{T} \gamma^t r(s_t)$.

Language reasoning models as policies. Throughout this work, we consider the fine-tuning of large language models that reason via autoregressive generation. We define a sequence of tokens as $x_{0..T}$ where a prefix of these tokens comes from an external prompt, and the following are generated autoregressively. We never take a loss on the prompt tokens. We assume access to a base model, defined as a neural network that parameterizes a probability distribution $\pi_\theta(x_t \mid x_{<t})$. To measure the performance of such models, we consider a verifiable setting in which a set of training problems are associated with a verifiable success criterion defined as an outcome reward $r(x)$.

3 Rewarding Partial Progress via Progressive Point Matching

In this section, we describe a simple framework for assigning partial credit in RL, then how to instantiate language reasoning tasks within this framework. We aim to satisfy the following desiderata:

(1) Segments within trajectories are assigned rewards proportional to their partial progress toward the solution, even for problems where no trajectories reach the goal.

(2) Optimal policies under per-segment rewards are optimal under outcome rewards, and vice versa.

We start from the observation that, in typical long chain-of-thought reasoning trajectories, intermediate results compress earlier portions of the reasoning process. For example, in a math problem, a trajectory may prove a lemma that can then be used in subsequent steps of a longer proof without revisiting its derivation. We call such intermediate results *reasoning points*. Prior work (Qu et al., 2026) has used a similar mental model to develop guidance-conditioned RL training approaches. Accordingly, we operate under the assumption that reasoning trajectories admit compact state representations: the *set* of reasoning points reached by the current trajectory prefix. We can then view a reasoning trajectory as a path through the state space of a goal-reaching MDP, where the goal state contains the goal point (i.e., the final answer), and each action adds one or more new reasoning points to the current set.

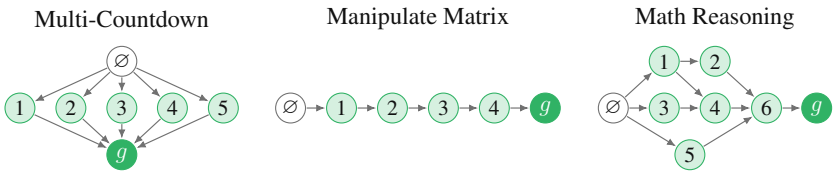


Figure 3: Reference trajectories from various MDPs consist of *reasoning points*, which can be arranged in *reasoning graphs*. By construction, every point is a prerequisite of the goal.

One implication of this formulation is that progress can never be harmed as a reasoning trace proceeds – each new trajectory segment may add new reasoning point(s) to this set, or leave the set unchanged. This is intentional and often desirable, e.g. deriving intermediate lemmas may be helpful to proving a particular theorem, but in general do not prevent further progress – one can always backtrack to a previous lemma. Thus, each reasoning trajectory is *non-decreasing in partial progress* (Figure 2). Successful trajectories therefore follow a desirable pattern: starting at an empty set, they make non-decreasing progress, and end by reaching the goal point.

Progressive point matching rewards. To construct a measure of partial progress, we rely on the reasoning points spanned by a reference trajectory. Over this set, we define the progressive point matching measure $\phi(s, s^{\text{ref}})$ (or simply $\phi(s)$) as the number of reference reasoning points $s^{\text{ref}} = \{s_1^{\text{ref}}, \ldots, s_{n-1}^{\text{ref}}, s_n^{\text{ref}} = g\}$ contained at a given state s:

$$\phi(s, s^{\text{ref}}) := \sum_{i=1}^{n} \mathbf{1}(s_i^{\text{ref}} \in s). \tag{1}$$

To derive per-segment rewards satisfying desiderata **(1)**, we can simply calculate the increase in the measure caused by that segment:[1]

$$\hat{r}(s_t, a_t) = \phi(s_{t+1}) - \phi(s_t). \tag{3}$$

Optimality equivalence. Training on progressive point matching rewards $\hat{r}$ directly can lead to suboptimal policies under outcome rewards. For example, a trajectory that reaches many intermediate reasoning points but fails to reach the goal may receive more reward than a trajectory that successfully reaches the goal via another path in the MDP.

Our solution to this problem is to give full credit for intermediate reasoning points that have been made obsolete by reaching further reasoning points. Specifically, we construct a latent *reasoning graph* of points (Figure 3), within which many paths (including the reference trajectory) can lead to the answer. The graph has vertices that are reasoning points, and the edge (u, v) is present if point u is a prerequisite of v. This notion of "shortcutting" is derived from the structure of the reasoning graph: a point is considered reached if all points that depend on this point have already been reached. In particular, reaching the goal makes *all* reasoning points reached. This framework allows progressive point matching to satisfy a key property:

Proposition 3.1. *Assume there exists a policy that reaches the goal g almost surely. Then the optimal policies under the progressive point matching and the sparse outcome rewards coincide.*

Proof. Since $\hat{r}$ telescopes and $\phi(s_0) = 0$,

$$\sum_{t=0}^{T-1} \hat{r}(s_t, a_t) = \phi(s_T). \tag{4}$$

Because $0 \leq \phi(s_T) \leq n$ and we are using shortcutting, the maximum expected return n is achieved iff the goal is reached almost surely. The sparse outcome reward has exactly the same set of optimal policies. □

[1]This framework matches potential shaping (Ng et al., 1999) at non-terminal states. Let $r(s) := \mathbf{1}\{g \in s\}$ denote the sparse outcome reward, and define $\Phi(s) := \phi(s) - r(s)$. Then, for non-terminal transitions,

$$\hat{r}(s_t, a_t) = r(s_{t+1}) + \Phi(s_{t+1}) - \Phi(s_t), \tag{2}$$

where $\gamma = 1$. Unlike potential shaping, however, progressive point matching retains the terminal potential for unsuccessful trajectories, providing partial credit for progress.

Algorithm 1 Progressive Point Matching

Require: Policy parameters θ, reference trajectory x^{ref}
  while not converged **do**
    Sample on-policy trajectory $x = x_{0..T} \sim \pi_\theta$
    Split into segments $x = (c_1, c_2, \ldots, c_K)$ (by turn boundaries or equal-length chunks)
    for each segment $k = 1, \ldots, K$ **do**
      Compute PPM measure over trajectory prefix: $m_k = \phi((c_1, \ldots, c_k),\, x^{\text{ref}})$
    end for
    Compute segment-level rewards: $r_k = m_k - m_{k-1}$, with $m_0 = 0$
    Compute token-level advantages: $A_t = f(r_1, \ldots, r_K)$ (e.g. GRPO group normalization)
    Update parameters: $\theta \leftarrow \theta + \alpha\, \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(x_t|x_{<t}) \cdot A_t\right]$
  end while

4 Empirical Study on Language Model Training

We now conduct a systematic study of language model RL, including our proposed progressive point matching method, specifically focusing on *scalability with task horizon*. To isolate this axis, we focus on reasoning tasks that are constructed out of a set of sub-problems that must first be solved before the final answer can be derived.

We instantiate progressive point matching via plug-and-play modifications to an off-the-shelf policy gradient algorithm. We consider the family of policy gradients that generally follow the REINFORCE objective with token-level advantages:

$$\Delta\theta = \mathbb{E}_{x\sim\pi_\theta}[\nabla \log \pi_\theta(x_t|x_{<t}) \cdot A_t], \tag{5}$$

where x_t is a sequence of tokens, and A_t is the token-level advantage. The common baseline of GRPO is a specific case of the above formulation in which a single advantage is assigned to all tokens in a given trajectory. We base our hyperparameter choices off GRPO unless otherwise described.

In the experiments below, we construct rewards by splitting each sampled trajectory into segments. These segments are constructed via natural indicators (e.g. turns in multi-turn environments), or simply via equal-length chunks. We then calculate the progressive point matching measure over every segment-wise *trajectory prefix*. Segment-level rewards are then calculated as the improvement between two trajectory prefixes, as described in Equation (3).

4.1 How does progressive point matching compare to baselines as tasks increase in horizon?

As one of our main motivations is to train on long-horizon tasks, we now experimentally and analytically analyze the effect of progressive point matching on synthetic environments as we increase horizon. Each represents a different reasoning graph structure:

- **Multi-Countdown.** The standard Countdown (Gandhi et al., 2024) task requires constructing an arithmetic from four given numbers that evaluates to a given target value. In Multi-Countdown, an agent is instead tasked with solving n such problems *independently*, with an outcome reward awarded if all problems are solved correctly.
- **Matrix Manipulation.** In this environment, adapted from Reasoning Gym (Stojanovski et al., 2025), the policy is provided with an initial matrix and a sequence of operations such as rotation, reflection, and element-wise operations, with one operation per turn. Each operation must be correctly performed in sequential order to progress to the next operation. This represents an opposite extreme, where each sub-problem is completely dependent on all preceding sub-problems.

Across these tasks, we consider three main classes of reward signal:

- **Sparse outcome** rewards broadcasted over the trajectory that indicate whether *all* sub-problems have been solved correctly.
- **Progressive point matching trajectory-level** rewards broadcasted over the trajectory that indicate the *fraction* of sub-problems that have been solved correctly.
- **Progressive point matching turn-level** rewards calculated via Equation (3).

Theoretical analysis. As the settings above can be seen as simple reasoning graphs, we can analyze the expected performance of various methods from the lens of signal-to-noise ratio (SNR) of the policy gradient estimator.

Theorem 4.1 (Informal)**.** *Let G be a reasoning graph on points $V = \{\varnothing, 1, \ldots, n, g\}$, where $\varnothing$ is an initial null point and g is the goal. For each v, conditioned on reaching all prerequisites of v, suppose that v is reached independently with probability p. The policy gradient estimators have signal-to-noise ratios (SNRs):*

- *Sparse outcome reward:* $\Theta(p^{n/2})$.
- *Progressive point matching trajectory-level:* $\Theta(n^{-1/2})$*, under regularity conditions.*
- *Progressive point matching turn-level: letting C denote the number of reached reasoning points, the SNR scales as* $\Theta\big(\mathbb{E}[C]/\sqrt{\mathrm{Var}(C)}\big)$*. Under regularity conditions, this is bounded below by* $\Omega(1/\log n)$.

The full assumptions, proof, and more general version are given in Appendix B. We can interpret Theorem 4.1 as follows: sparse outcome rewards become exponentially sparse with horizon, leading to exponentially small SNR in the policy gradient estimator. On the other hand, turn-level progressive point matching can yield *improving SNR with task horizon*.

As we show in Appendix B, the best case SNR of $\Theta(\sqrt{N})$ for progressive point matching occurs when each reasoning point is reached independently (Multi-Countdown MDP), resulting in independent gradients across turns. The sequential Matrix Manipulation MDP, despite having high correlations between reachability events, has a turn-level SNR of $\Theta(1)$, significantly outperforming trajectory-level progressive point matching.

Empirical analysis. We train Qwen3 models (Yang et al., 2025) on the two tasks above across all three reward designs, using GRPO as the base method. Hyperparameters are kept the same across trials. We use Qwen3-1.7B for Multi-Countdown, and Qwen3-4B-Instruct for Matrix Manipulation. In both cases, the environment consists of n turns with subproblems, and the model is given a budget of 512 tokens per turn to think and provide a final verifiable answer.

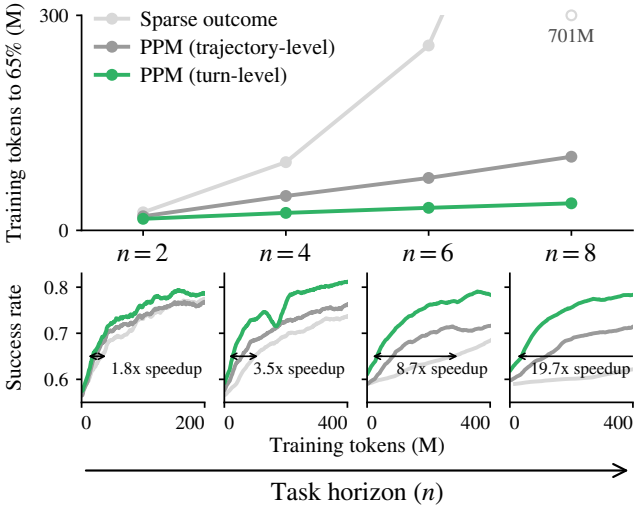


Figure 4: As the Multi-Countdown task horizon increases, progressive point matching results in larger performance improvements over trajectory-level partial credit and sparse outcome rewards. Theoretically and empirically, the training speed improves *exponentially* over sparse outcome rewards.

To instantiate progressive point matching, we utilize the symbolic structure of the tasks to identify the reference reasoning points. For Multi-Countdown, we check whether the solution to each subtask is correctly derived. For Matrix Manipulation, we check whether each matrix operation has been correctly performed. We intentionally chose easy-to-verify reasoning points to disentangle verification from the impact of progressive point matching on policy improvement – in the following section, we examine how to verify intermediate reasoning on less structured settings.

As shown in Figure 4 (Multi-Countdown) and Figure 8 (Matrix Manipulation), our main finding is that, across the board, progressive point matching consistently outperforms sparse outcome rewards. More importantly, **the margin between sparse outcome rewards and progressive point matching grows exponentially with the task horizon**. This matches our theoretical results and highlights the importance of dense partial credit assignment.

4.2 CAN PROGRESSIVE POINT MATCHING BE ADAPTED TO GENERAL REASONING TASKS?

The environments in Section 4.1 benefit from (i) natural turn-based structures that enable decomposition into subtasks and (ii) ground truth rewards per subtask. However, real-world long-horizon tasks typically do not admit this structure. Instead, many tasks implicitly demand extended reasoning (Research et al., 2026), with only the final answer directly verifiable. We investigate whether progressive point matching can yield stable improvements without environment-provided per-segment rewards.

We use GSM-Infinite (Zhou et al., 2025b) as an environment for long-horizon reasoning. In this environment, math problems are generated synthetically from graphs, in which nodes represent quantities and edges encode relationships between them. The task horizon is parameterized by the length of the shortest path between the given information and the desired quantity. GSM-Infinite also represents an intermediate reasoning graph between the two extremes of Multi-Countdown and Matrix Manipulation, and serves as a controllable proxy for more general reasoning tasks.

We compare against a representative set of on-policy RL baselines, focusing on methods that use reference trajectories:

- **GRPO** (Shao et al., 2024) with sparse outcome rewards.
- **Verifree** (Zhou et al., 2025a), a representative likelihood-based method that rewards reasoning based on the per-token likelihood of the solution conditioned on that reasoning.
- **On-policy self distillation (OPSD)** (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026), a representative context distillation method which places the solution in the context of the model to form a teacher distribution, whose logits are then distilled into the current policy.
- A **process reward** baseline, where we use an off-the-shelf Gemini model to reward correctness via logical consistency in segments (Miao et al., 2023; Khalifa et al., 2025).

