SELF-IMPROVING LANGUAGE MODELS WITH BIDIRECTIONAL EVOLUTIONARY SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Search has been proposed as an effective method for self-improving language models and agentic systems, both for post-training sample generation and for inference. However, widely used methods such as best-of-N sampling and tree search face two fundamental limitations: they are guided by sparse verification signals, and they construct candidates primarily through autoregressive expansion, restricting exploration to regions with substantial model probability mass. To address these, we propose Bidirectional Evolutionary Search (`BES`), a search framework that couples forward candidate evolution with backward goal decomposition. In the forward search, `BES` augments standard expansion with evolution operators that recombine partial trajectories to generate candidates that are difficult to obtain from a single model rollout. In the backward search, `BES` recursively decomposes the original task into checkable sub-goals, producing dense intermediate feedback that guides forward search. We provide motivation illustrating that candidates generated by expansion-only search may be confined to a narrow entropy shell, while evolution operators may escape it. Experiments show that on challenging post-training tasks where mainstream post-training algorithms fail to improve, `BES` enables consistent gains. At inference time, `BES` outperforms existing open-source frameworks on three open problem solving benchmarks in both average and best-case performance, and outperforms majority voting and Monte Carlo Tree Search on mathematical reasoning and competitive programming benchmarks.

1 INTRODUCTION

Large language models (LLMs) and agentic systems have demonstrated remarkable capabilities on complex reasoning problems (Qi et al., 2025; Guo et al., 2025). They can even solve open problems across mathematical and scientific domains (Novikov et al., 2025; Hubert et al., 2026) and surpass the best human performance on tasks such as code generation (Li et al., 2022; Yuksekgonul et al., 2026). In this context, the question of how to do better sampling from LLMs and agentic systems is of critical importance (Guan et al., 2025). This is particularly significant for problems at the frontier of model capability, where naive sampling methods may require too many samples to obtain a correct answer or may simply fail (Glazer et al., 2025; Yue et al., 2026). At training time, higher-quality samples enable more effective post-training and self-improvement (Zelikman et al., 2022; Yuan et al., 2024); at inference time, they serve as a natural mechanism for test-time scaling (Zelikman et al., 2024; Lange et al., 2025), which can further push the boundary of what models can achieve.

Currently, the two dominant sampling methods in post-training, self-improvement, and inference for LLMs and agentic systems are best-of-N sampling and tree search. Best-of-N sampling is simple and efficient. For problems of moderate difficulty, it typically suffices to find high-quality responses and is therefore widely adopted in post-training algorithms such as GRPO (Guo et al., 2025) and its variants (Yu et al., 2025), while also serving as a strong baseline for inference. Tree search methods such as beam search and Monte Carlo Tree Search (Kocsis & Szepesvari, 2006) can discover better responses more sample-efficiently than best-of-N on harder problems. For example, Tree-GRPO leverages tree search for sample generation during post-training (Ji et al., 2026), and Tree of Thoughts explores multiple reasoning paths at inference time (Yao et al., 2023).

However, both methods share two limitations. (1) The verification signal to guide the search is sparse. Effective search depends on the accuracy and granularity of the verifier, yet in common settings such

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.

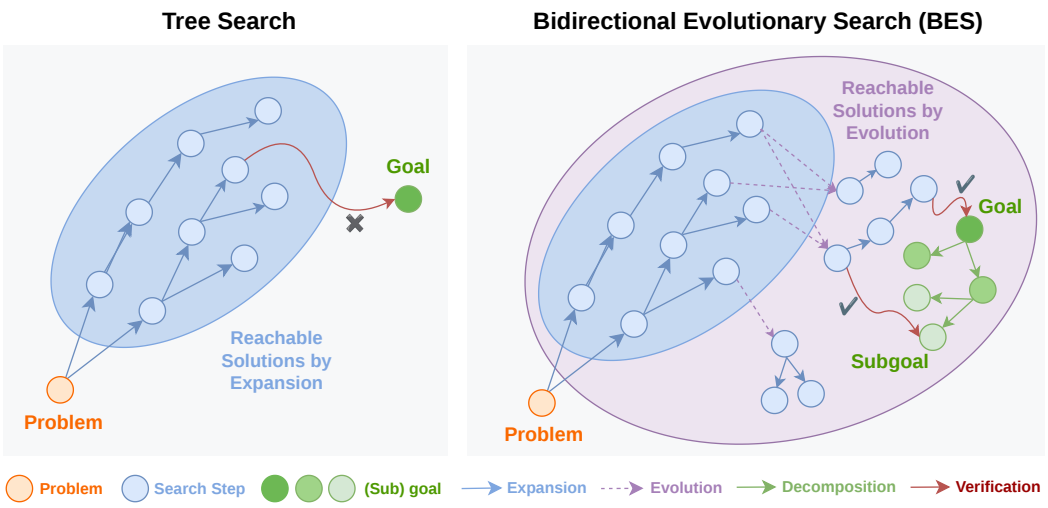

Figure 1: Comparison of tree search and Bidirectional Evolutionary Search (BES). **Left:** Tree search constructs candidates by sequentially expanding steps. Our motivating analysis suggests that such candidates are typically confined to a narrow entropy shell, limiting exploration to a small region of the solution space. **Right:** Evolution operators can produce candidates outside this shell, with backward search decomposing the problem into verifiable sub-goals that provide dense feedback to guide the forward search. ✓ and ✗ indicate whether a candidate satisfies or fails the (sub-)goal.

