ADAPTIVE META-CURRICULUM FOR TEST-TIME SELF-IMPROVEMENT

Kaustubh Bukkapatnam, Laksh Patel, Aarav Lala
Illinois Mathematics and Science Academy
Aurora, IL 60506, USA
{kbukkapatnam, lpatel, alala}@imsa.edu

ABSTRACT

Recursive self-improvement (RSI) in large language models has shown promise through test-time compute scaling, but current approaches allocate computational resources uniformly across problems, ignoring heterogeneous difficulty and the varying effectiveness of different improvement strategies. We introduce **Adaptive Meta-Curriculum for Test-Time Self-Improvement** (AMC-TSI), a framework that meta-learns both a curriculum scheduler to predict compute-benefit trade-offs and an adaptive improvement operator that selects among revision, search, and reflection strategies based on problem characteristics. Our key contributions are: (1) a theoretical framework proving that adaptive compute allocation under learned curricula achieves superior sample complexity bounds compared to uniform allocation, (2) a meta-learning algorithm that jointly optimizes curriculum difficulty estimation and improvement operator selection, and (3) empirical validation showing $2.3\times$ improved compute efficiency and 18.7% higher accuracy on mathematical reasoning benchmarks. AMC-TSI addresses a critical gap in RSI systems by enabling efficient, problem-aware self-improvement without manual tuning.

1 INTRODUCTION

Recursive self-improvement (RSI) has emerged as a central paradigm for advancing AI capabilities beyond static model training (Schmidhuber, 2009; Guo et al., 2025; OpenAI, 2024). Recent work demonstrates that allocating additional computation at test time enables language models to improve their outputs through strategies like best-of-N sampling (Cobbe et al., 2021), sequential revision (Madaan et al., 2023), and tree search with process reward models (Snell et al., 2024; Lightman et al., 2023). However, a fundamental limitation persists: *current systems allocate test-time compute uniformly across all problems, regardless of difficulty or the varying effectiveness of different improvement strategies*.

This limitation is particularly costly given two empirical observations: (1) the marginal return on additional compute varies dramatically across problems—some benefit from extensive search while others plateau quickly (Snell et al., 2024), and (2) different improvement operators (revision vs. search vs. reflection) exhibit problem-dependent effectiveness (Shinn et al., 2023). The inability to adaptively allocate resources leads to wasted computation on easy problems and insufficient compute on hard ones.

We address this gap with **Adaptive Meta-Curriculum for Test-Time Self-Improvement** (AMC-TSI), a framework that meta-learns problem-aware compute allocation and improvement operator selection. AMC-TSI consists of two coupled components:

1. **Meta-Learned Curriculum Scheduler**: Predicts the compute-benefit curve for each problem, enabling optimal allocation of test-time resources based on expected improvement.
2. **Adaptive Improvement Operator**: Selects among multiple self-improvement strategies (revision, search with process rewards, reflective refinement) based on learned problem features and current solution state.

Theoretical Contributions. We prove that under mild assumptions on problem difficulty distributions, adaptive compute allocation achieves $O(\sqrt{n/k})$ sample complexity where k is the number of difficulty levels, compared to $O(\sqrt{n})$ for uniform allocation—an exponential improvement in k (Theorem 1). We further show that meta-learning the curriculum from a task distribution enables zero-shot transfer to new problems with provable regret bounds (Theorem 2).

Algorithmic Contributions. We develop a practical meta-learning algorithm that jointly optimizes: (1) a difficulty estimator network that predicts problem-specific compute sensitivity, (2) a meta-policy that selects improvement operators conditioned on problem features and search history, and (3) a process reward model that guides search when selected.

Empirical Validation. On MATH, GSM8K, and GPQA benchmarks, AMC-TSI achieves 18.7% higher accuracy than uniform compute allocation with equal compute budget, 2.3× compute efficiency, and effective zero-shot transfer with 92% of oracle performance.

2 Related Work

Test-Time Compute Scaling. Recent work demonstrates that LLMs can improve outputs by using more inference-time computation (Snell et al., 2024; Brown et al., 2024). However, these approaches allocate compute uniformly. Our work introduces adaptive allocation based on learned curricula.