We use GRPO from Qwen3-1.7B with similar configurations across the board, as described in Table 3.

Table 1: GSM-Infinite success rates at 400M training tokens.

Method	$n = 8$	$n = 16$	$n = 24$
Base model	0.134	0.056	0.056
Sparse outcome reward	0.684	0.084	0.091
Verifree	**0.828**	0.126	0.098
OPSD	0.283	0.102	0.078
Process reward	0.329	0.075	0.080
PPM (trajectory-level)	0.758	0.334	0.089
PPM (segment-level)	0.652	**0.415**	**0.191**

We then implement progressive point matching, considering both the per-trajectory and per-segment variants. We utilize a text rendering of the ground-truth solution graph as the set of intermediate reasoning points. An off-the-shelf Gemini judge, which we evaluate in Appendix C.3, is then used to measure how many of these points a given trajectory reaches. In the segment-level variant, we split each sampled trajectory into four uniformly sliced chunks, and pass each chunk independently (including all previous chunks) into the judge model.

We measure performance on GSM-Infinite at various task horizons n. As shown in Table 1, while many baselines are comparable in the regime of easy problems ($n = 8$), **progressive point matching enables larger performance gains with task horizon**. At the hardest difficulty level ($n = 24$), there is a clear benefit from assigning rewards at a segment level over trajectory level.

4.3 HOW SHOULD ONE INSTANTIATE THE PROGRESSIVE POINT MATCHING JUDGE ϕ?

In synthetic environments such as Multi-Countdown or Matrix Manipulation, one can instantiate a ground-truth ϕ by matching substrings directly. However, in open-ended reasoning tasks such as GSM-Infinite, there are no ground-truth labels indicating whether trajectory segments match reasoning points without annotations from a human or a stronger model. We provide concrete guidance on how to instantiate ϕ as the progressive point matching measure *without annotations*.

We make the following observation: conditioned on reaching s_t, the Monte Carlo success rate $\bar{r}(s_t)$ of subsequent reasoning should grow *linearly* with the progressive point matching measure $\phi(s_t)$. By definition, conditioning on the reasoning points in s_t is equivalent to conditioning on any trajectory prefix that reaches them. We can therefore estimate $\bar{r}(s_t)$ by placing *prefixes of the reference trajectory* in the prompt and sampling continuations from the policy.

Table 2: Qwen3-4B trained on Polaris dataset at 8K token budget and evaluated at 16K.

	Train		Test		AIME25		HMMT25	
Method	p@1	p@8	p@1	p@8	p@1	p@8	p@1	p@8
Base model	0.156	0.292	0.416	0.745	0.510	0.740	0.346	0.569
SFT	0.210	**0.609**	0.264	0.651	0.098	0.246	0.060	0.175
GRPO	0.203	0.401	0.441	0.806	**0.560**	0.727	0.354	0.556
Verifree	0.238	0.412	**0.461**	0.799	**0.569**	**0.757**	0.358	0.578
Process rewards	0.195	0.396	0.451	**0.832**	0.542	**0.756**	0.367	0.537
OPSD	0.215	0.433	0.436	0.815	0.529	0.729	0.358	**0.597**
Progressive point matching	**0.258**	0.452	**0.459**	0.807	0.535	**0.748**	**0.379**	**0.598**

Consider the following two settings: (i) on hard problems where the policy cannot successfully solve the task, both the Monte Carlo return and the mean terminal progressive point matching measure should be low; and (ii) after providing the full reference trajectory in context, asking the policy to reproduce the solution should almost always succeed, and both quantities should be near one. In our experiments, we interpolate between these settings by providing reference prefixes of increasing length and comparing the Monte Carlo return at each prefix with the corresponding mean terminal progressive point matching measure (Figure 10). Our judge ϕ, which we instantiate as an off-the-shelf Gemini model with a specific prompt that optimizes the above relation, yields a Kendall rank correlation of 0.856, compared with 0.768 for a standard rubric prompt (see Appendix E).

We can additionally couple the above criterion with a small set of hand-designed validation trajectories, where manual inspection reveals where the trajectory deviates from empirical returns. A strong judge prompt should correctly show this deviation in its progressive point matching measure.

4.4 Does progressive point matching extrapolate well with test-time compute?

We now investigate whether progressive point matching can be applied to real math reasoning, and we specifically focus on setting in which we desire test-time extrapolation of the learned policies. We use the Polaris dataset (An et al., 2025) of math problems with verifiable numerical answers and oracle reference solutions generated by gemini-3-flash (Google DeepMind, 2025b).

In addition to the baselines of the previous section, we compare to supervised fine-tuning (SFT), which maximizes the likelihood of reference trajectories.

As shown in Table 2, progressive point matching reliably extracts signal from hard-to-solve math problems. In particular, policies trained via progressive point matching extrapolate cleanly when they are evaluated at 16K tokens, even though the policies are trained at much lower budgets. As an on-policy RL method, progressive point matching does not suffer from catastrophic collapse, and maintains consistent entropy to achieve considerable improvement at pass@8.

4.5 Can progressive point matching learn from near-impossible task sets?

We now investigate whether progressive point matching can provide partial credit signal to learn from, even in the case where ground-truth reward signal is near zero. We train on a subset of the POPE-hard dataset (Qu et al., 2026) with verifiable numerical answers and with oracle solutions generated by gemini-2.5-pro. Prompts are filtered such that when evaluated at an 8K token budget at 16 trials, the base Qwen3-4B-Instruct model never produces a successful trajectory (see Appendix D. As shown in Table 5, the aggregate base model pass rate at 8K tokens is 0.004, and pass rate at 4K tokens is zero.

The best-performing baseline in this setting is the Privileged On-Policy Exploration (POPE) (Qu et al., 2026) algorithm, which modifies tasks

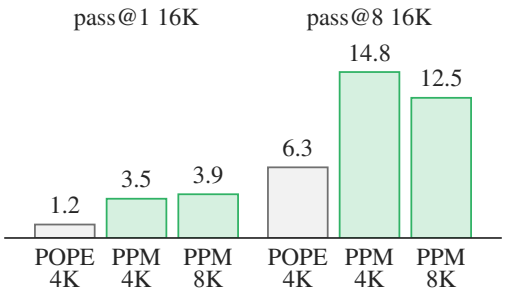


Figure 5: We train on the POPE-hard dataset (Qu et al., 2026) at length 4K or 8K tokens and evaluate at 16K tokens. Progressive point matching outperforms the best baseline (POPE). Moreover, the test-time performance is *comparable or better* when training on shorter sequences (4K).

by conditioning on prefixes of reference trajectories and instructing the model to continue solving the problem without restating any part of the reference trajectory. In our experiments (Figure 5, Table 5), POPE improves over GRPO and other baselines, but PPM displays the strongest performance overall.

Pitfall of greedy optimization. As shown in Figure 5, the progressive point matching policy trained at 4K tokens surprisingly outperforms the policy trained at 8K tokens, when evaluated at a test-time budget of 16K. We find that this is due to a collapse in output length – the policy trained at 8K length tends to greedily guess at the answer and end its reasoning process early. We hypothesize this is due to solve the task within the training length budget, biasing the policy toward greedy behavior. On the other hand, the policy trained at 4K length has no such pressure as its success rate is entirely zero, and thus all signal comes from reaching *intermediate* reasoning points. This finding highlights a surprising phenomenon where training at lower sequence lengths may be *beneficial*, and is a promising direction for future focus.

5 Discussion

The standard approach to reinforcement learning for LLMs requires exponentially many samples with the task horizon, manifesting in slow learning. To overcome this limitation, we introduce **Progressive Point Matching**, a framework that bridges RL and imitation learning by guiding exploration over the reasoning graph induced by a *single* reference trajectory per prompt. By rewarding the size of the set of reached intermediate reasoning points, progressive point matching can converge to the same set of optimal policies as with outcome rewards, learn diverse strategies not present in reference trajectories, and learn from hard math problems where existing RL approaches fail.

Progressive point matching enables strong performance improvements over prior approaches when evaluated at longer inference token budgets, as shown for math reasoning in both the standard (Section 4.4) and near-impossible (Section 4.5) data regimes. However, as we briefly discuss in Section 4.5, we hypothesize that there are three regimes based on the training token budgets. Specifically, **(i)** at very short sequences it is impossible for the policy to complete the task within the budget, hence the optimal behavior is to maximize partial progress; **(ii)** at medium-length sequences we experience greedy behavior due to length pressure; **(iii)** at long budgets there is no length pressure, hence the optimal behavior is to maximize task completion (Proposition 3.1). Our synthetic environments (Multi-Countdown, Matrix Manipulation, GSM-Infinite) generally operate within regime **(iii)**, while our math reasoning tasks operate in regime **(i)** or **(ii)**.

Limitations. We expect that, for each training dataset, there exists a *compute-optimal* training sequence length that maximizes evaluation performance at larger test-time token budgets. Eliciting extrapolation may therefore depend on training at an appropriate sequence length. For highly heterogeneous datasets, this could be achieved either by subsampling problems of suitable difficulty, as in Setlur et al. (2025), or by using our approach to reward partial progress in regimes **(i)** and **(iii)**. Standard outcome-reward RL cannot effectively train in regime **(i)**, because the policy never receives a nonzero reward signal; it must filter data as in prior work. In regime **(ii)**, however, partial-progress rewards may also be less effective, as both partial-progress and pure outcome-reward training may collapse to greedy behavior and thereby reduce exploration.

While this work covers such phenomena, empirical results are limited by scale, and would benefit from larger models, larger datasets to properly test generalization and scalability of reference trajectories, and longer-horizon tasks such as extended multi-turn interactions or agentic coding, in which the test-time token budget may be multiple orders of magnitude longer than the training token budget. We encourage future work to study the interplay between partial rewards and response lengths.

Future work. Practitioners should more rigorously characterize the three regimes described above and identify the interplay between length budget for RL, the prompt (data) mixture, and partial rewards. Another exciting direction – assuming access to sufficiently long-horizon tasks – is establishing scaling laws for partial credit assignment methods, with the objective of maximizing success rates at much larger test-time token budgets. Additionally, our framework introduces *reasoning points* and a method for evaluating them via Section 4.3; future work may explore training an LLM to improve reasoning point generation rather than following a manual and domain-specific procedure. Finally, we highlight that our approach relies *solely* on reference trajectories, from which we implicitly determine goal points. Therefore, progressive point matching and other methods improving on imitation learning may enable training in non-verifiable domains.

AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs in Appendix B, implementing methods, cleaning and reformatting datasets, and resolving grammar errors in the paper.

We have not used generative AI tools for generating synthetic datasets, helping theoretical models or conceptual frameworks, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, supporting qualitative and thematic data analysis, or interpreting results.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

In Appendix C we describe the experimental details required to reproduce all of our experiments, such as hyperparameters and implementation details. Our method can be reproduced in any standard LLM RL codebase; we will release an example implementation. The prompts used for progressive point matching and other methods are shown in Appendix E. We introduce several training environments and show examples of the tasks in Appendix D, as well as details on dataset construction. Our main theoretical result (Theorem 4.1) is proved in Appendix B, which provides a more formal explanation of the assumptions and claims, as well as a variety of examples to demonstrate extreme cases of reasoning graphs.

ACKNOWLEDGEMENTS

We thank the TPU research cloud (TRC) program for the compute resources that powered this work, and Google Cloud for the Gemini API.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018. URL https://arxiv.org/abs/1707.01495.

Locke Cai and Ivan Provilkov. Escaping the verifier: Learning to reason via demonstrations. *arXiv preprint arXiv:2511.21667*, 2025.

Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, et al. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *arXiv preprint arXiv:2411.04282*, 2024.

Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of llm reasoning. *arXiv preprint arXiv:2509.25666*, 2025.

Zhoujun Cheng*, Yutao Xie*, Yuxiao Qu*, Amrith Setlur*, Shibo Hao, Varad Pimpalkhute, Tongtong Liang, Feng Yao, Hector Liu, Eric Xing, Virginia Smith, Ruslan Salakhutdinov, Zhiting Hu, Taylor Killian, and Aviral Kumar. Isocompute playbook: Optimally scaling sampling compute for rl training of llms, 2026. URL https://compute-optimal-rl-llm-scaling.github.io/.

Chia-Cheng Chiang, Li-Cheng Lan, Wei-Fang Sun, Chien Feng, Cho-Jui Hsieh, and Chun-Yi Lee. Expert proximity as surrogate rewards for single demonstration imitation learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8336–8358. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chiang24a.html.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, Matthieu Geist, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation, 2022. URL https://arxiv.org/abs/2106.12142.

Google DeepMind. Gemini 2.5 flash-lite model card. Technical report, Google, 2025a. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Flash-Lite-Model-Card.pdf.

Google DeepMind. Gemini 3 flash model card. Technical report, Google, 2025b. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29, pp. 4565–4573. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. In Steven M. LaValle, Ming Lin, Timo Ojala, Dylan Shell, and Jingjin Yu (eds.), *Algorithmic Foundations of Robotics XIV*, pp. 313–329, Cham, 2021. Springer International Publishing. ISBN 978-3-030-66723-8. doi: 10.1007/978-3-030-66723-8_19. URL https://link.springer.com/chapter/10.1007/978-3-030-66723-8_19.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *CoRR*, abs/1912.05032, 2019. URL http://arxiv.org/abs/1912.05032.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

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, 2023. URL https://arxiv.org/abs/2305.20050.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

Yecheng Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Versatile offline imitation from observations and examples via regularized state-occupancy matching. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14639–14663. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ma22a.html.

Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*, 2023.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/cf9a242b70f45317ffd281241fa66502-Paper.pdf.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, pp. 305–313, 1988. URL https://papers.nips.cc/paper/1988/hash/812b4ba287f5ee0bc9d43bbf5bbe87fb-Abstract.html.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. Rlad: Training llms to discover abstractions for solving reasoning problems. *arXiv preprint arXiv:2510.02263*, 2025.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. SQIL: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xKd24twB.

Cursor Research, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, et al. Composer 2 technical report. *arXiv preprint arXiv:2603.24477*, 2026.

Stephane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning, 2014. URL https://arxiv.org/abs/1406.5979.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Yannick Schroecker and Charles Isbell. State aware imitation learning. In *Advances in Neural Information Processing Systems*, pp. 2911–2920, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/08e6bea8e90ba87af3c9554d94db6579-Abstract.html.