as RLVR post-training, verifiers typically provide only binary or coarse-grained feedback. (2) They struggle to generate candidates beyond the model's own distribution. They construct candidates by auto-regressively extending responses. This concentrates candidates in the high-probability region of the model's distribution, so trajectories the model assigns low probability are rarely sampled (Brown et al., 2024), even though they may be needed on hard problems.

To address these two limitations, we propose Bidirectional Evolutionary Search (BES). First, to tackle the sparsity of verification signals, we introduce bidirectional search: the forward search seeks better candidate solutions, while the backward search discovers finer-grained sub-goals to verify them. Second, to generate candidates beyond the model's own distribution, we draw inspiration from evolutionary biology. For much of the history of life, organisms reproduced asexually. Each offspring was a direct extension of its parent, and beneficial mutations arising independently in different individuals could never be combined (Fisher, 1999). Sexual reproduction fundamentally changed this through chromosomal recombination: gene segments from different lineages are spliced together to produce novel combinations that neither parent possessed (Muller, 1932). Analogously, in our setting, rather than only extending responses auto-regressively, we introduce four evolution operators: combination, translocation, deletion, and crossover. Combination, translocation, and crossover merge the strengths of two distinct responses in different ways to construct a new candidate, while deletion removes the least sound segment from a response. We give a motivating analysis suggesting that responses generated by expansion-only search may be confined to a narrow entropy shell, while evolution operators may escape it.

We evaluate BES on both post-training and inference across LLM and agent settings. For post-training, we test on challenging logical reasoning and multi-hop reasoning tasks where mainstream post-training algorithms such as GRPO (Guo et al., 2025), MaxRL (Tajwar et al., 2026), and Tree-GRPO (Ji et al., 2026) struggle to find sufficient high-quality training samples and consequently fail to improve or even degrade from the base model. In contrast, BES consistently discovers effective training samples, enabling meaningful improvements where these baselines nearly fail. For inference, we test on three open problem solving benchmarks, where BES discovers more stable and higher-quality solutions compared to all existing open-source frameworks, including OpenEvolve (Sharma, 2025), GEPA (Agrawal et al., 2026), and ShinkaEvolve (Lange et al., 2025). We further evaluate BES on mathematical reasoning (AIME (Zhang & Team, 2025)) and competitive programming (LiveCodeBench (Jain et al., 2025)) benchmarks, where BES outperforms majority voting and Monte Carlo Tree Search (Kocsis & Szepesvari, 2006) under the same compute budget. We also provide ablation studies validating the contribution of each component, case studies visualizing our search process, and cost analysis comparing the wall-clock time of BES against baselines.

2 PRELIMINARIES

We consider reasoning problems of the form $\mathcal{T} = (x, V)$, where x is a problem description and $V(x, y) \in [0, 1]$ is a *verifier* that assigns a score measuring how well a trajectory y solves problem x.

Given a policy $\pi_\theta(\cdot \mid x)$ (e.g. an LLM that generates reasoning traces, or an agent that interacts with an environment through a sequence of actions), our goal is to produce a terminal response y that maximizes the verifier score. Let $\mathcal{Y}_{\text{term}}(x)$ denote the set of valid terminal responses under the task specification and resource budget. We write the target as $y^\star(x) \in \arg\max_{y \in \mathcal{Y}_{\text{term}}(x)} V(x, y)$. Finding $y^\star$ is important in both training and inference. During training, high-quality candidates can facilitate post-training or iterative self-improvement. During inference, $y^\star$ is the response we seek. The challenge is that the maximization is intractable: on hard problems, the probability mass that π_θ assigns to correct trajectories can be extremely small. Practical algorithms therefore approximate $y^\star$ by searching over a set of candidates. We introduce two common methods below.

Best-of-N sampling. The simplest approach is to draw N independent trajectories and return the one with the highest verifier score. Formally, let $y^{(1)}, \ldots, y^{(N)} \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x)$. The method returns $y^{\text{BoN}} = \arg\max_{y \in \{y^{(1)}, \ldots, y^{(N)}\}} V(x, y)$. Best-of-$N$ is parallel and requires no structural knowledge of the problem. Its effectiveness, however, is bounded by the finite-sample coverage of π_θ: all N trajectories are drawn from the same distribution, so if the optimal trajectory lies in a region with very small policy mass, increasing N gives only linear coverage improvement.