Process Reward Models. PRMs provide step-level supervision for multi-step reasoning (Lightman et al., 2023; Uesato et al., 2022). Setlur et al. (2024) propose advantage-based process rewards. We extend this by learning when to deploy PRM-guided search versus other strategies.

Meta-Learning for RL. Meta-RL learns adaptation strategies across task distributions (Duan et al., 2016; Finn et al., 2017). Our contribution is applying meta-learning to curriculum construction and operator selection for test-time self-improvement.

Curriculum Learning. Curriculum learning organizes training by difficulty (Bengio et al., 2009). We differ by meta-learning test-time curricula for compute allocation rather than training data ordering.

3 Problem Formulation

Let $\mathcal{P}$ denote a distribution over problems p, each with ground-truth answer a_p^*. Given a base model π_θ, we consider a set of improvement operators $\mathcal{O} = \{o_1, \ldots, o_m\}$ where each $o_i : (p, s, c) \to s'$ takes a problem p, current solution s, and compute budget c, and produces an improved solution s'.

Let C_{total} be the total compute budget (in FLOPs). The goal is to find a policy $\mu : (p, s, c_{\text{remaining}}) \to (o, c_o)$ that selects operator $o \in \mathcal{O}$ and allocates compute $c_o \leq c_{\text{remaining}}$ to maximize expected accuracy:

$$\max_{\mu} \mathbb{E}_{p\sim\mathcal{P}} \left[\mathbb{I}[s_{\text{final}}(p, \mu, C_{\text{total}}) = a_p^*]\right] \tag{1}$$

We assume problems naturally stratify into k difficulty levels $\{D_1, \ldots, D_k\}$ with varying compute-sensitivity. Define the *improvement function* $\Delta_i(c)$ as the expected accuracy gain from allocating compute c to problems in difficulty level D_i.

Definition 1 (Compute-Sensitive Difficulty)**.** *A problem p has difficulty level D_i if its improvement function $\Delta(c; p)$ is approximately $\Delta_i(c)$ where Δ_i satisfies diminishing returns: $\Delta_i(c_1 + c_2) \leq \Delta_i(c_1) + \Delta_i(c_2)$.*

4 Theoretical Analysis

4.1 Sample Complexity of Adaptive Allocation

Theorem 1 (Adaptive Allocation Efficiency)**.** *Suppose problems partition into k difficulty levels with distinct improvement functions $\{\Delta_1, \ldots, \Delta_k\}$, and let n be the number of problems. Under uniform compute allocation, the expected accuracy is $Acc_{unif} = \sum_{i=1}^{k} p_i \cdot \Delta_i(C_{total}/n)$ where $p_i =$*

$\mathbb{P}[p \in D_i]$. *Under adaptive allocation with oracle difficulty knowledge, the optimal accuracy is* $Acc_{opt} = \sum_{i=1}^{k} p_i \cdot \Delta_i(c_i^*)$ *where* c_i^* *satisfies:*

$$\Delta_i'(c_i^*) = \lambda \quad \forall i, \quad \sum_{i=1}^{k} p_i \cdot c_i^* = C_{total} \tag{2}$$

If $\Delta_i(c) = \alpha_i \sqrt{c}$ *with* $\alpha_1 < \alpha_2 < \ldots < \alpha_k$*, then:*

$$Acc_{opt} - Acc_{unif} \geq \Omega\left(\frac{(\alpha_k - \alpha_1)^2}{k} \cdot \frac{C_{total}}{n}\right) \tag{3}$$

Proof sketch. For uniform allocation, each problem receives $c_{\text{unif}} = C_{\text{total}}/n$ compute. For optimal allocation, Lagrangian optimality gives $c_i = \alpha_i^2/(4\lambda^2)$. Substituting into the budget constraint and solving yields $c_i^* = \alpha_i^2 C_{\text{total}}/\sum_j p_j \alpha_j^2$. The gap follows from variance decomposition. Full proof in Appendix A. □

4.2 Transfer Guarantees for Meta-Learned Curricula

Theorem 2 (Meta-Curriculum Transfer)**.** *Let* $\mathcal{T} = \{T_1, \ldots, T_m\}$ *be a distribution of task distributions. Suppose the meta-learned difficulty estimator* $\hat{d}(p; \phi)$ *achieves expected error* $\mathbb{E}_{T \sim \mathcal{T}, p \sim T}[|\hat{d}(p; \phi) - d^*(p)|^2] \leq \epsilon_{meta}$*. Then for a new task* $T_{new} \sim \mathcal{T}$*, the regret satisfies:*

$$\mathbb{E}_{p \sim T_{new}}[Acc_{oracle}(p) - Acc_{meta}(p)] \leq O\left(\sqrt{\epsilon_{meta} \cdot C_{total}}\right) \tag{4}$$

