Learning to Discover at Test Time

Authors: Yuksekgonul et. al.
Presenters: Kangping Hu, Joshua Diao

Georgia Tech

Background

- Test-Time Scaling/Search: Using extra compute at inference to find better answers.
- Models generate thousands of attempts and filter by reward.
- Search relies on prompting frozen LLMs (e.g., AlphaEvolve).
- Fatal flaw: The model never updates its weights to internalize failures.

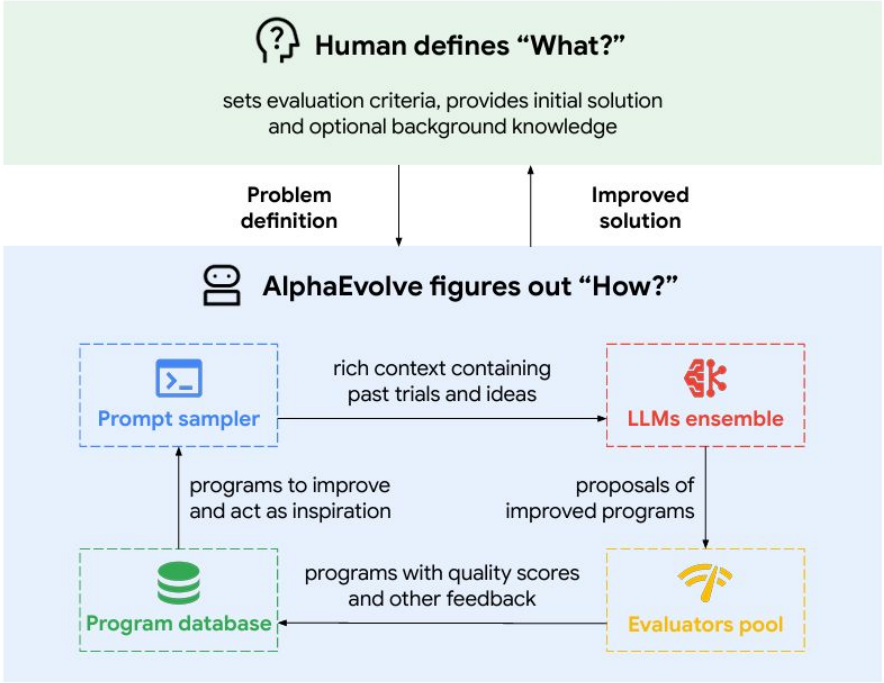


Figure 1 | *AlphaEvolve* high-level overview.

Georgia Tech

Problem Formulation

- Formulates scientific discovery as a Markov Decision Process.
- States (*s*): The current solution (code, proof, or algorithm).
- Actions (*a*): LLM generations (edits, thoughts, or new tokens).
- Rewards (*R*): Continuous score from a simulator or compiler.
- Discovery: An event where a state *s* is found such that $R(s) > r_{sota}$

Problem	State s	Action a	Transition	Reward $R(s)$
Erdős Minimum Overlap Autocorr. Inequality (1st) Autocorr. Inequality (2nd)	Step function certificate	Thinking tokens and code	$s' = \texttt{Python}(\texttt{Parse}(a))$	1/Upper bound 1/Upper bound Lower bound
Kernel Engineering Algorithm Competition Single Cell Analysis	Kernel code Algorithm code Analysis code	Thinking tokens and code	$s' = \texttt{Parse}(a)$	1/Runtime Test score 1/MSE

Table 1. Overview of the science and engineering problems in our paper, and the environments they induce (§2.1). Note that the reward is 0 if *s* fails validity checks.

Georgia Tech

Why Standard RL Fails for Discovery

- Standard RL optimizes for expected average performance (generalization).
- Train: Indifferent to state-of-the-art outliers.
- Reuse: empty.
- Discovery problems only care about the absolute maximum reward.
- One exceptional outlier is worth more than a thousand average attempts.

$$\texttt{train}: \quad \theta_{i+1} = \theta_i + \eta \nabla_\theta \mathbb{E}_{a \sim \pi_{\theta_i}(\cdot|s)}[R(s,a)], \qquad \texttt{reuse}(\mathcal{H}_i) = \delta_{\texttt{<empty>}},$$

Georgia Tech

Test-Time Training to Discover (TTT-Discover)

- Runs RL entirely at test time on a single target problem.
- Train: An Entropic Objective to update LLM weights.
- Reuse: A PUCT search subroutine to warm starts with previous solutions.
- Return the state with the highest reward.