John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Amrith Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms, 2025. URL https://arxiv.org/abs/2506.09026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760.

Wen Sun, Arun Venkatraman, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3309–3318. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sun17d.html.

Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6036–6045. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/sun19b.html.

Yunhao Tang, Sid Wang, Rémi Munos, et al. Learning to chain-of-thought with jensen's evidence lower bound. *arXiv e-prints*, pp. arXiv–2503, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024. URL https://arxiv.org/abs/2312.08935.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan 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, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An

open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E Gonzalez. The wisdom of hindsight makes language models better instruction followers. In *International Conference on Machine Learning*, pp. 41414–41428. PMLR, 2023.

Xin Zhang, Yanhua Li, Ziming Zhang, and Zhi-Li Zhang. f-GAIL: Learning f-divergence for generative adversarial imitation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12805–12815. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/967990de5b3eac7b87d49a13c6834978-Abstract.html.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *arXiv preprint arXiv:2505.21493*, 2025a.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity?, 2025b. URL https://arxiv.org/abs/2502.05252.

A Related Work

Partial progress rewards. Learning from sparse outcome rewards has been a long-standing challenge in RL due to their low signal and high exploration requirements. A number of prior works have proposed methodologies to reward *partial progress*. In principle, multi-task value learning methods (Schaul et al., 2015; Yue et al., 2025; Schulman et al., 2017; Snell et al., 2022) provide a notion of partial credit, as trajectories that fail to reach the answer can nevertheless be rewarded by a learned estimate of their answer-reaching likelihood. Hindsight relabeling methods (Andrychowicz et al., 2018; Zhang et al., 2023) also provide signal by rewriting the original goal such that failing trajectories now succeed (for a different goal).

Particularly for language models, a large body of work has studied process rewards, which provide feedback to intermediate reasoning by checking whether each segment is logically correct. This feedback may come from a human annotated dataset (Lightman et al., 2023; Uesato et al., 2022; Zheng et al., 2025) or via querying a language model judge (Miao et al., 2023; Khalifa et al., 2025). However, training on correctness rewards may not converge to an optimal policy under outcome rewards, as logical correctness may not correlate with task success (Setlur et al., 2024; Zhang et al., 2025). Closest to our work are methods that do not rely on external definitions of correctness and attempt to predict success rate directly, largely via Monte Carlo estimation (Luo et al., 2024; Wang et al., 2024; Cui et al., 2025). While we share a similar viewpoint of directly rewarding task success, our methodology avoids learning an expensive reward or value model entirely, and instead uses per-task reference trajectories to provide partial credit.

Imitation learning and inverse reinforcement learning. Our work takes inspiration from imitation learning methods that derive signal from reference policies or trajectories. The simplest of these are behavioral cloning methods (Pomerleau, 1988), which perform supervised learning on actions over a dataset of reference trajectories. Online imitation learning methods (Ross et al., 2011; Ross & Bagnell, 2014; Sun et al., 2017) instead aim to minimize distribution drift by directly operating on sampled trajectories, copying teacher actions at each state. If an explicit teacher model is not available, some methods resort to learning an adversarial critic (Ho & Ermon, 2016; Ke et al., 2021; Zhang et al., 2020), which can be used to provide signal to the student. Other methods pose imitation learning as a *state occupancy matching* problem (Nachum et al., 2019; Kostrikov et al., 2019; Ma et al., 2022; Sun et al., 2019; Garg et al., 2022), which train policies to exactly match the steady-state distribution of a teacher. Our approach is most closely related to (Schroecker & Isbell, 2017; Reddy et al., 2020; Chiang et al., 2024), which directly construct on-policy rewards derived from demonstrations. While similar in inspiration, our method critically differs in that we do not aim to match state *occupancies*, which involves matching the distribution of all intermediate states, but rather we reward progress as the set of reached reasoning points, giving credit to intermediate points via a *shortcutting* mechanism. Thus, progressive point matching enables policies to learn fundamentally different goal-reaching strategies than the demonstrations.

Language model training via privileged information. Prior works have examined mechanics to best extract learning signal from demonstrations or reference solutions. Strategies involve placing privileged information in-context, e.g. partial solutions (Qu et al., 2026) or reasoning abstractions (Qu et al., 2025; Chen et al., 2025). These in-context policies can also be treated as a target for distillation (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026). However, directly providing privileged information creates a risk of distribution shift (Zelikman et al., 2022) or overfitting, characterized by a collapse in exploration ability (Qu et al., 2026). In contrast, our method is most similar to works that learn fully on-policy and use privileged information only to construct rewards. This has often taken the form of using likelihood-based rewards (Chen et al., 2024; Tang et al., 2025; Zhou et al., 2025a) or adversarial critics (Cai & Provilkov, 2025). However, instead of relying on model likelihoods, we draw a tight connection to goal-reaching MDPs and opt to directly reward partial progress.

B Progressive Point Matching as Variance Reduction

Assume that the task horizon is $T = n$, and at turn t, the policy always attempts to perform the reasoning at point t. Let P_v be the set of prerequisites of v, including v and excluding 0, and $d_v = |P_v|$. Let $B_v \sim \text{Bernoulli}(p)$ be independent across v where $p \in (0, 1)$ is fixed, and define $I_v = \prod_{j \in P_v} B_j$ to be an indicator for whether v has been reached. Similarly define I_g.

Since the score $z_t = \nabla_\theta \log \pi_\theta(a_t|s_t)$ is high-dimensional (say $z_t, \theta \in \mathbb{R}^k$), we instead analyze the variance of its projection $x_t = u^T z_t$ along an arbitrary unit vector u. However, we consider only the asymptotic behavior of each estimator with n, which does not depend on the choice of u.

We assume that pairs (B_v, x_v) are independent over v. Define constants μ, λ, η independent of v and n satisfying

$$\mathbb{E}[x_v] = 0, \quad \mathbb{E}[B_v x_v] = \mu \neq 0, \quad \mathbb{E}[B_v x_v^2] = \lambda, \quad \mathbb{E}[x_v^2] = \eta. \tag{6}$$

By Cauchy–Schwarz, we have

$$\mu^2 = \mathbb{E}[B_v x_v]^2 \leq \mathbb{E}[B_v]\mathbb{E}[B_v x_v^2] = p\lambda, \tag{7}$$

so $\mu \neq 0$ and $p < 1$ imply $\lambda \geq \mu^2/p > \mu^2$.

We are now ready to define the three gradient estimates:

$$\text{Sparse outcome reward:} \quad Y_1 = u^T \hat{g}_1 = I_g \sum_{v=1}^{n} x_v \tag{8}$$

$$\text{Progressive point matching trajectory-level:} \quad Y_2 = u^T \hat{g}_2 = \frac{1}{n} \left(\sum_{v=1}^{n} I_v \right) \left(\sum_{v=1}^{n} x_v \right) \tag{9}$$

$$\text{Progressive point matching segment-level:} \quad Y_3 = u^T \hat{g}_3 = \frac{1}{n} \sum_{v=1}^{n} I_v x_v. \tag{10}$$

In this appendix we characterize the relationship between the signal-to-noise ratio, $\mathrm{SNR}(Y) := |\mathbb{E}[Y]|/\sqrt{\mathrm{Var}(Y)}$, and the task horizon n of each of these estimators. Coarsely, our main theoretical results can be summarized as

$$\mathrm{SNR}(Y_1) \ll \mathrm{SNR}(Y_2) \ll \mathrm{SNR}(Y_3), \tag{11}$$

where the latter inequality is shown in Proposition B.10. Additionally, we provide worst-case analyses for the trajectory-level and turn-level progressive point matching gradient estimators. The general case lower bounds increase *exponentially* from Y_1 to Y_2 and as $\mathrm{poly}(n)$ from Y_2 to Y_3, as shown in Propositions B.2 and B.7. Subject to additional regularity conditions that upper-bound the correlations between the reachability of points in the graph, these lower bounds can be improved by $\mathrm{poly}(n)$. Under these assumptions, Propositions B.1 and B.3 and corollary B.8 comprise Theorem 4.1.

We used OpenAI's GPT-5.6 Sol to assist in developing and refining portions of the mathematical proofs. We subsequently checked, revised, and independently verified all resulting arguments.

B.1 SNR VIA SPARSE OUTCOME REWARDS

Proposition B.1. $\mathrm{SNR}(Y_1) = \Theta(p^{n/2})$.

Proof. First, for every $v \in \{1, \ldots, n\}$,

$$\mathbb{E}[I_g x_v] = \mathbb{E}[B_v x_v] \prod_{\substack{j=1 \\ j \neq v}}^{n} \mathbb{E}[B_j] = \mu p^{n-1}, \tag{12}$$

hence $\mathbb{E}[Y_1] = n\mu p^{n-1}$. Next, we compute the second moment:

$$\mathbb{E}[Y_1^2] = \sum_{v=1}^{n} \mathbb{E}[I_g x_v^2] + \sum_{v \neq w} \mathbb{E}[I_g x_v x_w]. \tag{13}$$

For the diagonal terms,

$$\mathbb{E}[I_g x_v^2] = \mathbb{E}[B_v x_v^2] \prod_{\substack{1 \leq j \leq n \\ j \neq v}} \mathbb{E}[B_j] = \lambda p^{n-1}. \tag{14}$$

For the off-diagonal terms,

$$\mathbb{E}[I_g x_v x_w] = \mathbb{E}[B_v x_v]\mathbb{E}[B_w x_w] \prod_{\substack{1\le j\le n \\ j\notin\{v,w\}}} \mathbb{E}[B_j] = \mu^2 p^{n-2}. \tag{15}$$

Subtracting $\mathbb{E}[Y_1]^2$ from the total, we obtain

$$\mathrm{Var}(Y_1) = n\lambda p^{n-1} + n(n-1)\mu^2 p^{n-2} - n^2\mu^2 p^{2n-2} = \Theta(n^2 p^{n-2}). \tag{16}$$

Finally,

$$\mathrm{SNR}(Y_1)^2 = \frac{\mathbb{E}[Y_1]^2}{\mathrm{Var}(Y_1)} = \frac{\Theta(n^2 p^{2n-2})}{\Theta(n^2 p^{n-2})} = \Theta(p^n), \tag{17}$$

as claimed. □

B.2 SNR VIA TRAJECTORY-LEVEL PROGRESSIVE POINT MATCHING

We analyze the estimator

$$Y_2 = \frac{1}{n}\left(\sum_{v=1}^{n} I_v\right)\left(\sum_{v=1}^{n} x_v\right). \tag{18}$$

We define random variables

$$C := \sum_{v=1}^{n} I_v, \quad D := \sum_{v=1}^{n} d_v I_v, \tag{19}$$

where C is the count of reached points, and D is the depth over reached points. For pairs of points, also define $d_{vw} := |P_v \cup P_w|$. Throughout the proofs, we will leverage the following facts:

$$\mathbb{E}[C] = \sum_{v=1}^{n} p^{d_v}, \quad \mathbb{E}[D] = \sum_{v=1}^{n} d_v p^{d_v}, \quad \mathbb{E}[C^2] = \sum_{v=1}^{n}\sum_{w=1}^{n} p^{d_{vw}}. \tag{20}$$

Additionally, observe that since every DAG has a source node v with $d_v = 1$, we have

$$\mathbb{E}[D] \ge \mathbb{E}[C] \ge p. \tag{21}$$

Proposition B.2. $\mathrm{SNR}(Y_2) = \Omega(\sqrt{\log n}/n)$.

Proposition B.3. *Suppose that*

$$\mathbb{E}[C^2] = \Theta\big(\mathbb{E}[D]^2\big) \tag{22}$$

and

$$\sum_{v=1}^{n}\sum_{w=1}^{n} d_{vw}^2 p^{d_{vw}} = O\big(\mathbb{E}[C^2]\big). \tag{23}$$

Then $\mathrm{SNR}(Y_2) = \Theta(n^{-1/2})$.

Before providing a proof of Propositions B.2 and B.3, we first provide some intuition for the assumptions, which encapsulate the reasoning graphs induced by both Multi-Countdown and Matrix Manipulation MDPs.

The first condition can be rewritten as

$$\frac{\sqrt{\mathbb{E}[C^2]}}{\mathbb{E}[C]} = \Theta\left(\frac{\mathbb{E}[D]}{\mathbb{E}[C]}\right). \tag{24}$$

The left hand side measures the "burstiness" of the total number of reached reasoning points, and the right hand side measures the average depth of reached reasoning points,

$$\frac{\mathbb{E}[D]}{\mathbb{E}[C]} = \frac{\sum_v d_v p^{d_v}}{\sum_v p^{d_v}}. \tag{25}$$

By requiring this ratio to remain $\Theta(1)$, this condition prevents deep and jointly reachable pairs (v, w) from dominating the second moment of Y_2.

As we will see in the following examples, the dandelion graph has highly correlated visitation events, producing bursty C. On the other hand, a complete binary tree of depth h with $p = 1/2$ has $\mathbb{E}[D] = \Theta(h^2)$ and $\mathbb{E}[C] = \Theta(h)$, achieving high average depth.

For the second condition, observe that $p^{d_{vw}} = \Pr(I_v = I_w = 1)$. Consequently,

$$\frac{\sum_{v,w} d_{vw}^2 p^{d_{vw}}}{\mathbb{E}[C^2]} = \frac{\sum_{v,w} d_{vw}^2 \Pr(I_v = I_w = 1)}{\sum_{v,w} \Pr(I_v = I_w = 1)}. \tag{26}$$

The condition requires that this ratio is $\Theta(1)$, which prevents deep and jointly reachable pairs (v, w) from dominating the second moment of Y_2.

- The **Multi-Countdown** case satisfies the assumptions of Proposition B.3. Here $P_v = \{v\}$, so

$$\mathbb{E}[D] = np = \Theta(n), \quad \mathbb{E}[C^2] = np + n(n-1)p^2 = \Theta(n^2) = \Theta\big(\mathbb{E}[D]^2\big). \tag{27}$$

  Moreover,

$$\sum_{v,w} d_{vw}^2 p^{d_{vw}} = np + 4n(n-1)p^2 = \Theta(n^2) = O\big(\mathbb{E}[C^2]\big). \tag{28}$$

- The **Matrix Manipulation** case also satisfies the assumptions of Proposition B.3. Here $P_v = \{1, \dots, v\}$ and $d_{vw} = \max(v, w)$, so

$$\mathbb{E}[D] = \sum_{v=1}^{n} v p^v = \Theta(1), \quad \mathbb{E}[C^2] = \sum_{k=1}^{n} (2k-1)p^k = \Theta(1) = \Theta\big(\mathbb{E}[D]^2\big). \tag{29}$$

  The second condition follows from

$$\sum_{v,w} d_{vw}^2 p^{d_{vw}} = \sum_{k=1}^{n} (2k-1)k^2 p^k = \Theta(1) = O\big(\mathbb{E}[C^2]\big). \tag{30}$$

- The lower bound in Proposition B.2 is achieved by the **dandelion graph**. Fix $r = \lfloor \log_{1/p}(n \log n) \rfloor$, and construct the graph $0 \to 1 \to \cdots \to r$, and for each leaf $v > r$ create a dependency $r \to v$. Intuitively, conditioned on passing point r, $\Theta(n)$ leaves become reachable together, which produces $\Theta(n^2)$ highly correlated leaf pairs, corresponding to a low SNR. We show in Example B.6 that

$$\mathbb{E}[D] = \Theta(1), \quad \mathbb{E}[C^2] = \Theta\left(\frac{n}{\log n}\right), \tag{31}$$

  which notably do not satisfy the assumptions of Proposition B.3.