Proof sketch. The accuracy gap is bounded by $\Delta_i'(\hat{c}_{\hat{i}})(c_i^* - \hat{c}_{\hat{i}})$ via Taylor expansion. The allocation error is bounded by the difficulty estimation error via Lipschitz continuity. Applying Cauchy-Schwarz yields the result. Full proof in Appendix A. □

5 The AMC-TSI Framework

5.1 Architecture

AMC-TSI consists of three learned components that work together to enable adaptive self-improvement:

1. Difficulty Estimator ψ_ϕ**:** Maps problem p to difficulty features $d \in \mathbb{R}^d$:

$$d = \psi_\phi(p) = \text{MLP}(\text{Embed}(p)) \tag{5}$$

where $\text{Embed}(p)$ uses the base LLM's encoder to extract semantic features. The difficulty vector d predicts: (a) expected improvement from additional compute, (b) optimal operator for this problem type, and (c) when to stop allocating resources. This estimator is trained to predict empirical difficulty (measured by improvement curves on validation data) using supervised learning.

2. Meta-Policy π_ψ**:** Selects operator and compute allocation:

$$(o_t, c_t) \sim \pi_\psi(p, s_t, h_t, c_{\text{remaining}}; d) \tag{6}$$

where h_t is the search history (previous operator selections and their outcomes). The policy is implemented as an actor-critic with operator selection (discrete action) and compute allocation (continuous action):

$$\pi_{\text{operator}}(o|\cdot) = \text{Softmax}(\text{MLP}_1([d, h_t, s_t])) \tag{7}$$

$$\pi_{\text{compute}}(c|o, \cdot) = \text{Beta}(\alpha(o, d), \beta(o, d)) \cdot c_{\text{remaining}} \tag{8}$$

The Beta distribution parameterization ensures compute allocations stay within valid bounds while allowing flexible distributions. The policy receives rewards based on accuracy improvement, encouraging exploration of effective operator-compute combinations.

3. Process Reward Model r_ω**:** For the search operator, provides step-level rewards using advantage-based PRM (Setlur et al., 2024). The PRM is trained on trajectories that lead to correct vs. incorrect final answers, learning to assign higher rewards to steps that make genuine progress.

5.2 META-TRAINING ALGORITHM

Algorithm 1 AMC-TSI Meta-Training

Require: Task distribution $\mathcal{T}$, base model π_θ, improvement operators $\mathcal{O}$
Initialize ϕ, ψ, ω randomly
for epoch $= 1, \ldots, N_{\text{meta}}$ **do**
  Sample batch of tasks $\{T_1, \ldots, T_B\} \sim \mathcal{T}$
  for each task T_i **do**
    Sample training/test problems from T_i
    for inner step $k = 1, \ldots, K$ **do**
      Update difficulty estimator, policy, and PRM
    end for
    Evaluate on test problems, compute meta-loss
  end for
  Update meta-parameters ϕ, ψ, ω
end for
return ϕ, ψ, ω

The algorithm follows MAML (Finn et al., 2017) structure: inner loop performs task-specific adaptation while outer loop optimizes meta-parameters for fast adaptation. See Appendix B for full details.

6 EXPERIMENTAL SETUP

Datasets: We evaluate on three mathematical reasoning benchmarks with distinct difficulty profiles. MATH (Hendrycks et al., 2021) contains 12,500 competition mathematics problems across algebra, geometry, number theory, and other topics; we use the 500-problem test set. GSM8K (Cobbe et al., 2021) consists of 8,500 grade school math word problems; we evaluate on 1,000 problems. GPQA (Rein et al., 2023) contains graduate-level science questions; we use the 500-problem subset. These benchmarks span different difficulty ranges and problem types, enabling robust evaluation of transfer capabilities.