Algorithm 1 Test-Time Training to Discover (TTT-Discover)

Input: problem description d and policy π_{θ_0} with initial weights θ_0.
$R, T = \texttt{get\_env}(d)$ ▷ d induces the reward and transition functions of the environment (§2.1)
$\mathcal{H}_0 = \{(\texttt{<empty>}, R(\texttt{<empty>}), \{\})\}$ ▷ Initialize buffer with the empty solution (§2.2)
for $i = 0, 1, \ldots, N-1$ **do**
  $s_i, c_i \sim \texttt{reuse}(\mathcal{H}_i)$ ▷ Sample initial state and context with a `reuse` heuristic
  $a_i \sim \pi_{\theta_i}(\cdot \mid d, s_i, c_i)$ ▷ Sample action from policy
  $s'_i = T(a_i)$ ▷ Transition to next state
  $r_i = R(s'_i)$ ▷ Evaluate reward of next state
  $\mathcal{H}_{i+1} = \mathcal{H}_i \cup \{(s_i, a_i, s'_i, r_i)\}$ ▷ Add current attempt to buffer
  $\theta_{i+1} = \texttt{train}(\theta_i, (d, s_i, c_i, a_i, r_i))$ ▷ Improve the model weights with `train`
end for
return s_{i^*}, where $i^* = \arg\max_{i=0,1,\ldots,N-1} r_i$ ▷ Return the state with the highest reward

Georgia Tech

The Entropic Objective

- Modifies the learning objective to exponentially amplify high rewards.
- Gradient weights: $w_{\beta(s)}(a) \propto e^{\beta(s)R(s,a)}$
- Disproportionately weights the absolute best trajectories.
- Explicitly incentivizes extreme outliers over safe averages.

$$J_\beta(\theta) = \mathbb{E}_{s\sim \text{reuse}(\mathcal{H})}\left[\log \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[e^{\beta(s)R(s,a)}\right]\right],$$

$$\nabla_\theta J_\beta(\theta) = \mathbb{E}_{\substack{s\sim \text{reuse}(\mathcal{H}) \\ a\sim\pi_\theta(\cdot|s)}}\left[w_{\beta(s)}(a)\nabla_\theta \log \pi_\theta(a \mid s)\right], \qquad w_{\beta(s)}(a) = \frac{e^{\beta(s)R(s,a)}}{\mathbb{E}_{\pi_\theta(\cdot|s)}\left[e^{\beta(s)R(s,a)}\right]},$$

Georgia Tech

Smart Exploration via Tree Search

- Uses a PUCT-inspired rule to select initial states for rollouts.
- Reuses the best partial solutions as starting points for new attempts.
- Balances exploring novel ideas against exploiting known high-reward paths.

$$\texttt{train}: \quad \theta_{i+1} = \theta_i + \eta \nabla_\theta J_{\beta(s_i)}(\theta_i), \qquad \texttt{reuse}: \quad s_i \sim \mathrm{PUCT}(\mathcal{H}_i).$$

$$\textbf{State reuse:} \quad s_i \sim \texttt{reuse}(\mathcal{H}_i),\ a_i \sim \pi_\theta(\cdot \mid d, s_i),\ \mathcal{H}_{i+1} = \mathcal{H}_i \cup \{(s_i', r_i)\}.$$

$$\textbf{State-action reuse:} \quad s_i, c_i \sim \texttt{reuse}(\mathcal{H}_i),\ a_i \sim \pi_\theta(\cdot \mid d, s_i, c_i),\ \mathcal{H}_{i+1} = \mathcal{H}_i \cup \{(s_i, a_i, s_i', r_i)\}.$$

Georgia Tech

Model and Training

- 120B open-weights model (gpt-oss-120b) via LoRA (Low-Rank Adaptation).
- Generate code rollouts via PUCT search.
- Evaluate in compiler/simulator target environment.
- Update 120B model weights via Entropic Objective.
- Repeat until termination.