In preparation for the proof of Proposition B.2, we first bound the second moment of reached progress $\mathbb{E}[C^2]$ in terms of expected cumulative depth $\mathbb{E}[D]$ (Corollary B.5), with the help of the following analytic lemma. This relation appears when combining on-diagonal and off-diagonal terms in the expansion of $\mathbb{E}[Y_2^2]$ (Equation (53)).

Lemma B.4. *Let $q_1, \dots, q_n \in (0, 1]$, and define*

$$A_n = \sum_{v=1}^{n} q_v (1 + \log(1/q_v)). \tag{32}$$

Then

$$\sum_{v,w} \min(q_v, q_w) = O\left(\frac{n A_n}{1 + \log(n/A_n)}\right). \tag{33}$$

Proof. Define $F(t) = |\{v : q_v \geq t\}|$, so that

$$\sum_{v,w} \min(q_v, q_w) = \int_0^1 \left(\sum_v \mathbf{1}\{q_v \geq t\}\right)^2 dt = \int_0^1 F(t)^2 dt. \tag{34}$$

It remains to bound $F(t)$. Let $g(t) = t(1 + \log(1/t))$ be increasing. If $q_v \geq t$, then

$$q_v(1 + \log(1/q_v)) = g(q_v) \geq g(t). \tag{35}$$

Summing over the $F(t)$ values of v with $q_v \geq t$, we obtain

$$A_n \geq F(t)t(1 + \log(1/t)), \tag{36}$$

hence

$$F(t) \leq \min\left(n, \frac{A_n}{t(1 + \log(1/t))}\right). \tag{37}$$

Take $\ell_n = 1 + \log(n/A_n)$ and $t_0 = A_n/(n\ell_n)$. Since $A_n \leq n$, we have $0 < t_0 \leq 1$, so

$$\int_0^1 F(t)^2 dt \leq n^2 t_0 + A_n^2 \int_{t_0}^1 \frac{dt}{t^2(1 + \log(1/t))^2}. \tag{38}$$

A change of variables $x = 1/t$ gives

$$\int_{t_0}^1 \frac{dt}{t^2(1 + \log(1/t))^2} = \int_1^{1/t_0} \frac{dx}{(1 + \log x)^2} = O\left(\frac{1}{t_0(1 + \log(1/t_0))^2}\right). \tag{39}$$

It follows that

$$\int_0^1 F(t)^2 dt \leq n^2 t_0 + O\left(\frac{A_n^2}{t_0 \ell_n^2}\right) = O\left(\frac{nA_n}{\ell_n}\right), \tag{40}$$

as desired. □

Corollary B.5. *Let* $\ell_n = 1 + \log_+(n/\mathbb{E}[D])$, *where* $\log_+(x) = \max(0, \log x)$. *Then*

$$\mathbb{E}[C^2] = O\left(\frac{n\mathbb{E}[D]}{\ell_n}\right). \tag{41}$$

Proof. Set $q_v := \mathbb{P}(I_v = 1) = p^{d_v}$. Since $d_{vw} \geq \max(d_v, d_w)$,

$$\mathbb{E}[C^2] = \sum_{v,w} p^{d_{vw}} \leq \sum_{v,w} \min(q_v, q_w). \tag{42}$$

For the quantity A_n in Lemma B.4,

$$A_n = \sum_v q_v \left(1 + \log(1/q_v)\right) = \Theta\left(\sum_v d_v q_v\right) = \Theta(\mathbb{E}[D]), \tag{43}$$

where the constants depend only on the fixed p. It follows that $1 + \log(n/A_n) = \Theta(\ell_n)$. Therefore, Lemma B.4 gives

$$\mathbb{E}[C^2] \leq \sum_{v,w} \min(q_v, q_w) = O\left(\frac{nA_n}{1 + \log(n/A_n)}\right) = O\left(\frac{n\mathbb{E}[D]}{\ell_n}\right), \tag{44}$$

which proves the result. □

Proof of Proposition B.2. As shown in the proof of Proposition B.1,

$$\mathbb{E}[I_v x_i] = \begin{cases} \mu p^{d_v - 1}, & i \in P_v, \\ 0, & i \notin P_v. \end{cases} \tag{45}$$

Therefore

$$\mathbb{E}[Y_2] = \frac{\mu}{pn}\mathbb{E}[D]. \tag{46}$$

For the second moment, let $P_{vw} := P_v \cup P_w$, whose cardinality is d_{vw}. Since $I_v I_w = \prod_{j \in P_{vw}} B_j$, expanding $(\sum_i x_i)^2$ gives diagonal terms

$$\mathbb{E}[I_v I_w x_i^2] = \begin{cases} \lambda p^{d_{vw} - 1}, & i \in P_{vw}, \\ \eta p^{d_{vw}}, & i \notin P_{vw}, \end{cases} \tag{47}$$

and, for $i \neq j$, off-diagonal terms

$$\mathbb{E}[I_v I_w x_i x_j] = \begin{cases} \mu^2 p^{d_{vw}-2}, & i,j \in P_{vw}, \\ 0, & \text{otherwise.} \end{cases} \tag{48}$$

There are respectively d_{vw}, $n - d_{vw}$, and $d_{vw}(d_{vw}-1)$ terms of these three types. Therefore

$$\mathbb{E}[Y_2^2] = \frac{1}{n^2} \sum_{v,w} \left[d_{vw} \lambda p^{d_{vw}-1} + (n - d_{vw}) \eta p^{d_{vw}} + d_{vw}(d_{vw}-1) \mu^2 p^{d_{vw}-2} \right]. \tag{49}$$

The middle term is $O(\mathbb{E}[C^2]/n)$. For the first and third terms, we can bound

$$d_{vw} \lambda p^{d_{vw}-1} + d_{vw}(d_{vw}-1)\mu^2 p^{d_{vw}-2} \leq O(d_{vw}^2 p^{d_{vw}-2}) \leq O(d_v^2 p^{d_v - 1} + d_w^2 p^{d_w - 1}), \tag{50}$$

where the second inequality follows from $d_{vw} \geq \max(d_v, d_w)$. Consequently, their total contribution T satisfies

$$T = O\left(\frac{1}{n^2} \sum_{v,w} d_{vw}^2 p^{d_{vw}-2} \right) = O\left(\frac{1}{n} \sum_{v} d_v^2 p^{d_v - 1} \right). \tag{51}$$

To bound the remaining sum, take $L = \lceil 3 \log_{1/p} n \rceil$. points with $d_v \leq L$ contribute at most $O(L\mathbb{E}[D])$. Since $d^2 p^d$ decreases exponentially beyond L, the at most n points with $d_v > L$ contribute $O(nL^2 p^L) = o(1) = O(\mathbb{E}[D])$, where the last equality follows from Equation (21). Therefore

$$\sum_v d_v^2 p^{d_v - 1} = O(\mathbb{E}[D] \log n), \tag{52}$$

and hence $T = O(\mathbb{E}[D] \log n / n)$. It follows that

$$\mathbb{E}[Y_2^2] = O\left(\frac{\mathbb{E}[C^2]}{n} + \frac{\mathbb{E}[D] \log n}{n} \right) = O\left(\frac{\mathbb{E}[D]}{\ell_n} \right), \tag{53}$$

where the final equality uses the preceding corollary together with $\ell_n \log n = O(n)$. Consequently,

$$\mathrm{SNR}(Y_2)^2 \geq \frac{\mathbb{E}[Y_2]^2}{\mathbb{E}[Y_2^2]} = \Omega\left(\frac{\mathbb{E}[D]\, \ell_n}{n^2} \right). \tag{54}$$

Equation (21) gives $\mathbb{E}[D] \geq p$. If $\mathbb{E}[D] \leq n$, monotonicity of $t \mapsto t(1 + \log(n/t))$ on $[p, n]$ gives $\mathbb{E}[D]\ell_n = \Omega(\log n)$. If $\mathbb{E}[D] > n$, then $\ell_n = 1$ and the same conclusion is immediate. Thus $\mathrm{SNR}(Y_2)^2 = \Omega(\log n / n^2)$ as claimed. □

Proof of Proposition B.3. The middle term in Equation (49) satisfies

$$\sum_{v,w} (n - d_{vw}) p^{d_{vw}} = n\mathbb{E}[C^2] - \sum_{v,w} d_{vw} p^{d_{vw}} = \Theta\big(n\mathbb{E}[C^2]\big). \tag{55}$$

The second regularity condition makes the remaining terms lower order, so

$$\mathbb{E}[Y_2^2] = \Theta\left(\frac{\mathbb{E}[C^2]}{n} \right) = \Theta\left(\frac{\mathbb{E}[D]^2}{n} \right). \tag{56}$$

Since $\mathbb{E}[Y_2] = \mu \mathbb{E}[D]/(pn)$, its squared mean is lower order and

$$\mathrm{SNR}(Y_2)^2 = \frac{\Theta(\mathbb{E}[D]^2/n^2)}{\Theta(\mathbb{E}[D]^2/n)} = \Theta(n^{-1}) \tag{57}$$

as desired. □

Example B.6. *The dandelion graph with $r = \lfloor \log_{1/p}(n \log n) \rfloor$ achieves the $\Theta(\sqrt{\log n}/n)$ lower bound of Proposition B.2.*

Proof. In this graph,

$$\mathbb{E}[D] = \sum_{v=1}^{r} vp^v + \sum_{v=r+1}^{n} (r+1)p^{r+1} = \Theta(1). \tag{58}$$

Therefore $\mathbb{E}[Y_2] = \mu\mathbb{E}[D]/(pn) = \Theta(1/n)$.

By the choice of r, we have $r = \Theta(\log n)$ and $p^r = \Theta(1/(n \log n))$. Decomposing the sum in $\mathbb{E}[C^2]$ gives

$$\sum_{v,w \le r} p^{d_{vw}} = \sum_{k=1}^{r} (2k-1)p^k = \Theta(1), \tag{59}$$

$$2\sum_{\substack{v \le r \\ w > r}} p^{d_{vw}} = 2r(n-r)p^{r+1} = \Theta(1), \tag{60}$$

$$\sum_{v > r} p^{d_{vv}} = (n-r)p^{r+1} = \Theta(1/\log n), \tag{61}$$

$$\sum_{\substack{v,w > r \\ v \ne w}} p^{d_{vw}} = (n-r)(n-r-1)p^{r+2} = \Theta(n/\log n). \tag{62}$$

Thus $\mathbb{E}[C^2] = \Theta(n/\log n)$. It follows from Equation (53) that

$$\mathbb{E}[Y_2^2] = O\left(\frac{\mathbb{E}[C^2]}{n} + \frac{\mathbb{E}[D]\log n}{n}\right) = O(1/\log n). \tag{63}$$

We can additionally lower-bound $\mathbb{E}[Y_2^2]$ via Equation (49),

$$\mathbb{E}[Y_2^2] \ge \frac{1}{n^2} \sum_{\substack{v,w > r \\ v \ne w}} (n - d_{vw})\eta p^{d_{vw}} = \Theta(1/\log n). \tag{64}$$

Hence $\mathrm{Var}(Y_2) = \Theta(1/\log n)$ and $\mathrm{SNR}(Y_2) = \Theta(\sqrt{\log n}/n)$. □

B.3 SNR VIA TURN-LEVEL PROGRESSIVE POINT MATCHING

We analyze the estimator

$$Y_3 = \frac{1}{n}\sum_{v=1}^{n} I_v x_v. \tag{65}$$

Proposition B.7. *We have*

$$\mathrm{SNR}(Y_3) = \frac{\mathbb{E}[C]}{\sqrt{\mathrm{Var}(C) + \kappa\,\mathbb{E}[C]}}, \quad \kappa := \frac{p\lambda}{\mu^2} - 1 \ge 0. \tag{66}$$

Consequently,

$$\mathrm{SNR}(Y_3) = \Theta\left(\frac{\mathbb{E}[C]}{\sqrt{\mathrm{Var}(C)}}\right) \ge \Omega(n^{-1/2}). \tag{67}$$

As we discuss in the examples below, this lower bound is achievable by a dandelion graph, and in this setting $\mathrm{SNR}(Y_2) = \mathrm{SNR}(Y_3) = \Theta(n^{-1/2})$. However, as we show below, for non-pathological graphs, we realize the benefits of turn-level credit assignment, i.e. $\mathrm{SNR}(Y_2) \ll \mathrm{SNR}(Y_3)$.

Corollary B.8. *If* $\mathrm{Var}(C) = O(\mathbb{E}[C])$, *then* $\mathrm{SNR}(Y_3) = \Theta\big(\sqrt{\mathbb{E}[C]}\big)$.

Proposition B.9. *If* $\mathbb{E}[C^2] = O\big(\mathbb{E}[D]^2\big)$, *then*

$$\mathrm{SNR}(Y_3) = \Omega\left(\frac{\mathbb{E}[C]}{\mathbb{E}[D]}\right) = \Omega(1/\log n). \tag{68}$$

Proposition B.10. $\mathrm{SNR}(Y_3)/\mathrm{SNR}(Y_2) = \Omega(\sqrt{n}/\log n)$.

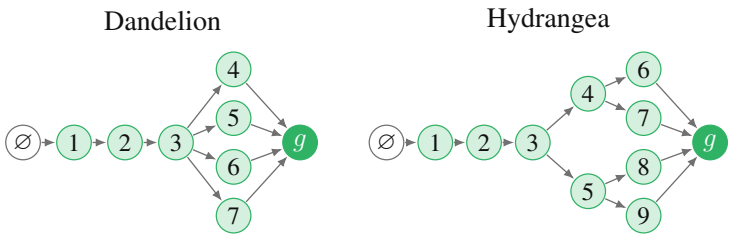


Figure 6: Dandelion graphs achieve the worst-case SNR bounds in Propositions B.2 and B.7, since a large number of visitation events are highly correlated conditioned on the low-probability event of reaching point $r = 3$. With an additional condition on pairwise reachability, we achieve a better worst case (Proposition B.9). This worst case is the Hydrangea graph (Example B.11), which distributes correlations over multiple tree depths.