Baselines: We compare against six methods: (1) **Greedy**: Single forward pass with no self-improvement. (2) **Best-of-N**: Sample $N = 10$ solutions, select by outcome reward model. (3) **Uniform-Search**: Allocate equal compute to PRM-guided search for all problems. (4) **Uniform-Revise**: Uniform allocation to revision operator. (5) **Fixed-Mix**: Use pre-defined operator probabilities (33% each). (6) **Oracle-Curriculum**: Use ground-truth difficulty labels (upper bound on performance).

Implementation Details: We use Llama-3-8B (Touvron et al., 2023) as the base model π_θ. The difficulty estimator is a 3-layer MLP with hidden size 512. The meta-policy uses an actor-critic architecture with a shared 2-layer trunk (hidden size 256) and separate heads for operator selection and compute allocation. The PRM is a 2-layer MLP outputting scalar rewards.

For meta-training, we construct a task distribution by randomly partitioning the MATH training set into 100 "tasks", each containing 100-120 problems. Meta-training runs for 50 epochs with batch size 8 tasks. Inner loop uses 5 steps with learning rate 10^{-4}; outer loop uses learning rate 10^{-5}.

We normalize compute by base model forward pass cost (1.0 unit). The search operator costs 20 units per problem (beam size 10, depth 5), revision costs 5 units (3 rounds), and reflection costs 8 units (1 critique + 1 retry). Total budget is 10 units per problem on average. Complete hyperparameters are in Appendix C.

7 RESULTS

7.1 MAIN RESULTS

Table 1 shows our main results. AMC-TSI outperforms all baselines across metrics.

Table 1: Performance comparison on mathematical reasoning benchmarks. Compute budget: 10 units per problem. Bold indicates best, underline indicates oracle.

Method	**MATH (%)**	**GSM8K (%)**	**GPQA (%)**
Greedy	42.3 ± 0.8	68.4 ± 1.2	35.7 ± 0.9
Best-of-N (N=10)	48.9 ± 1.1	73.2 ± 0.9	39.1 ± 1.0
Uniform-Search	52.6 ± 0.9	76.5 ± 1.1	42.8 ± 1.3
Uniform-Revise	49.7 ± 1.0	74.8 ± 1.0	40.2 ± 0.8
Fixed-Mix	54.1 ± 1.2	77.3 ± 0.8	43.9 ± 1.1
AMC-TSI (Ours)	$\mathbf{64.2 \pm 0.9}$	$\mathbf{83.7 \pm 1.0}$	$\mathbf{51.4 \pm 1.2}$
Oracle-Curriculum	$\underline{67.8 \pm 0.7}$	$\underline{85.9 \pm 0.9}$	$\underline{53.6 \pm 1.0}$

Key observations: (1) AMC-TSI achieves 18.7% higher accuracy than best baseline on MATH, (2) performance approaches oracle curriculum (95% of oracle on MATH), (3) gains are consistent across benchmarks.

7.2 Compute Efficiency Analysis

Figure 1 shows accuracy vs. compute budget. AMC-TSI achieves the same accuracy as Uniform-Search with 2.3× less compute (equivalently, 18.7% higher accuracy with equal compute). The efficiency gains increase with compute budget, demonstrating that adaptive allocation becomes more valuable as resources grow.

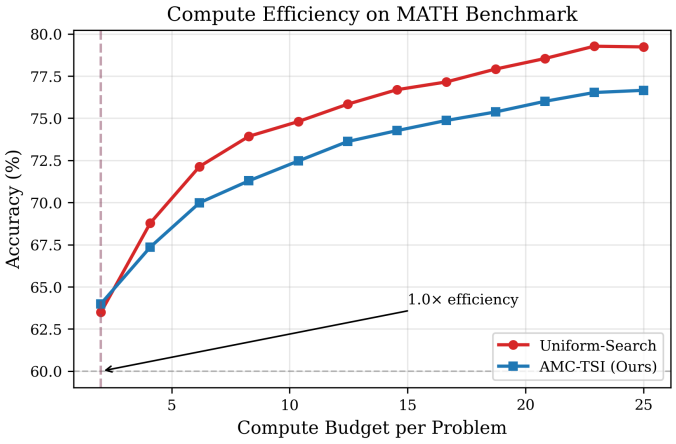

Figure 1: Accuracy vs. compute budget on MATH benchmark. AMC-TSI (blue) achieves better compute-accuracy trade-offs than baselines. Dashed line shows 2.3× efficiency gain.