Applications

Applications

- Evaluate TTT-Discover on problems in GPU kernel engineering, mathematics, algorithm design, and biology
- Each application, we report best known human results and best known AI results
- Report best-of-n baseline that matches the sampling budget and model that TTT-Discover uses
  - This is the control baseline we will refer to as LLM is frozen
  - We perform 50 steps with 512 rollouts per step, and compare to the Best-of-25600 baseline

1. Mathematics

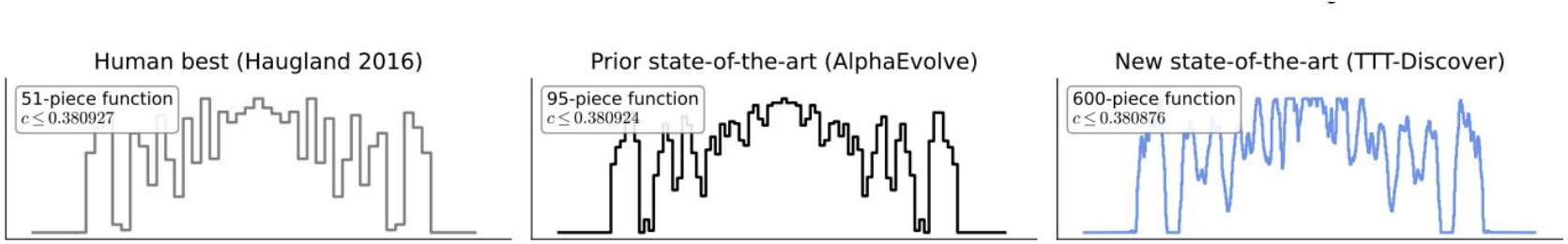
- Explore multiple open problems in mathematics, where small numerical improvements carry real weight
- Aim to create proofs by construction (i.e. step function or sequence) that certifies a condition (i.e. a bound of inequality can be achieved)

Environment

- State: A concrete mathematical construction - a step function or a numerical array
- Action: Thinking tokens followed by Python code that either creates a new construction or modifies an existing one
- Reward: The specific bound certified by the construction (e.g., the inverse of the upper bound for minimization problems)
- The model has a 10-minute limit per action to execute its code and evaluate the result

Erdos' Minimum Overlap Problem

- Classic problem in combinatorial number theory
  - Take numbers 1,2,...n
  - Split them into two groups of equal size
  - Add one number from each group to a sum
  - Goal: arrange split to number of times the same sum appears is small as possible
- Results
  - Improve upper bound on the problem to **0.380876**, from **0.380924** from Alpha Evolve
  - Discovered a 600-piece asymmetric step function (FFT-accelerated gradient descent)



Georgia Tech

Autocorrelation Inequalities

- Analysis problem that deal with properties of function when it is convolved with itself
- Focus on two primary inequality in terms of *f* step function
  - C_1 finds the tightest upper bound
  - C_2 finds the tightest lower bound
- Results
  - Improved upper bound to C_1 <= 1.50286 with a 30,000-piece step function
  - Did not improve lower bound, getting C_2 >= 0.959 - AlphaEvolve had a tighter lower bound of 0.961

$$\max_{|t|\le 1/2}(f*f)(t) \ge C_1\left(\int f\right)^2 \qquad C_2 = \sup_{f\ge 0}\frac{\|f*f\|_2^2}{\|f*f\|_1\,\|f*f\|_\infty}.$$

Georgia Tech

Mathematical Results

Method	Model	Erdős' (↓)	AC1 (↓)	AC2 (↑)
best human	–	0.380927	1.50973	0.9015
AlphaEvolve [50]	Gemini-2.0 Pro + Flash	0.380924	1.50530	0.8962
AlphaEvolve V2 [14]	Gemini-2.0 Pro + Flash	0.380924	1.50317	**0.9610**
ThetaEvolve [78]	R1-Qwen3-8B	n/a	1.50681	0.9468
ThetaEvolve w/ SOTA reuse (1.50317)	R1-Qwen3-8B	n/a	1.50314	n/a
OpenEvolve [61]	gpt-oss-120b	0.380965	1.50719	0.9449
Best-of-25600	gpt-oss-120b	0.380906	1.51004	0.9344
TTT-Discover	Qwen3-8B	0.380932	1.50525	0.9472
TTT-Discover	gpt-oss-120b	**0.380876**	**1.50287**	0.9591

Circle Packing

- Goal of the problem is to maximize the sum of radii of n non-overlapping circles packed inside inside a unit square