As above, we first illustrate how the moments of C encode the relevant graph structure.

- In **Multi-Countdown**, C is binomial with $\mathbb{E}[C] = np$ and $\mathrm{Var}(C) = np(1-p)$. Hence Corollary B.8 gives $\mathrm{SNR}(Y_3) = \Theta(\sqrt{n})$.
- In **Matrix Manipulation**, $d_{vw} = \max(v, w)$, so

$$\mathbb{E}[C] = \sum_{v=1}^{n} p^v = \Theta(1), \tag{69}$$

$$\mathbb{E}[C^2] = \sum_{v,w} p^{d_{vw}} = \sum_{k=1}^{n} (2k-1)p^k = \Theta(1). \tag{70}$$

  Together with $\mathrm{Var}(C) \geq (1-p)\mathbb{E}[C]$, this gives $\mathrm{Var}(C) = \Theta(1)$, so Corollary B.8 gives $\mathrm{SNR}(Y_3) = \Theta(1)$.
- For the **dandelion graph** with $r = \lfloor \log_{1/p} n \rfloor$, we have $p^r = \Theta(1/n)$ and

$$\mathbb{E}[C] = \sum_{v=1}^{r} p^v + (n-r)p^{r+1} = \Theta(1), \tag{71}$$

$$\mathbb{E}[D] = \sum_{v=1}^{r} vp^v + (n-r)(r+1)p^{r+1} = \Theta(\log n). \tag{72}$$

  Distinct leaf pairs contribute

$$(n-r)(n-r-1)p^{r+2} = \Theta(n) \tag{73}$$

  to $\mathbb{E}[C^2]$. The remaining pair classes satisfy

$$\sum_{v,w \leq r} p^{d_{vw}} = \Theta(1), \tag{74}$$

$$2\sum_{\substack{v \leq r \\ w > r}} p^{d_{vw}} = 2r(n-r)p^{r+1} = \Theta(\log n), \tag{75}$$

$$\sum_{v > r} p^{d_{vv}} = (n-r)p^{r+1} = \Theta(1). \tag{76}$$

  Hence $\mathbb{E}[C^2] = \Theta(n)$, so $\mathrm{Var}(C) = \Theta(n)$ and Proposition B.7 gives $\mathrm{SNR}(Y_3) = \Theta(n^{-1/2})$. This graph does not satisfy $\mathbb{E}[C^2] = O(\mathbb{E}[D]^2)$.
- Under the condition $\mathbb{E}[C^2] = O(\mathbb{E}[D]^2)$, the **hydrangea graph** in Example B.11 attains the lower bound in Proposition B.9. It has

$$\mathbb{E}[C] = \Theta(1), \quad \mathbb{E}[D] = \Theta(\log n), \quad \mathbb{E}[C^2] = \Theta((\log n)^2). \tag{77}$$

  Consequently $\mathrm{Var}(C) = \Theta((\log n)^2)$ and Proposition B.7 gives $\mathrm{SNR}(Y_3) = \Theta(1/\log n)$.

Proof of Proposition B.7. For each v,

$$\mathbb{E}[I_v x_v] = \mathbb{E}[B_v x_v] \prod_{j \in P_v \setminus \{v\}} \mathbb{E}[B_j] = \mu p^{d_v - 1}, \tag{78}$$

where the factorization follows from independence over v. Therefore, by linearity and Equation (20),

$$\mathbb{E}[Y_3] = \frac{1}{n} \sum_{v=1}^{n} \mathbb{E}[I_v x_v] = \frac{\mu}{n} \sum_{v=1}^{n} p^{d_v - 1} = \frac{\mu}{pn} \mathbb{E}[C]. \tag{79}$$

For the second moment, the diagonal terms satisfy $\mathbb{E}[I_v x_v^2] = \lambda p^{d_v - 1}$, while for $v \neq w$, $\mathbb{E}[I_v I_w x_v x_w] = \mu^2 p^{d_{vw} - 2}$. Hence

$$\mathbb{E}[Y_3^2] = \frac{1}{n^2} \left[\frac{\lambda}{p} \mathbb{E}[C] + \frac{\mu^2}{p^2} \big(\mathbb{E}[C^2] - \mathbb{E}[C] \big) \right] = \frac{\mu^2 \mathbb{E}[C^2] + (p\lambda - \mu^2)\mathbb{E}[C]}{p^2 n^2}. \tag{80}$$

Subtracting $\mathbb{E}[Y_3]^2 = (\mu^2/(p^2 n^2))\mathbb{E}[C]^2$ gives

$$\mathrm{Var}(Y_3) = \frac{\mu^2 \mathrm{Var}(C) + (p\lambda - \mu^2)\mathbb{E}[C]}{p^2 n^2}. \tag{81}$$

Factoring $\mu^2/(p^2 n^2)$ from the variance and cancelling $|\mu|/(pn)$ from the signal and standard deviation proves

$$\mathrm{SNR}(Y_3) = \frac{\mathbb{E}[C]}{\sqrt{\mathrm{Var}(C) + \left(\frac{p\lambda}{\mu^2} - 1 \right) \mathbb{E}[C]}}, \tag{82}$$

which is Equation (66).

For distinct v, w, we have $d_{vw} \leq d_v + d_w$, and hence

$$\mathrm{Cov}(I_v, I_w) = p^{d_{vw}} - p^{d_v + d_w} \geq 0. \tag{83}$$

Also, $d_v \geq 1$ implies

$$\mathrm{Var}(I_v) = p^{d_v}(1 - p^{d_v}) \geq (1 - p)\mathbb{E}[I_v]. \tag{84}$$

Summing gives

$$\mathrm{Var}(C) \geq \sum_v \mathrm{Var}(I_v) \geq (1 - p)\mathbb{E}[C]. \tag{85}$$

Thus the additional $\kappa \mathbb{E}[C]$ term in Equation (66) is $O(\mathrm{Var}(C))$, which proves the Θ characterization in Equation (67).

For the universal lower bound, $0 \leq C \leq n$ gives

$$\mathrm{Var}(C) \leq \mathbb{E}[C^2] \leq n\mathbb{E}[C]. \tag{86}$$

By Equation (21), $\mathbb{E}[C] \geq p$. Since κ is fixed, we obtain

$$\mathrm{SNR}(Y_3)^2 \geq \frac{\mathbb{E}[C]}{n + \kappa} \geq \frac{p}{n + \kappa} = \Omega(n^{-1}), \tag{87}$$

as desired. □

Proof of Corollary B.8. If $\mathrm{Var}(C) = O(\mathbb{E}[C])$, then Equation (85) gives $\mathrm{Var}(C) = \Theta(\mathbb{E}[C])$, and the result follows from Proposition B.7. □

Proof of Proposition B.9. Suppose $\mathbb{E}[C^2] = O(\mathbb{E}[D]^2)$. Since $\mathbb{E}[D] \geq \mathbb{E}[C] \geq p$ per Equation (21), we have $\mathbb{E}[C] \leq \mathbb{E}[D]^2/p$. Therefore

$$\mathrm{Var}(C) + \kappa \mathbb{E}[C] \leq \mathbb{E}[C^2] + \frac{\kappa}{p} \mathbb{E}[D]^2 = O(\mathbb{E}[D]^2). \tag{88}$$

Proposition B.7 yields

$$\mathrm{SNR}(Y_3) = \Omega\left(\frac{\mathbb{E}[C]}{\mathbb{E}[D]} \right). \tag{89}$$

To bound this ratio, define

$$r_v := \frac{p^{d_v}}{\mathbb{E}[C]}, \quad \sum_{v=1}^{n} r_v = 1. \tag{90}$$

It follows that

$$\frac{\mathbb{E}[D]}{\mathbb{E}[C]} = \sum_{v=1}^{n} r_v d_v = \frac{\sum_{v=1}^{n} r_v \log(1/r_v) - \log \mathbb{E}[C]}{\log(1/p)} \le \log_{1/p}\left(\frac{n}{\mathbb{E}[C]}\right) = O(\log n), \tag{91}$$

where the inequality uses the entropy bound $\sum_v r_v \log(1/r_v) \le \log n$. □

Proof of Proposition B.10. We first compute the expectations,

$$\mathbb{E}[Y_2] = \frac{\mu}{pn}\mathbb{E}[D], \quad \mathbb{E}[Y_3] = \frac{\mu}{pn}\mathbb{E}[C]. \tag{92}$$

Next, we consider the asymptotics of $\mathrm{Var}(Y_3)$. Recall Equation (81):

$$\mathrm{Var}(Y_3) = \frac{1}{p^2 n^2}\left[\mu^2 \mathrm{Var}(C) + (p\lambda - \mu^2)\mathbb{E}[C]\right]. \tag{93}$$

Since $\mathrm{Var}(C) \ge (1-p)\mathbb{E}[C]$, we have

$$\frac{\mu^2}{p^2 n^2}\mathrm{Var}(C) \le \mathrm{Var}(Y_3) \le \frac{1}{p^2 n^2}\left(\mu^2 + \frac{p\lambda - \mu^2}{1-p}\right)\mathrm{Var}(C), \tag{94}$$

and hence $\mathrm{Var}(Y_3) = \Theta(\mathrm{Var}(C)/n^2)$.

We next show that

$$\mathrm{Var}(Y_2) = \Omega\left(\frac{\mathrm{Var}(C)}{n}\right). \tag{95}$$

Consider first the $v = w$ summands in the expansion of $n^2\mathrm{Var}(Y_2)$ obtained from Equation (49). Writing $q_v := p^{d_v} = \mathbb{E}[I_v]$, the vth diagonal summand is

$$\frac{d_v q_v}{p}\left(\lambda - \frac{\mu^2 q_v}{p}\right) + (n - d_v)\eta q_v + \frac{\mu^2 d_v(d_v - 1)q_v}{p^2}(1 - q_v). \tag{96}$$

Because $q_v/p \le 1$, the first parenthesis is at least $\lambda - \mu^2 > 0$. Thus, if $d_v \ge n/2$, the first term is $\Omega(nq_v)$; otherwise the second term is $\Omega(nq_v)$. Since $q_v \le p < 1$, we have $\mathrm{Var}(I_v) = q_v(1-q_v) = \Theta(q_v)$, so the diagonal contribution is

$$\Omega\left(n\sum_v \mathrm{Var}(I_v)\right). \tag{97}$$

For fixed $v \ne w$, let

$$\gamma_{vw} := \mathrm{Cov}(I_v, I_w) = p^{d_{vw}} - p^{d_v + d_w} \ge 0. \tag{98}$$

The corresponding summand from Equation (49) can be rearranged as

$$\begin{aligned}\left[\eta(n - d_{vw}) + \frac{\mu^2}{p^2}d_{vw}(d_{vw} - 1)\right]\gamma_{vw} + \frac{\lambda d_{vw}}{p}p^{d_{vw}} \\ + \left[\eta(n - d_{vw}) + \frac{\mu^2}{p^2}\left(d_{vw}(d_{vw} - 1) - d_v d_w\right)\right]q_v q_w.\end{aligned} \tag{99}$$

The first term is $\Omega(n\gamma_{vw})$: use its $\eta(n - d_{vw})$ term when $d_{vw} \le n/2$ and its quadratic term when $d_{vw} > n/2$. The second term is nonnegative. The third term is also nonnegative because $P_v \ne P_w$ in a DAG, so their cardinalities are at most d_{vw} and $d_{vw} - 1$ in some order. Hence $d_v d_w \le d_{vw}(d_{vw} - 1)$. Thus,

$$n^2\mathrm{Var}(Y_2) = \Omega\left(n\sum_v \mathrm{Var}(I_v) + n\sum_{v \ne w}\mathrm{Cov}(I_v, I_w)\right) = \Omega(n\mathrm{Var}(C)), \tag{100}$$

which proves Equation (95).

Combining, we obtain

$$\frac{\mathrm{SNR}(Y_3)}{\mathrm{SNR}(Y_2)} = \frac{\mathbb{E}[C]}{\mathbb{E}[D]}\sqrt{\frac{\mathrm{Var}(Y_2)}{\mathrm{Var}(Y_3)}} = \Omega\left(\sqrt{n}\,\frac{\mathbb{E}[C]}{\mathbb{E}[D]}\right) \geq \Omega\left(\frac{\sqrt{n}}{\log n}\right), \tag{101}$$

where the final inequality follows from Equation (91). □

Example B.11. *Supposing that $\mathbb{E}[C^2] = O(\mathbb{E}[D]^2)$, the Hydrangea graph constructed below achieves the $\Theta(1/\log n)$ lower bound of Proposition B.9.*

Proof. For simplicity, first take $p = 1/2$. Fix a depth d, let $r = \lfloor \log_2 d \rfloor$, construct a stem $0 \to 1 \to \cdots \to r$, and attach a complete binary tree of depth d at r. The graph has $n = \Theta(2^d)$ points. See Figure 6.

For a tree point at depth k, $d_v = r + k$. Therefore

$$\mathbb{E}[C] = \sum_{j=1}^{r} 2^{-j} + \sum_{k=1}^{d} 2^k 2^{-(r+k)} = \Theta(1), \tag{102}$$

$$\mathbb{E}[D] = \sum_{j=1}^{r} j2^{-j} + \sum_{k=1}^{d} 2^k (r+k) 2^{-(r+k)} = \Theta(d) = \Theta(\log n). \tag{103}$$

To compute $\mathbb{E}[C^2]$, we split the sum into cases. First consider ordered pairs (v, w) of tree points at depths a, b (relative to r) whose lowest common prerequisite has tree depth $\ell < \min(a, b)$. There are $\Theta(2^{a+b-\ell})$ such pairs, and $d_{vw} = r + a + b - \ell$. Each fixed (a, b, ℓ) therefore contributes $\Theta(2^{-r})$. Summing,

$$\mathbb{E}[C^2] = \Theta\left(2^{-r} \sum_{a=1}^{d} \sum_{b=1}^{d} \min(a, b)\right) = \Theta(2^{-r} d^3) = \Theta(d^2). \tag{104}$$

Next, consider ordered pairs of tree points (v, w) where one point is a prerequisite of the other. For tree depths $1 \leq a \leq b \leq d$, there are $O(2^b)$ such pairs, each with $d_{vw} = r + b$. So their total contribution is

$$\sum_{1 \leq a \leq b \leq d} O\left(2^b 2^{-(r+b)}\right) = O(2^{-r} d^2) = O(d). \tag{105}$$

For a fixed stem point j and tree depth k, there are $O(2^k)$ ordered stem–tree pairs, each with $d_{vw} = r + k$. Hence these pairs contribute

$$\sum_{j=1}^{r} \sum_{k=1}^{d} O\left(2^k 2^{-(r+k)}\right) = O(rd2^{-r}) = O(r). \tag{106}$$

Finally, two stem points j, ℓ have $d_{j\ell} = \max(j, \ell)$, so

$$\sum_{j,\ell=1}^{r} 2^{-\max(j,\ell)} = \sum_{m=1}^{r} (2m-1)2^{-m} = O(1). \tag{107}$$

Because $r = \Theta(\log d)$, the total is $\mathbb{E}[C^2] = \Theta(d^2) = \Theta(\mathbb{E}[D]^2)$, satisfying the assumption.