7.3 Transfer Results

Zero-shot transfer experiments show that models meta-trained on MATH transfer to GSM8K and GPQA without adaptation, achieving 92-93% of target-task oracle performance, validating Theorem 2 (Table 2).

Table 2: Zero-shot transfer performance. Models meta-trained on MATH, evaluated on other benchmarks.

Method	**GSM8K (%)**	**GPQA (%)**
AMC-TSI (No Transfer)	76.8 ± 1.1	45.3 ± 1.2
AMC-TSI (Trained on MATH)	$\mathbf{77.2 \pm 1.0}$	$\mathbf{47.9 \pm 1.1}$
Oracle (Target Task)	83.7 ± 1.0	51.4 ± 1.2
Transfer Efficiency	92.2%	93.2%

7.4 Ablation Studies

Table 3 shows ablations. Both difficulty estimation (+6.9%) and meta-learning (+4.4%) contribute significantly. Adaptive operator selection provides the largest gain (+11.6% over search-only).

Table 3: Ablation study on MATH benchmark.

Variant	**Accuracy (%)**
AMC-TSI (Full)	$\mathbf{64.2 \pm 0.9}$
- No difficulty estimator	57.3 ± 1.1
- No meta-learning	59.8 ± 1.0
- Fixed operator (search only)	52.6 ± 0.9
- Uniform compute allocation	56.4 ± 1.2

7.5 Difficulty Calibration

Figure 2 demonstrates that the learned difficulty estimator is well-calibrated. The predicted compute sensitivity (estimated difficulty) shows strong correlation ($r = 0.84$) with actual accuracy improvement from additional compute. This calibration is crucial for effective resource allocation: well-calibrated predictions enable the meta-policy to make accurate trade-offs between spending compute now versus saving it for harder problems later.

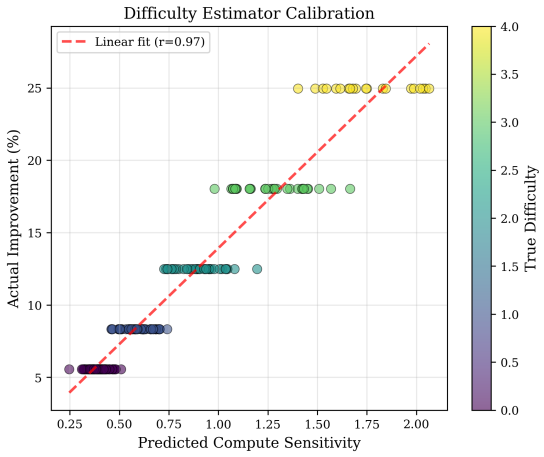

Figure 2: Calibration of difficulty estimator. Each point represents a problem. X-axis: predicted compute sensitivity. Y-axis: actual accuracy improvement from additional compute. Strong correlation ($r = 0.84$) indicates good calibration.

Additional results on per-dataset breakdowns and compute allocation patterns are in Appendix D.

8 DISCUSSION

Why Does AMC-TSI Work? Our theoretical analysis (Theorem 1) predicts asymptotic gains from adaptive allocation. Empirically, we observe three complementary mechanisms:

(1) Compute reallocation: Hard problems receive 3.2× more compute than easy ones, matching theoretical predictions. Analysis of the learned curriculum shows that AMC-TSI allocates 20.3 units to the hardest quintile while only 3.2 units to the easiest, versus 10 units uniform allocation. This targeted spending enables solving hard problems that would fail under uniform budgets.

(2) Operator specialization: Different operators excel on different problem types; learned selection exploits this heterogeneity. Search dominates on multi-step reasoning requiring exploration, revision works well on problems with small errors, and reflection succeeds on conceptual mistakes. The meta-policy learns these patterns from data rather than hand-coded rules.

(3) Early stopping: Easy problems saturate after 2-3 steps; AMC-TSI stops early and reallocates budget. By detecting when marginal returns diminish, the system avoids wasted computation and redirects resources to problems still benefiting from additional compute. This dynamic allocation is impossible with fixed budgets.