Environment

- State: List of circle centers and radii
- Action: Thinking tokens followed by Python code that optimizes circle positions and radii
- Reward: sum of radii achieved for valid packings

Results

- TTT-discover with Qwen3-8B matches best known constructions for both n values

Method	Model	$n = 26$ (↑)	$n = 32$ (↑)
AlphaEvolve [50]	Gemini-2.0 Pro + Flash	2.635862	2.937944
AlphaEvolve V2 [14]	Gemini-2.0 Pro + Flash	2.635983	2.939572
ShinkaEvolve [37]	Ensemble (see caption)	2.635982	n/a
ThetaEvolve [78]	R1-Qwen3-8B	2.635983	n/a
TTT-Discover	Qwen3-8B	2.635983	2.939572

Georgia Tech

2. Kernel Engineering

- Implement faster kernels through optimization methods
- Test TTT-discover on two competitions from GPUMODE (open community)
  - TriMul (triangular matrix multiplication)
  - DeepSeek MLA (multi-head latent attention)

Environment

- State: The current GPU kernel code
- Action: "Thinking tokens" followed by optimized kernel code written in Triton
- Reward: The inverse of the geometric mean of runtimes on a fixed set of input shapes. If the code failed correctness checks or timed out -> reward = 0
- Initial State: The process began with simple, unoptimized kernels

Results

- TriMul - achieved SOTA results across all GPU types
- MLA Decode - discovered kernels were not better than top human submission

Georgia Tech

Kernel Engineering Results

Method	Model	TriMul (↓, μs)			
		A100	H100	B200 [95% CI]	AMD MI300X [95% CI]
1st human	–	4531.5	1371.1	1038.9 [1016.3, 1061.6]	2515.8 [2510.9, 2520.7]
2nd human	–	4918.5	2368.0	2362.4 [2335.7, 2389.1]	5165.0 [5163.0, 5167.0]
3rd human	–	5182.2	2545.7	1931.0 [1910.9, 1951.1]	5359.3 [5343.5, 5375.1]
4th human	–	6097.8	3654.8	2248.9 [2089.4, 2408.4]	5981.4 [5978.4, 5984.4]
5th human	–	8345.0	4233.1	6503.8 [6400.5, 6607.1]	8365.1 [8347.7, 8382.5]
Best-of-25600	gpt-oss-120b	9219.7	5390.3	3254.9 [3252.5, 3257.4]	4902.0 [4897.6, 4906.4]
TTT-Discover	gpt-oss-120b	**2198.2**	**1161.2**	**914.2** [907.3, 921.1]	**1555.7** [1550.8, 1560.6]

Method	Model	AMD MI300X - MLA Decode (↓, μs) [95% CI]		
		Instance 1	Instance 2	Instance 3
1st human	–	1653.8 [1637.3, 1670.3]	1688.6 [1672.8, 1704.3]	1668.7 [1637.0, 1700.3]
2nd human	–	1662.8 [1648.8, 1676.8]	1688.6 [1677.6, 1699.5]	1679.7 [1653.4, 1705.9]
3rd human	–	1723.0 [1711.5, 1734.5]	1765.8 [1758.1, 1773.5]	1718.0 [1698.3, 1737.7]
4th human	–	1768.7 [1750.3, 1787.2]	1769.9 [1755.2, 1784.6]	1767.0 [1736.2, 1797.8]
5th human	–	2038.6 [2017.8, 2059.3]	2037.3 [2021.0, 2053.6]	2041.9 [1989.0, 2094.8]
Best-of-25600	gpt-oss-120b	2286.0 [2264.2, 2307.8]	2324.1 [2306.0, 2342.1]	2275.2 [2267.3, 2283.1]
TTT-Discover	gpt-oss-120b	1669.1 [1649.2, 1688.9]	1706.1 [1685.9, 1726.3]	1671.3 [1646.0, 1696.5]

Georgia Tech

3. Algorithm Engineering

- Hard optimization problems found in industries like factory production planning and package delivery routing - write a better algorithm
- Optimization challenges
  - ahc039 ("Purse Seine Fishing") - computational geometry problem where the AI must design a closed 2D polygon (a "net") to capture target points while avoiding penalty points under a budget
  - ahc058 ("Apple Incremental Game")- production planning problem where the AI must decide when to spend resources on upgrading a hierarchy of machines to maximize the final production output