Since $\mathbb{E}[C] = \Theta(1)$,

$$\mathrm{Var}(C) = \mathbb{E}[C^2] - \mathbb{E}[C]^2 = \Theta(d^2). \tag{108}$$

Applying Proposition B.7 yields

$$\mathrm{SNR}(Y_3) = \Theta\left(\frac{\mathbb{E}[C]}{\sqrt{\mathrm{Var}(C)}}\right) = \Theta(1/d) = \Theta(1/\log n). \tag{109}$$

For general fixed $p \in (0, 1)$, take $r = \lfloor \log_{1/p} d \rfloor$ and use a balanced rooted tree whose kth level contains $\lceil p^{-k} \rceil$ points. Here balanced means that descendants are distributed as evenly as possible, so each point at level i has $\Theta(p^{-(j-i)})$ descendants at level $j > i$. Then $n = \Theta(p^{-d})$, so $d = \Theta(\log n)$, and the same moment calculations give $\mathbb{E}[C] = \Theta(1)$, $\mathbb{E}[D] = \Theta(d)$, and $\mathbb{E}[C^2] = \Theta(d^2)$. □

C Experiment Details

In our experiments, we train the Qwen3-1.7B, Qwen3-4B, and Qwen3-4B-Instruct models on v5e-32 TPU pods. Each experiment ran for 12–24 hours, with longer runtimes for standard GRPO due to a large number of sampling iterations with no successful trajectories. We adopt a synchronous RL framework with GRPO. We train fully on-policy, i.e. we alternate between 1 sampling step and 1 training step. We do not use either entropy or KL regularization, and found empirically that tuning the clip high ratio was sufficient to stabilize entropy over training. Following Yu et al. (2025), we filter out prompts where all advantages are zero. The full hyperparameters are shown in Table 3. By default, we mask out tokens with an at least $2\times$ train-inference probability mismatch.

Table 3: Hyperparameters for RL training.

Hyperparameter	Value
Sampling temperature	1.0
Clip ratio (low/high)	0.20 / 0.28
Learning rate	3e-6
Training steps	150
Trajectories per prompt	16
Prompts per batch	8
Global batch size	128

Throughout our experiments we instantiate the judge model as gemini-2.5-flash-lite (Google DeepMind, 2025a), with prompts provided in Appendix E. Our reasoning points and reasoning graphs are generated by gemini-3-flash.

Table 4: Progressive point matching ablations on GSM-Infinite $n = 24$ at 400M training tokens.

Method	Success rate
Base model	0.057
Sparse outcome reward	0.091
PPM	**0.191**
PPM, no outcome reward	0.063
PPM, stdev normalization	0.071
PPM, trajectory-length chunks	0.148
PPM, no length filter	0.171

Throughout the paper we refer to token budgets as multiples of K $= 1024$, e.g. a 4K token budget is 4096 tokens.

For simplicity, we tuned the hyperparameters in Table 3 for standard GRPO, and we generally reuse the same configurations for all other methods. Thus, your favorite GRPO hyperparameters will generally work well with progressive point matching out of the box.

Progressive point matching differs from standard GRPO in several ways. We show ablations for each of these configuration options in Table 4 and describe them in more detail below:

- **Outcome reward.** Empirically, we found that training on the *sum* of the outcome reward and the method-specific reward significantly improves training speed, so that the policy can learn to correctly format answers and output them within the training token budget. For methods involving splitting trajectories into segments, we add the outcome reward to each segment's reward, then proceed with GRPO advantage calculations.
- **No standard deviation (stdev) normalization.** Typically, advantages are computed within each GRPO group by subtracting the group mean and dividing by the group standard deviation. We observe that with progressive point matching, it is beneficial to only perform the subtraction but not division.
- **Length filter.** In the loss, we mask out trajectories that are truncated by the training token budget. This alleviates length pressure.

When segmenting trajectories without natural turn-level boundaries, we use a simple baseline of **equal-length chunks**. As in Figure 7 (middle), we may have some empty chunks if the trajectory is considerably shorter than the training token budget, which are assigned progressive point matching rewards of zero for advantage calculations. Empirically, we find that this chunking scheme generally outperforms and leads to more stable training than trajectory-length chunks (Figure 7, bottom). Trajectory-length chunks may result in length pressures, as token-level advantages are applied to chunks of different lengths within the same GRPO group.

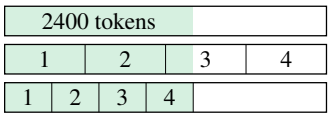


Figure 7: Progressive point matching (middle) follows the heuristic of equal-length chunks regardless of sampled trajectory length.

C.1 METHODS AND BASELINES

We next describe the methods and baselines to which we compared progressive point matching. As above, we train on the sum of outcome rewards and progressive point matching rewards for Verifree and process rewards. We always use gemini-2.5-flash-lite as the judge for process rewards and progressive point matching; we evaluate this in Appendix C.3.

- **Supervised fine-tuning (SFT).** Throughout the paper, we assume access to human- or oracle-written *solutions*, rather than full reasoning traces (after all, this is simply distillation from a stronger model). Thus, we directly fine-tune the model on sequences `[prompt, solution]`, using the *non-thinking* tokenizer mode in Qwen3 models.
- **Verifree (Zhou et al., 2025a).** The original method maximizes the likelihood of generating the dataset answer. Specifically, the model is prompted to generate responses formatted as

  `[preceding tokens] <answer> \boxed{answer here} </answer> <eos>`

  where "preceding tokens" include the prompt, thinking, and all output tokens except for the answer. Then, the reward is the likelihood of the `black` tokens, with `answer here` replaced by the dataset answer, conditioned on the `gray` tokens.

  However, we found empirically that this instantiation of Verifree is sensitive to the format of the answer tokens and underperforms outcome rewards, even when clipping token logprobs to $[-10, 0]$ for stability or summing Verifree rewards over multiple LLM-reformatted dataset answers. For instance, the answer $-2/3$ can be written as `-2/3`, `-\frac 23`, or `$\dfrac{-2}{3}$`, among a variety of answer formats. This instantiation of Verifree matched outcome rewards only in the setting where the dataset was filtered to only integer answers, thus reducing variance due to answer formatting. But in this setting these rewards are conceptually identical: if, for example, the conditioning tokens end with

  `The final answer is 60.\n\n<answer`

  then the likelihood of

  `> \boxed{dataset answer} </answer> <eos>`

  is effectively 1 if the dataset answer is 60 and 0 otherwise, matching outcome rewards.

  Instead, we observe empirically that this reward decreases in variance when averaging token logprobs over longer target sequences, simply by maximizing the likelihood of the reference solution:

  `[prompt] <think> [thinking] </think> [reference solution] <eos>`

  Specifically, we decompose the sampled sequence as $x_{0:T} = [x^{\text{prompt}}, x^{\text{think}}, x^{\text{out}}]$ and let $\tilde{x} = [x^{\text{prompt}}, x^{\text{think}}]$ denote the sequence obtained by discarding the model's final output. For an oracle solution x^{ref} with length L tokens, we maximize

  $$\mathbb{E}_{x \sim \pi_\theta}\left[\exp\left(\frac{1}{L}\sum_{t=1}^{L}\log \pi_\theta\left(x_t^{\text{ref}} \mid \tilde{x}, x_{<t}^{\text{ref}}\right)\right)\right] \tag{110}$$

- **On-policy self distillation (OPSD) (Zhao et al., 2026).** We disable group filtering and use a *fixed* teacher initialized as the base policy, which we found empirically led to more stable training than using a teacher parameterized as a moving average of the policy weights. We otherwise follow the OPSD official code. The objective is

  $$\mathbb{E}_{x^{\text{gen}} \sim \pi_\theta(\cdot \mid x^{\text{prompt}})}\left[\frac{1}{T}\sum_{t=1}^{T} D_{\text{JS}}\left(\pi_{\theta_0}\left(\cdot \mid x^{\text{prompt}}, x^{\text{ref}}, x_{<t}^{\text{gen}}\right) \,\middle\|\, \pi_\theta\left(\cdot \mid x^{\text{prompt}}, x_{<t}^{\text{gen}}\right)\right)\right] \tag{111}$$

  where D_{JS} denotes the Jensen–Shannon divergence.
- **Process rewards.** We follow the training configuration for progressive point matching. However, following prior work in process rewards (Miao et al., 2023; Khalifa et al., 2025), there are three key differences: (i) the judge prompt evaluates correctness of each step rather than progress to the goal, (ii) no shortcutting, and (iii) rewards are obtained from judging each trajectory segment independently $r_k := \phi'(c_k, x^{\text{ref}})$ rather than taking finite differences on the full prefix $\phi((c_1, \ldots, c_k), x^{\text{ref}})$. As we show in Proposition 3.1, (i) and (ii) result in a *biased* objective. Furthermore, in progressive point matching experiments, we have found that (iii) conditioning on full prefixes led to lower-variance segment-level rewards, as it is beneficial to have access to relevant context such as definitions and notation introduced early in the trajectory.

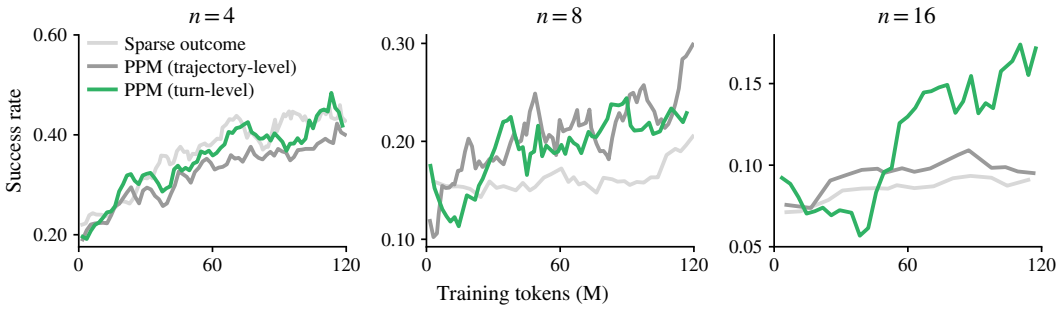


Figure 8: When training on the sequential Matrix Manipulation task, progressive point matching yields increasing performance improvements over sparse outcome rewards with the task horizon (n). However, agreeing with Theorem 4.1, this gap increases at a slower rate than in the Multi-Countdown case (Figure 4).

- **Privileged on-policy exploration (POPE) (Qu et al., 2026).** We adapt the prompt in the original paper. In each batch, half of the prompts contain a prefix of the reference solution, and half of the prompts do not.
- **Progressive point matching, exact order baseline.** We provide a reward only if the reasoning points are visited in the same order as in the reference trajectory. This baseline provides an approximation to imitation learning (without teacher forcing). However, as we show in Table 5, this baseline significantly reduces exploration compared to standard progressive point matching, perhaps because some reasoning points may be "out of support" for the policy.
- **Progressive point matching, no shortcutting baseline.** We use the reward in Equation (3) without shortcutting. As we argue in the proof to Proposition 3.1, shortcutting is required for the set of optimal policies to coincide with those from outcome rewards.

C.2 Environment-specific training details

We follow the defaults in Table 3 and as described above in Appendix C.1. The environments and datasets themselves are described in Appendix D.

- **Multi-Countdown.** We train Qwen3-1.7B, and each sub-problem is given a 512 token budget (for example, in the case $n = 8$, the policy generates $8 \cdot 512 = 4K$ tokens). We observed that when the agent failed to produce an answer within this budget, it would sometimes continue working on the unfinished sub-problem in subsequent turns. To prevent this effect, 50 tokens before the end of each turn's budget, we force the model to terminate its reasoning with `</think>` and begin its final response with `<answer>`, leaving sufficient budget to produce an answer.
- **Matrix Manipulation.** We train Qwen3-4B-Instruct-2507 with thinking mode disabled, and each manipulation operation is given a 512 token budget. We use gemini-2.5-flash-lite as the judge model with thinking disabled. We show horizon scaling results in Figure 8.
- **GSM-Infinite.** We train Qwen3-1.7B with thinking disabled at a training budget of 4K tokens.
- **Polaris.** We train Qwen3-4B at a training budget of 8K tokens, and we evaluate at 16K tokens. Our RL runs use a learning rate of 1e-6.
  - **SFT.** We train for 200 steps with a global batch size of 128 and learning rate of 5e-6.
- **POPE-hard.** We train Qwen3-4B-Instruct at a training budget of 4K or 8K tokens, and we evaluate 16K tokens.
  - **POPE.** We use 1-paragraph prefixes from oracle solutions. Note in Table 5 that the training reward is measured on the mixed dataset (both guided and non-guided), and thus cannot be directly compared to the other training rewards.
  - **Progressive point matching.** We remark that our method makes it possible to train on almost all trajectories (around 95% averaged over training) compared to almost none (around 0.2% averaged over training) with sparse outcome rewards.

Table 5: Qwen3-4B-Instruct trained on POPE-hard dataset and evaluated at larger test-time budget.

	Train		Test Time	
Method	4K Tokens pass@1	4K Tokens pass@8	16K Tokens pass@1	16K Tokens pass@8
Base model	0.000	0.000	0.004	0.023
GRPO	0.000	0.000	0.004	0.023
Verifree	0.008	0.008	0.006	0.040
POPE	0.013	0.078	0.012	0.063
Process rewards	0.0003	0.002	0.009	0.054
PPM, exact order	0.0009	0.014	0.0004	0.032
PPM, no shortcutting	0.001	0.019	0.021	0.093
PPM	0.005	0.019	0.035	**0.148**
	8K Tokens pass@1	8K Tokens pass@8	16K Tokens pass@1	16K Tokens pass@8
Base model	0.004	0.031	0.004	0.023
GRPO	0.002	0.012	0.004	0.023
PPM, exact order	0.005	0.027	0.006	0.023
PPM, no shortcutting	0.026	0.069	0.041	0.117
PPM	0.018	0.074	0.039	**0.125**

We use 16 prompts per batch and a KL penalty with the k3 estimator and 0.002 coefficient. We train only on the PPM reward, rather than summing the outcome reward and PPM reward.