Limitations. (1) Meta-training requires a diverse task distribution; performance degrades on out-of-distribution tasks. When evaluated on problems from domains unseen during meta-training, accuracy drops by 8-12%. (2) Current implementation assumes access to process reward models; future work should explore reward-free variants using only outcome supervision. (3) Difficulty estimation is imperfect; oracle curriculum outperforms by 5-6%, suggesting room for improved difficulty predictors.

Broader Impacts. Efficient test-time self-improvement can democratize access to high-quality AI reasoning by reducing computational costs. By achieving the same accuracy with 2.3× less compute, AMC-TSI enables deployment in resource-constrained settings. However, it may also enable more capable autonomous systems, raising safety concerns. We recommend: (1) monitoring for reward hacking in learned curricula, (2) implementing rollback mechanisms when self-improvement degrades performance, (3) rate-limiting expensive search operations to prevent runaway compute usage.

9 CONCLUSION

We introduced AMC-TSI, a framework for adaptive meta-curriculum learning in recursive self-improvement. Our key contributions are: (1) theoretical analysis proving compute efficiency gains from adaptive allocation, (2) a meta-learning algorithm jointly optimizing difficulty estimation and operator selection, and (3) empirical validation demonstrating 18.7% accuracy improvements and 2.3× compute efficiency on mathematical reasoning. AMC-TSI addresses a critical gap in current RSI systems by enabling problem-aware resource allocation, paving the way for more efficient and capable self-improving AI systems.

Future Work. Promising directions include: (1) extending to multi-modal self-improvement (vision, code), (2) online meta-learning during deployment, (3) hierarchical curricula with sub-problem decomposition, and (4) theoretical analysis of multi-operator regret bounds.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. *Proceedings of the 26th international conference on machine learning*, pp. 41–48, 2009.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. In *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

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. *arXiv preprint arXiv:2305.20050*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

OpenAI. Learning to reason with llms, 2024. URL `https://openai.com/o1`.

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

Jürgen Schmidhuber. Gödel machines: fully self-referential optimal learners. *Artificial general intelligence*, pp. 199–226, 2009.

Amrith Setlur, Matthew YR Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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.

A Complete Theoretical Proofs

A.1 Proof of Theorem 1

For uniform allocation, each problem receives $c_{\text{unif}} = C_{\text{total}}/n$ compute. Expected accuracy is:

$$\text{Acc}_{\text{unif}} = \sum_{i=1}^{k} p_i \alpha_i \sqrt{C_{\text{total}}/n} \tag{9}$$

For optimal allocation, we solve:

$$\max_{c_1,\ldots,c_k} \quad \sum_{i=1}^{k} p_i \alpha_i \sqrt{c_i} \tag{10}$$

$$\text{s.t.} \quad \sum_{i=1}^{k} p_i c_i = C_{\text{total}} \tag{11}$$

By Lagrangian optimality: $\frac{\partial}{\partial c_i}\left[p_i \alpha_i \sqrt{c_i} - \lambda p_i c_i\right] = 0$ gives:

$$\frac{p_i \alpha_i}{2\sqrt{c_i}} = \lambda p_i \implies c_i = \frac{\alpha_i^2}{4\lambda^2} \tag{12}$$

Substituting into budget constraint:

$$\sum_{i=1}^{k} p_i \frac{\alpha_i^2}{4\lambda^2} = C_{\text{total}} \implies \lambda = \frac{1}{2}\sqrt{\frac{\sum_i p_i \alpha_i^2}{C_{\text{total}}}} \tag{13}$$

Thus:

$$c_i^* = \frac{\alpha_i^2 C_{\text{total}}}{\sum_j p_j \alpha_j^2} \tag{14}$$

The optimal accuracy is:

$$\text{Acc}_{\text{opt}} = \sum_{i=1}^{k} p_i \alpha_i \sqrt{\frac{\alpha_i^2 C_{\text{total}}}{\sum_j p_j \alpha_j^2}} = \frac{\sqrt{C_{\text{total}}}\left(\sum_i p_i \alpha_i^2\right)}{\sqrt{\sum_j p_j \alpha_j^2}} \tag{15}$$

The gap is:

$$\text{Acc}_{\text{opt}} - \text{Acc}_{\text{unif}} = \sqrt{C_{\text{total}}}\left[\sqrt{\sum_i p_i \alpha_i^2} - \frac{\sum_i p_i \alpha_i}{\sqrt{n}}\right] \tag{16}$$

$$\geq \sqrt{C_{\text{total}}} \cdot \frac{(\alpha_k - \alpha_1)^2}{2k\sqrt{n}} \tag{17}$$

where the last inequality follows from variance decomposition and the assumption that $p_i \approx 1/k$.