Environment

- State: Candidate algorithm implementation written in C++
- Action: "Thinking tokens" followed by updated C++ code
- Reward: The score the code achieves when run against a local test case generator

Results

- First place for both competitions

Georgia Tech

Algorithm Engineering Results

Method	Model	Geometry (ahc039)	Scheduling (ahc058)
1st human	–	566,997	847,674,723
2nd human	–	557,212	846,938,871
3rd human	–	554,334	846,350,877
4th human	–	552,933	845,489,747
5th human	–	549,746	845,324,831
ALE-Agent [27]	Ensemble (see caption)	550,647	848,373,282
ShinkaEvolve [37]	Ensemble (see caption)	558,026	n/a
Best-of-25600	gpt-oss-120b	554,171	772,429,752
TTT-Discover	gpt-oss-120b	**567,062**	**848,414,228**

Georgia Tech

4. Single Cell Analysis

- Single cell RNA sequencing helps us understand how organisms work and get sick by resolving biology at individual cell level (what genes are used)
- Apply to recent benchmark OpenProblems (important set of open problems)

Environment

- State: Candidate implementation of a denoising algorithm
- Action: Reasoning followed by Python code
- Reward: The Mean Squared Error (MSE) score in log-normalized space with a "Poisson negative log-likelihood" metric is used as a hard constraint
    - if the algorithm fails this statistical check, it is rejected

Results

- Outperforms all existing methods

Single Cell Analysis Results

		PBMC			Tabula		
Method	**Model**	**Score (↑)**	**MSE (↓)**	**Poisson (↓)**	**Score (↑)**	**MSE (↓)**	**Poisson (↓)**
MAGIC (A, R)	–	0.64	0.19	0.05	0.64	0.18	0.03
MAGIC (R)	–	0.64	0.19	0.05	0.64	0.18	0.03
ALRA (S, RN)	–	0.50	0.26	0.05	0.47	0.27	0.03
MAGIC (A)	–	0.42	0.19	0.16	0.40	0.18	0.12
MAGIC	–	0.42	0.19	0.16	0.40	0.18	0.12
OpenEvolve	gpt-oss-120b	0.70	0.16	0.05	0.71	0.15	0.03
Best-of-25600	gpt-oss-120b	0.62	0.20	0.05	0.65	0.18	0.03
TTT-Discover	gpt-oss-120b	**0.71**	**0.15**	**0.05**	**0.73**	**0.14**	**0.03**

Georgia Tech

Ablations

- Three Sets of Ablations

1. Training Method - keep reuse method (PUCT) fixed
2. Reuse - keep adaptive entropic training fixed
3. Naive RL baseline - compare against standard non-discovery methods

	`train`	`reuse`	Best runtime ($\downarrow, \mu s$)
Best Human Kernel	–	–	1371.1
TTT-Discover	TTT with adaptive entropic	PUCT	**1203.10**
Ablations for `train`	TTT with constant β entropic	PUCT	1483.83
	TTT with expected reward (no entropic)	PUCT	1985.67
	No TTT	PUCT	2060.70
Ablations for `reuse`	TTT with adaptive entropic	ϵ-greedy	1328.89
	TTT with adaptive entropic	no reuse	5274.03
Naive Test-time RL	TTT with expected reward	no reuse	5328.73
Best-of-N	no TTT	no reuse	5352.36

Georgia Tech

Discussion

Related Work

- Continual Learning: Adapting to changing distributions over time.
  - Contrast: TTT-Discover adapts to a single target problem.
- Test-Time Training: Optimizing models on unlabeled data at inference.
  - Contrast: TTT-Discover targets the maximum outlier, not average test loss.
- RL on a Single Training Problem.
  - Contrast: TTT-Discover trains directly on the test problem.
- RL on the Entire Test Set.
  - Contrast: TTT-Discover isolates optimization to one specific problem at a time.

Future Work

- Current limitation: Strictly requires continuous, programmatic rewards.
- Future Direction 1: Scaling to problems with sparse or binary rewards.
- Future Direction 2: Operating in non-verifiable domains.

Critique

- The Claim: TTT-Discover is a form of continual learning.
- The Reality: Extreme overfitting to a single test environment.
- The model acquires no generalizable knowledge.
- The policy is discarded immediately after solving the problem.
- Therefore, it acts as a highly effective, differentiable search heuristic.
- Continual from the start, but unable to continue learning afterwards