We define segments by splitting the training token budget into 4 equal-length chunks. For example, Following Equation (3), we obtain segment-level rewards r_k.

Throughout the paper we refer to pass@k (or p@k for brevity) as the probability that at least one of k trajectories sampled i.i.d. from the policy for a given prompt attains an outcome-level reward of 1. For each prompt we draw $n \gg k$ trajectories, of which c receive a reward of 1, and use the standard unbiased estimator of this probability

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \tag{112}$$

which is the probability that a uniformly random size-k subset of the n drawn trajectories contains at least one success. We report the mean of $\widehat{\text{pass@}k}$ over the prompts in the evaluation set, which is likewise unbiased for the population quantity $\mathbb{E}_{x^{\text{prompt}} \sim \mathcal{D}}\left[\text{pass@}k(x^{\text{prompt}})\right]$.

C.3 Reasoning points, graphs, and judges

Progressive point matching relies on LLMs besides the policy to (i) generate reasoning points, (ii) generate reasoning graphs, and (iii) judging policy-generated responses. How sensitive is the method to the choice of LLM in each case? In this section, we show that any model with sufficient instruction following capabilities can serve as any of the three roles.

Before evaluating each of these three requirements, we remark that the use of external LLMs in the 4-chunk case is fairly wall clock efficient: reasoning points and graphs are constructed once along with the dataset, and judging largely relies on prefill (vs. autoregressive generation).

Evaluating reasoning points. We compare against ground truth reasoning points via GSM-Infinite. Qualitatively, an example is shown in Figure 9. Numerically, we find that various models achieve roughly identical performance in defining reasoning points (Table 6).

Evaluating reasoning graphs. In Section C.3, we evaluate what fraction of prerequisite pairs in ground truth graphs via GSM-Infinite that are also present in LLM-generated graphs. While there is degradation in reasoning graphs with horizon, this degradation occurs uniformly across models, and may occur if the models have not been trained sufficiently on XML-like data.

Ground-truth reasoning points

1. Defines the target quantity as a variable: let x be the public highschool in Clearwater Bay.
2. Derives that the average number of teachers per private middle school in Ruby Bay is 2.
3. Derives that the average number of teachers per public highschool in Ruby Bay is 8.
4. Derives that the average number of teachers per public highschool in Clearwater Bay is 8.
5. Derives that the number of private middle school in Clearwater Bay is 8.
6. Derives that the total number of schools in Clearwater Bay is $x + 8$.
7. Uses the given fact that the total number of schools in Clearwater Bay equals 11.
8. Solves the equation $x + 8 = 11$ to obtain $x = 3$.
9. Provides the final answer in the format $\boxed{3}$.

Qwen3-4B-Instruct reasoning points

1. The number of public highschool in Ruby Bay is correctly identified as 4.
2. The average number of teachers per private middle school in Ruby Bay is correctly identified as 2.
3. The average number of teachers per public highschool in Ruby Bay is correctly calculated as 8.
4. The average number of teachers per public highschool in Clearwater Bay is correctly calculated as 8.
5. The number of private middle school in Clearwater Bay is correctly calculated as 8.
6. The total number of schools in Clearwater Bay is correctly expressed as the sum of the number of public highschool in Clearwater Bay and the number of private middle school in Clearwater Bay, so it is correctly expressed as $x + 8$.
7. The total number of schools in Clearwater Bay is correctly given as 11, so the equation $x + 8 = 11$ is formed.
8. The number of public highschool in Clearwater Bay is correctly solved as 3.
9. The final answer is provided in the format $\boxed{3}$.

Figure 9: We evaluate LLM generated reasoning points against ground truth reasoning points from GSM-Infinite. Above is the only failure at $n = 8$ operations of 32 held-out examples.

Table 6: **Reasoning point evaluation.** Fraction of ground truth reasoning points that are also LLM-generated reasoning points, on 32 held-out prompts, judged by gemini-3-flash-preview.

Horizon n	Qwen3-4B-Instruct	Qwen3-32B	gemini-2.5-flash-lite	gemini-3-flash-preview
8	**0.9965**	**0.9965**	**0.9965**	0.9773
16	**0.9765**	0.9455	0.9555	0.9418
24	**0.9712**	0.9591	0.9551	0.9654

Table 7: Reasoning graph evaluation.

Horizon n	Qwen3-4B-Instruct	Qwen3-32B	gemini-2.5-flash-lite	gemini-3-flash-preview
8	**0.9965**	**0.9965**	**0.9965**	0.9773
16	**0.9765**	0.9455	0.9555	0.9418
24	**0.9712**	0.9591	0.9551	0.9654

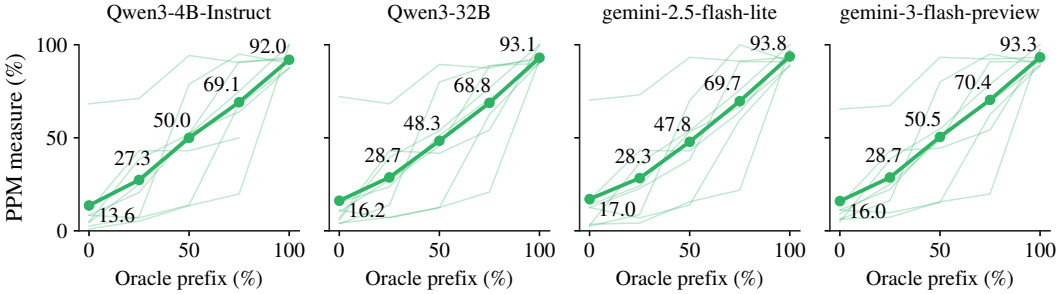


Figure 10: **PPM measure ϕ evaluation.** Across models, ϕ increases roughly linearly with the proportion of the reference trajectory we condition on in the prompt.

Evaluating the PPM measure ϕ. We follow the evaluation described in Section 4.3 on the POPE-hard dataset (Qu et al., 2026), which is very challenging for the base policy (Qwen3-1.7B). We sample 8 prompts, 5 reasoning states to condition on in the prompt (0%, 25%, 50%, 75%, 100% of the reference solution), and 8 trajectory continuations per setting. We sample 4K tokens, with `</think>` forcing at 1024 tokens remaining and `<answer>` forcing at 50 tokens remaining.

We find that ϕ increases roughly linearly with the length of the reference trajectory prefix we condition on, and that judge scores are consistent across models.

D ENVIRONMENTS

- **Multi-Countdown.** We use the standard 4-number Countdown task in a multi-turn setting. At each turn, the agent receives a new sub-problem and is given a budget of 512 tokens to solve it.
- **Matrix Manipulation.** We adapt the Matrix Manipulation environment from Reasoning Gym (Stojanovski et al., 2025). We resolved several bugs with the original environment, such as ambiguous or incorrectly labeled text descriptions of the matrix operations, and eliminating operations such as cropping that can artificially reduce the task horizon. The adapted environment is included as part of our code release. Each manipulation operation is presented as its own turn.
- **GSM-Infinite.** The standard GSM-Infinite environment (Zhou et al., 2025b) generates datasets containing problems with a mixture of task horizons. To isolate the effect of task complexity, we instead construct datasets at a fixed complexity n by rejection sampling from the original environment. For each prompt, we additionally fix the graph composition such that 40% of edges are task-relevant and the remaining 60% are distractors. **The datasets with 5000 prompts each are available here:**

  $n = 8$: omitted for anonymity
  $n = 16$: omitted for anonymity
  $n = 24$: omitted for anonymity

  The datasets are automatically generated with ground truth reasoning points, which can be derived from the underlying graphs.
- **Polaris.** We begin by obtaining a filtered dataset (omitted for anonymity) of 3903 math problems, which were originally sourced from `POLARIS-Project/Polaris-Dataset-53K` and `nvidia/AceReason-Math`. Here `mean_reward` denotes average success rates of Qwen3-4B-Instruct-2507 when evaluated at length 32K tokens. We then perform some filtering to remove problems that cannot be easily verified: these include proof-like or multiple-choice problems, problems that refer to missing figures, or malformed answers, leaving 3509 usable problems.

  We then generated solutions, reasoning points ("rubrics"), and reasoning graphs ("obsoletion graphs") via gemini-3-flash-preview. Filtering to usable rows (i.e., by generating solutions that ag reed with dataset answers and correctly-formatted reasoning points and graph) yielded 2287 rows.

  The resulting dataset is available here: omitted for anonymity

  We train on the `train_boxed` split, which is identical to `train` except for answer formatting.
- **POPE-hard.** We begin by filtering `CMU-AIRe/POPE-source-128x32k` (Qu et al., 2026), a dataset of 2515 challenging math problems. We sample 16 trajectories per prompt with Qwen3-4B with an 8K token budget, while providing different numbers of paragraphs (0 paragraphs and standard generation, 10 paragraphs with a prompt to continue reasoning) from prefixes of the gemini-2.5-pro solutions.

  The resulting dataset is available here: `https://huggingface.co/datasets/kvfrans/POPE-HARD-w-oracle-solution-gemini-rubric`.

  The `train` split consists of all 601 prompts with zero pass rate over 16 trajectories with no reference solution prefix. The `0p10p` split consists of the 382 `train` prompts with nonzero pass rate over 16 trajectories when conditioning on 10 paragraphs. Thus, `0p10p` consists of the set of prompts that are amenable to methods such as POPE (Qu et al., 2026). The reasoning points and reasoning graphs are generated with gemini-3-flash-preview.

  Throughout the paper, all references to the "POPE-hard dataset" correspond to the `0p10p` split.

Example problems and solutions for the three environments are shown below.

Multi-Countdown

Prompt

```
<|im_start|>system
You are a helpful assistant. You first think about the reasoning process in the mind,
and then provide the user with the answer. You will be given a set of base numbers along
 with a target number. Your job is to create an equation that equals the target using
basic arithmetic operations (+, -, *, /). Each provided number must be used exactly once.
 Show your work in <think> </think> tags. Think for only ten sentences, then return the
final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>. You
 will solve countdown problems multiple times, and each problem will be a different set
of numbers and target.<|im_end|>
<|im_start|>user
[Problem 1/2] Base Numbers: [20, 32, 59, 75]. Target: 82.<|im_end|>
<|im_start|>assistant
<think>
...
<|im_end|>
<|im_start|>user
[Problem 2/2] Base Numbers: [72, 20, 6, 50]. Target: 208.<|im_end|>
<|im_start|>assistant
<think>
...
<|im_end|>
```

Reference solution

```
Turn 1: 59 - 20 - 32 + 75
Turn 2: 50*6 - 20 - 72
```

Manipulate Matrix

Prompt

```
<|im_start|>system
You will be given a matrix and a sequence of operations. After each operation, output
the resulting matrix in <answer> </answer> tags, one row per line, values separated by
spaces. Example:
<answer>
1 2
4 5
</answer><|im_end|>
<|im_start|>user
Initial matrix:
6 0 1 8
1 5 9 0
8 3 0 1
6 6 1 3

Apply operation 1/3: Mirror the matrix along the counterdiagonal.
Output the resulting matrix in <answer> </answer> tags.<|im_end|>
<|im_start|>assistant
<think>
...
<|im_end|>
<|im_start|>user
Apply operation 2/3: Rotate the matrix 90 degrees counterclockwise.
Output the resulting matrix in <answer> </answer> tags.<|im_end|>
<|im_start|>assistant
<think>
...
<|im_end|>
<|im_start|>user
Apply operation 3/3: Remove every 4-th column (1-indexed).
Output the resulting matrix in <answer> </answer> tags.<|im_end|>
<|im_start|>assistant
<think>
...
<|im_end|>
```

Reference solution

```
Turn 1
```

```
<answer>
3 1 0 8
1 0 9 1
6 3 5 0
6 8 1 6
</answer>

Turn 2
<answer>
6 6 1 3
8 3 0 1
1 5 9 0
6 0 1 8
</answer>

Turn 3
<answer>
6 6 1
8 3 0
1 5 9
6 0 1
</answer>
```

GSM-Infinite

Prompt

```
<|im_start|>system
Answer the question below.

Notes:
- The total number of adult animals in a location is the sum of all adult animals of
every type ever mentioned for that location, EXCLUDING newborn children.
- If a type of animal is never mentioned for a location, assume its count there is 0.
- Average newborn children per adult may differ across locations.
- The total newborn children in a location is the sum, over each type of adult animal
mentioned for that location, of (count of that adult animal in the location) times (
average newborn children per that adult animal in the location).

**Important:** The final answer should be written as \boxed{...}.
<|im_end|>
<|im_start|>user
The average number of newborn children per adult owl in Pine Ridge equals the average
number of newborn children per adult eagle in Pine Ridge. The average number of newborn
children per adult kangaroo in Glintshade Grove equals the sum of the average number of
newborn children per adult owl in Cedar Valley and the number of adult owl in Cedar
Valley. The number of adult eagle in Pine Ridge equals 4. The average number of newborn
children per adult koala in Glintshade Grove equals the average number of newborn
children per adult owl in Cedar Valley. The number of adult owl in Cedar Valley equals 3
 times the average number of newborn children per adult eagle in Beverly Forest. The
number of adult owl in Pine Ridge equals 1. The average number of newborn children per
adult eagle in Pine Ridge equals 3 times the sum of the average number of newborn
children per adult eagle in Beverly Forest and the number of adult owl in Cedar Valley.
The number of adult koala in Glintshade Grove equals the sum of the number of adult
eagle in Pine Ridge and the average number of newborn children per adult owl in Cedar
Valley. The average number of newborn children per adult owl in Cedar Valley equals the
sum of the number of adult owl in Cedar Valley and the number of adult owl in Pine Ridge.
 The number of adult eagle in Beverly Forest equals 4 plus the number of adult eagle in
Pine Ridge. The average number of newborn children per adult eagle in Beverly Forest
equals 1. The number of adult kangaroo in Glintshade Grove equals the average number of
newborn children per adult owl in Cedar Valley. Question: What is the total number of
adult animals in Beverly Forest?
<|im_end|>
<|im_start|>assistant
<think>
```