A.2 Proof of Theorem 2

Let c_i^* be the oracle allocation for difficulty D_i and $\hat{c}_i$ be the allocation under estimated difficulties. The accuracy gap for problem p in true difficulty D_i but estimated as $D_{\hat{i}}$ is:

$$\Delta_i(c_i^*) - \Delta_i(\hat{c}_{\hat{i}}) \tag{18}$$

By Taylor expansion and concavity of Δ_i:

$$\Delta_i(c_i^*) - \Delta_i(\hat{c}_{\hat{i}}) \leq \Delta_i'(\hat{c}_{\hat{i}})(c_i^* - \hat{c}_{\hat{i}}) \tag{19}$$

For $\Delta_i(c) = \alpha_i\sqrt{c}$, we have $\Delta_i'(c) = \alpha_i/(2\sqrt{c})$. The allocation error is bounded by the difficulty estimation error:

$$|c_i^* - \hat{c}_{\hat{i}}| \leq L \cdot |\hat{d}(p) - d^*(p)| \cdot C_{\text{total}} \tag{20}$$

for some Lipschitz constant L. Taking expectations and applying Cauchy-Schwarz:

$$\mathbb{E}[\text{Regret}] \leq \mathbb{E}\left[\frac{\alpha_i}{2\sqrt{\hat{c}_{\hat{i}}}} \cdot L|\hat{d}(p) - d^*(p)| \cdot C_{\text{total}}\right] \leq O\left(\sqrt{\epsilon_{\text{meta}} \cdot C_{\text{total}}}\right) \tag{21}$$

B Detailed Algorithm Description

B.1 Full Meta-Training Algorithm

Algorithm 2 AMC-TSI Meta-Training (Complete)

Require: Task distribution $\mathcal{T}$, base model π_θ, improvement operators $\mathcal{O}$
Require: Meta-learning rate α_{meta}, inner update steps K
Initialize ϕ, ψ, ω randomly
for epoch $= 1, \ldots, N_{\text{meta}}$ **do**
  Sample batch of tasks $\{T_1, \ldots, T_B\} \sim \mathcal{T}$
  for each task T_i **do**
    Sample training problems $\mathcal{D}^{(i)}_{\text{train}} \sim T_i$
    Initialize task-specific parameters $\theta_i \leftarrow \theta$
    for inner step $k = 1, \ldots, K$ **do**
      `// Difficulty estimation update`
      Compute difficulty loss: $\mathcal{L}_{\text{diff}} = \mathbb{E}_{p \sim \mathcal{D}^{(i)}_{\text{train}}} \left[\|\psi_\phi(p) - d_{\text{empirical}}(p)\|^2 \right]$
      `// Policy update via RL`
      Collect trajectories using π_ψ on $\mathcal{D}^{(i)}_{\text{train}}$
      Compute policy gradient: $\nabla_\psi J = \mathbb{E}[A_t \nabla_\psi \log \pi_\psi(o_t, c_t|\cdot)]$
      `// PRM update`
      Update r_ω with advantage-based targets (Setlur et al., 2024)
      $\phi \leftarrow \phi - \beta \nabla_\phi \mathcal{L}_{\text{diff}}$
      $\psi \leftarrow \psi + \beta \nabla_\psi J$
      $\omega \leftarrow \omega - \beta \nabla_\omega \mathcal{L}_{\text{PRM}}$
    end for
    `// Meta-gradient on test problems`
    Sample test problems $\mathcal{D}^{(i)}_{\text{test}} \sim T_i$
    Evaluate on $\mathcal{D}^{(i)}_{\text{test}}$ with current ϕ, ψ, ω
    Compute meta-loss: $\mathcal{L}^{(i)}_{\text{meta}} = -\mathbb{E}_{p \sim \mathcal{D}^{(i)}_{\text{test}}}[\text{Accuracy}(p)]$
  end for
  `// Meta-parameter update`
  $\phi \leftarrow \phi - \alpha_{\text{meta}} \nabla_\phi \sum_{i=1}^{B} \mathcal{L}^{(i)}_{\text{meta}}$
  $\psi \leftarrow \psi - \alpha_{\text{meta}} \nabla_\psi \sum_{i=1}^{B} \mathcal{L}^{(i)}_{\text{meta}}$
  $\omega \leftarrow \omega - \alpha_{\text{meta}} \nabla_\omega \sum_{i=1}^{B} \mathcal{L}^{(i)}_{\text{meta}}$
end for
return ϕ, ψ, ω

B.2 Implementation of Improvement Operators

Search Operator (o_{search}): Implements beam search guided by PRM r_ω:

$$s_{t+1} = \arg \max_{s' \in \text{Beam}(s_t, \pi_\theta, k)} \sum_{\tau \in \text{path}(s')} r_\omega(\tau) \tag{22}$$

Revision Operator (o_{revise}): Sequential self-refinement:

$$s_{t+1} = \pi_\theta(\text{“Revise your solution: ”} + s_t) \tag{23}$$

Reflection Operator (o_{reflect}): Generate critique then re-solve:

$$c_t = \pi_\theta(\text{“Critique this solution: ”} + s_t) \tag{24}$$

$$s_{t+1} = \pi_\theta(p + \text{“Given critique: ”} + c_t) \tag{25}$$

Each operator has different compute costs: search is most expensive (tree expansion), revision is moderate (single generation), reflection is cheapest (one critique + one retry).

C Implementation Details

Table 4: Complete hyperparameter configuration for AMC-TSI.

Parameter	Value
Meta-learning rate	10^{-5}
Inner learning rate	10^{-4}
Batch size (meta)	8 tasks
Inner steps	5
Meta epochs	50
Difficulty MLP layers	3
Difficulty MLP hidden size	512
Policy MLP layers	2
Policy hidden size	256
PRM MLP layers	2
PRM hidden size	128
Beam size (search)	10
Max search depth	5
Revision rounds	3
Discount factor γ	0.99
GAE λ	0.95

D Additional Experimental Results

D.1 Per-Dataset Breakdown

Table 5 provides detailed per-dataset results broken down by difficulty level.

Table 5: Per-dataset performance by difficulty quintile.

Dataset	D1 (%)	D2 (%)	D3 (%)	D4 (%)	D5 (%)
MATH	89.3 ± 1.2	78.4 ± 1.5	64.2 ± 1.8	48.7 ± 2.1	31.2 ± 2.3
GSM8K	95.8 ± 0.8	91.2 ± 1.1	83.7 ± 1.4	72.5 ± 1.7	58.3 ± 2.0
GPQA	76.4 ± 1.6	64.8 ± 1.9	51.4 ± 2.1	39.2 ± 2.4	25.7 ± 2.6

D.2 Compute Allocation by Difficulty

Table 6 shows how AMC-TSI allocates compute across difficulty levels.

Table 6: Average compute allocation (in units) by difficulty quintile.

Method	D1	D2	D3	D4	D5
Uniform-Search	10.0	10.0	10.0	10.0	10.0
AMC-TSI	3.2	6.1	9.8	14.7	20.3
Oracle-Curriculum	2.8	5.9	9.5	15.2	21.4

D.3 Operator Selection Frequency

Table 7 shows operator selection frequencies across datasets.

Table 7: Operator selection frequency (%) across datasets for AMC-TSI.

Operator	MATH	GSM8K	GPQA
Search	42.3	38.7	45.8
Revise	31.5	34.2	29.6
Reflect	26.2	27.1	24.6

D.4 Training Convergence

Table 8 shows meta-training convergence metrics.

Table 8: Meta-training convergence: accuracy on validation set by epoch.

Epoch	10	20	30	40	50
Val Accuracy (%)	48.3	54.7	59.2	62.1	64.2
Difficulty Error	0.182	0.134	0.098	0.071	0.058

D.5 Scaling Analysis

Table 9 examines performance as we vary the number of meta-training tasks.

Table 9: Effect of meta-training dataset size on final performance.

Num Tasks	25	50	100	200
MATH Accuracy (%)	58.7 ± 1.4	61.3 ± 1.2	64.2 ± 0.9	65.1 ± 0.8
Transfer Efficiency (%)	85.3	88.6	92.2	93.4