Reference solution

```
Define adult eagle in Pine Ridge as s; so s = 4. Define adult eagle in Beverly Forest as
 H; Q = s = 4; so H = 4 + Q = 4 + 4 = 8. Define total number of adult animals in Beverly
 Forest as Y; so Y = H = 8.
\\boxed{8}
```

Rubric

```
1. The number of adult eagle in Pine Ridge is correctly identified as 4.
2. The number of adult eagle in Beverly Forest is correctly calculated as 8.
```

```
3. The total number of adult animals in Beverly Forest is correctly identified as 8.
4. Provides the final answer in the format \boxed{8}.
```

Reasoning Graph

```
[([2], [1]), ([3], [2]), ([4], [3])]
```

E PROMPTS

Reasoning Points Generation Prompt

```
You are given a problem statement and a reference solution that should be treated as
correct. Generate a rubric for evaluating other solutions against the reasoning actually
 present in the reference solution.

The rubric must be faithful to the reference solution. Do not add lemmas, cases, rigor,
bounds, or proof obligations that are not explicitly present or clearly implied. If the
reference solution uses an approximation or informal argument, describe that argument as
 it appears rather than replacing it with a stronger rigorous version. Rubric items
should test whether another solution follows the same essential reasoning path as the
reference solution.

Each rubric item must:
- be a single sentence;
- be evaluable as YES or NO without extra context;
- state one concrete mathematical requirement, such as deriving a claim, comparing
expressions, handling a case, or reaching a conclusion;
- represent one logically distinct requirement, with no duplicates;
- be explicitly stated or clearly implied by the reference solution.

The final rubric item must state the final conclusion or answer.

Do not include:
- definitions, formulas, or background facts unless essential to the solution's
reasoning;
- vague criteria such as "correctly analyzes," "uses appropriate reasoning," or "
understands the problem";
- presentation or style criteria;
- point values, grading guidance, solution text, hints, or references to these
instructions;
- stronger arguments than the reference solution gives;
- requirements needed only for a fully rigorous proof but absent from the reference
solution.

## Output format

Short breakdown of the reference solution. (4 sentences max)

Estimated number of rubric items for each key step. (2 sentences max)

<rubric>
1. Rubric item 1
2. Rubric item 2
and so on... (6-12 items)
</rubric>

## Example problem statement
Determine all $\alpha > 1$ for which $\sum_{k=1}^{n} \left\lfloor k \sqrt{\alpha} \right\
rfloor > \left\lfloor \frac{n^2}{\sqrt{\alpha}} \right\rfloor$, where $\left\lfloor . \
right\rfloor$ denotes the integer part.

## Example reference solution
We are tasked with determining all $\alpha > 1$ for which the inequality

$$
\sum_{k=1}^n \left\lfloor k \sqrt{\alpha} \right\rfloor > \left\lfloor \frac{n^2}{\sqrt{\
alpha}} \right\rfloor
$$

holds for all positive integers $n$.

---

### Step 1: Approximate the Sum and the Floor Terms
```

```
Let us approximate the left-hand side:

$$
\sum_{k=1}^n \left\lfloor k \sqrt{\alpha} \right\rfloor \approx \sum_{k=1}^n k \sqrt{\
alpha} - \sum_{k=1}^n \left\{k \sqrt{\alpha} \right\}
$$

The sum of the first $n$ integers is $\frac{n(n+1)}{2}$, so:

$$
\sum_{k=1}^n \left\lfloor k \sqrt{\alpha} \right\rfloor \approx \sqrt{\alpha} \cdot \
frac{n(n+1)}{2}
$$

For the right-hand side:

$$
\left\lfloor \frac{n^2}{\sqrt{\alpha}} \right\rfloor \approx \frac{n^2}{\sqrt{\alpha}} -
 \left\{ \frac{n^2}{\sqrt{\alpha}} \right\}
$$

---

### Step 2: Compare the Leading Terms

We compare the leading terms:

$$
\sqrt{\alpha} \cdot \frac{n(n+1)}{2} \approx \frac{\sqrt{\alpha}}{2} n^2
$$

$$
\frac{n^2}{\sqrt{\alpha}}
$$

For the inequality to hold, we must have:

$$
\frac{\sqrt{\alpha}}{2} n^2 > \frac{n^2}{\sqrt{\alpha}} \quad \Rightarrow \quad \frac{\
sqrt{\alpha}}{2} > \frac{1}{\sqrt{\alpha}} \quad \Rightarrow \quad \alpha > 2
$$

---

### Step 3: Consider $\alpha = 2$

We analyze $\alpha = 2$:

- The leading terms are equal: $\frac{\sqrt{2}}{2} n^2 = \frac{n^2}{\sqrt{2}}$
- However, the fractional parts $\{k \sqrt{2}\}$ are distributed uniformly, and the
difference between the sums is positive for large $n$, ensuring the inequality holds.

---

### Step 4: Conclusion

The inequality holds for all $\alpha \geq 2$.

<answer>[2, \infty)</answer>

## Example rubric
<rubric>
1. The solution states that the inequality is required to hold for all positive integers
 n.
2. The solution expresses or approximates the sum \sum_{k=1}^n \lfloor k\sqrt{\alpha}\
rfloor using \sum_{k=1}^n k\sqrt{\alpha} and fractional parts.
3. The solution identifies \sum_{k=1}^n k = n(n+1)/2.
4. The solution identifies the leading-order term of \sum_{k=1}^n \lfloor k\sqrt{\alpha}\
rfloor as \frac{\sqrt{\alpha}}2 n^2.
5. The solution identifies the leading-order term of \left\lfloor n^2/\sqrt{\alpha}\
right\rfloor as n^2/\sqrt{\alpha}.
6. The solution compares the leading-order terms to derive \frac{\sqrt{\alpha}}2 > \
frac1{\sqrt{\alpha}}.
7. The solution concludes from this comparison that \alpha > 2 is required away from the
 boundary.
8. The solution separately considers the boundary case \alpha = 2.
9. The solution argues that the boundary case \alpha = 2 should be included, for example
 using fractional-part behavior.
10. The solution concludes that the desired set is \alpha \in [2,\infty).
```

```
</rubric>

## Problem statement
{problem}

## Reference solution
{solution}

## Instructions
Provide the solution breakdown, rubric item allocation, and rubric as described in the
system prompt.
```

Reasoning Graph Prompt

```
You are given a problem statement and a grading rubric (a numbered list of criteria).
Construct an obsoletion graph: a set of rules stating that satisfying some rubric items
makes other items redundant to grade, because the satisfied items already imply or
subsume them.

For the obsoletion graph, a later item may obsolete earlier steps when it clearly
depends on them or substantially subsumes their mathematical content. Allow natural
local chains, such as a derived inequality obsoleting the intermediate comparison that
directly produced it, or a case resolution obsoleting merely considering that case. Do
not let a final answer obsolete specific derivations, constructions, case analyses, or
impossibility arguments. Prefer a reasonably sparse graph that reduces redundant grading
 without skipping substantive reasoning.

Each rule has the form <rule if_all="i,j" makes_obsolete="a,b,c" />, meaning that if
items i and j are all satisfied, then items a, b, and c need not be graded separately.
Only reference item numbers that exist in the rubric.

## Output format

Brief explanation of the dependency policy used in the obsoletion graph. (4 sentences
max)

<makes_obsolete_graph>
<rule if_all="..." makes_obsolete="..." />
<rule if_all="..." makes_obsolete="..." />
and so on...
</makes_obsolete_graph>

## Example problem statement
Determine all $\alpha > 1$ for which $\sum_{k=1}^{n} \left\lfloor k \sqrt{\alpha} \right\
rfloor > \left\lfloor \frac{n^2}{\sqrt{\alpha}} \right\rfloor$, where $\left\lfloor . \
right\rfloor$ denotes the integer part.

## Example rubric
<rubric>
1. The solution states that the inequality is required to hold for all positive integers
 n.
2. The solution expresses or approximates the sum \sum_{k=1}^n \lfloor k\sqrt{\alpha}\
rfloor using \sum_{k=1}^n k\sqrt{\alpha} and fractional parts.
3. The solution identifies \sum_{k=1}^n k = n(n+1)/2.
4. The solution identifies the leading-order term of \sum_{k=1}^n \lfloor k\sqrt{\alpha}\
rfloor as \frac{\sqrt{\alpha}}2 n^2.
5. The solution identifies the leading-order term of \left\lfloor n^2/\sqrt{\alpha}\
right\rfloor as n^2/\sqrt{\alpha}.
6. The solution compares the leading-order terms to derive \frac{\sqrt{\alpha}}2 > \
frac1{\sqrt{\alpha}}.
7. The solution concludes from this comparison that \alpha > 2 is required away from the
 boundary.
8. The solution separately considers the boundary case \alpha = 2.
9. The solution argues that the boundary case \alpha = 2 should be included, for example
 using fractional-part behavior.
10. The solution concludes that the desired set is \alpha \in [2,\infty).
</rubric>

## Example obsoletion graph
Item 4 obsoletes items 2 and 3 because identifying the leading term of the left-hand
side usually already uses the approximation of the floor sum and the formula for $\sum
k$. Item 6 obsoletes items 4 and 5 because comparing the leading terms via the displayed
 inequality presupposes those leading terms. Item 7 obsoletes item 6 because concluding
$\alpha > 2$ from the leading-term comparison normally subsumes the intermediate
inequality comparison. Item 9 obsoletes item 8 because arguing that $\alpha = 2$ is
included necessarily means the boundary case was considered. Item 10 obsoletes items 7
```

```
and 9 because the final interval $[2,\infty)$ combines the threshold conclusion with
inclusion of the boundary case.

<makes_obsolete_graph>
<rule if_all="4" makes_obsolete="2,3" />
<rule if_all="6" makes_obsolete="4,5" />
<rule if_all="7" makes_obsolete="6" />
<rule if_all="9" makes_obsolete="8" />
<rule if_all="10" makes_obsolete="7,9" />
</makes_obsolete_graph>

## Problem statement
{problem}

## Rubric
{rubric}

## Instructions
Provide the dependency policy explanation and the obsoletion graph as described in the
system prompt.
```

Naive Rubric Grading Prompt

```
You are given:
1. A mathematical problem.
2. A grading rubric consisting of numbered criteria.
3. A student solution to the problem.

Your task is to grade the student solution against the rubric.
For each statement, output only "YES" or "NO" based on whether the solution satisfies
the criterion. Format your response as a numbered list, e.g. if there are n rubric items,
 your response should follow the format:
1. YES
2. NO
...
n. YES

Problem statement:
{problem}

--------------------

Rubric:
{rubric}

--------------------

Student solution:
{student solution}

--------------------
Based on this information, please judge the partial rollout against the rubric with
satisfiability as described in the system prompt.
```

Progressive Point Matching Prompt

```
You are given:
1. A mathematical problem.
2. A grading rubric consisting of numbered criteria.
3. A partial rollout (reasoning trace + solution) to solving the problem.

Grading rules:
- For each rubric criterion i, output YES if *any* part of the rollout exactly satisfies
 criterion i.
- Answer NO if you have checked the entire rollout and are certain that the criterion is
 nowhere satisfied.
- Always answer NO if the rubric item describes a feature not present in the given
rollout. For example, if the rubric item asks "Do the <answer> </answer> tags contain
the number 3", but there are no answer tags at all, answer NO.
- Judge each criterion independently. For example, if step 2 depends on step 1, grade
step 2 based solely on whether the rollout's claimed step-2 result matches the rubric,
regardless of whether step 1 was correct.
- Judge whether the rubric criterion itself is correct in the rollout, not whether the
reasoning that led to it was sound.
- Do not use the rubric or problem text as evidence; only quote from the rollout.
```

```
You MUST respond in exactly two sections, in this order:

Section 1 -- Reasoning:
1. Criterion: <verbatim criterion including whitespace>.
   Evidence: <verbatim quote from rollout including whitespace> or "not present in
rollout".
   Reason: <explain how evidence relates to criterion, maximum 15 words>.
   Verdict: YES/NO.
2. Criterion: ...
   Evidence: ...
   Reason: ...
   Verdict: ...
... and so on

Section 2 -- Grades (one line per criterion):
1. YES
2. NO
... and so on

Do NOT output Section 2 before completing Section 1. Do NOT skip Section 1.

# Problem statement

{problem}

# Partial rollout

<rollout>
{student solution}
</rollout>

# Rubric

{rubric}

# Instructions

Based on this information, please judge the partial rollout against the rubric with
satisfiability as described in the system prompt.
Only the text inside <rollout> ... </rollout> counts as evidence. Ignore the problem
statement and rubric when judging satisfaction.
Include the section 1 and section 2 headers.
```

Process Reward Model Prompt

```
**System prompt**

You will be given a math problem along with a solution. They will be formatted as
follows:

# Problem
...(math problem)...

# Solution
<part_1>
...(part 1 of solution)...
</part_1>
...
<part_n>
...(part n of solution)...
</part_n>

Your task is to review each part of the solution in sequence, analyzing, verifying, and
critiquing the reasoning in detail. Provide the analyses and the conclusion in the
following format:

<analysis_1>
...(analysis of part 1)...
</analysis_1>
...
<analysis_n>
...(analysis of part n)...
</analysis_n>
<conclusion>
Correct/Incorrect
</conclusion>
```

```
Grading rules:
- When you analyze each part, use proper verification, recalculation, or reflection to
indicate whether it is logically and mathematically valid. Elaborate on the analysis
process carefully.
- If an error is detected in any part, describe the nature and cause of the error in
detail, and suggest how to correct the error or the correct approach.
- Once a part is found to contain any error, stop further analysis of subsequent parts (
as they may depend on the identified error) and directly provide the conclusion of "
Incorrect."
- For instance, given a solution of five parts, if an error is found in the third part,
reply with <analysis_1>, <analysis_2>, and <analysis_3> (the latter containing the
detailed critique and correction guideline), then <conclusion>Incorrect</conclusion>.
Skip the analyses of parts 4 and 5.
- Respond with your analyses and conclusion directly.

# Problem
{problem}

# Solution
{tagged chunk}

# Task

Provide the analysis/verification/critique/conclusion in the format described above.
```

POPE Prefix Prompt

```
You are given a problem and a {partial solution/reference solution}. Your task is to
carefully study the {partial solution/reference solution} and use it as guidance to
derive a complete and correct solution.

Use the information from the {partial solution/reference solution} silently. Do not copy,
 rephrase, or explicitly mention anything from it.

**Important**: Show your reasoning step-by-step, and present the final answer using
LaTeX within <answer>...</answer> tags.

# Problem
{problem}

# {Partial Solution/Reference Solution}
{partial}

Derive the full solution to the problem, re-deriving each step yourself. Make sure your
final answer is written as: <answer>your answer here</answer>
```