Tree search. Tree-search methods exploit the sequential structure of a trajectory. A trajectory is decomposed into steps, and the search maintains a tree whose nodes are partial trajectories and whose edges correspond to appending a step. Branches are selected and extended according to a heuristic value so that compute is concentrated on prefixes that look promising. Classic methods include beam search, best-first search, and Monte Carlo Tree Search (Kocsis & Szepesvari, 2006); recent work applies these ideas to LLM and agent reasoning (Yao et al., 2023; Ji et al., 2026). Tree search can be more sample-efficient than best-of-N when the per-step signal is informative, but it still materializes terminal candidates one lineage at a time through sequential extension.

3 BES: BIDIRECTIONAL EVOLUTIONARY SEARCH

BES performs a bidirectional evolutionary search that alternates between two coupled processes: a *forward* search that seeks better candidates, and a *backward* search that decomposes the problem into fine-grained sub-goals to evaluate each forward node. The full pseudocode is given in Appendix B, and a case study illustrating the search process is provided in Appendix F.

3.1 FORWARD SEARCH: EXPANDING THE REACHABLE SOLUTION SPACE

We represent each candidate partial trajectory as a node $n = (y_1, \ldots, y_t)$, where y_i denotes the i-th step (e.g. a reasoning segment or an action). The search maintains a candidate set $\mathcal{P}$ of such nodes, and at each search step applies one of two types of operators, *expansion* or *evolution*, to produce a child node n', which is scored by the backward search (Section 3.2) and added to $\mathcal{P}$.

Expansion. Expansion extends a parent node by sampling new steps. Given $n = (y_1, \ldots, y_t)$, we sample a step count $K \sim \text{Uniform}\{1, \ldots, K_{\max}\}$ and draw up to K new steps from π_θ:

$$y_{t+k} \sim \pi_\theta\big(\cdot \mid x \oplus y_1 \oplus \cdots \oplus y_{t+k-1}\big), \qquad k = 1, \ldots, K, \tag{1}$$

Evolution. A key limitation of expansion alone is that each candidate is built by sequentially extending a single trajectory, so it cannot combine useful parts from different candidates. Evolution operators overcome this by constructing a new candidate from the content of existing ones. As illustrated in Figure 2, we define four operators inspired by biological evolution, each specified by what it takes from which parent: *(i) Combination* produces a candidate that draws on the content of two trajectories beyond what they already share; *(ii) Deletion* removes one segment of a trajectory, producing a condensed candidate; *(iii) Translocation* transplants a single useful segment from a donor trajectory into a recipient ; and *(iv) Crossover* continues the beginning of one trajectory with the latter part of another. Together, these operators allow the search to restructure, condense, and recombine existing trajectories. As with mutations in nature, evolution operations are not guaranteed to produce

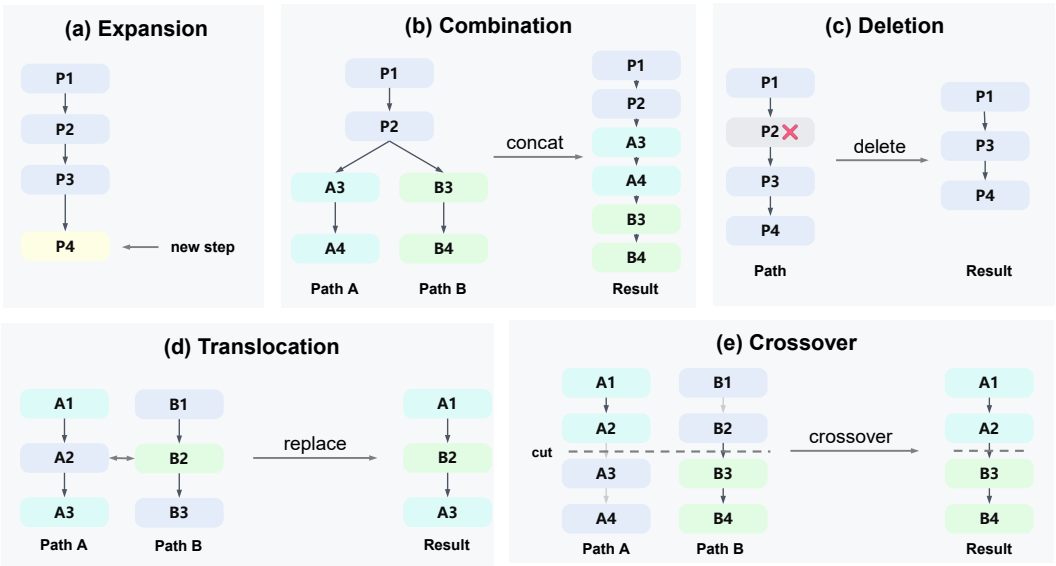

Figure 2: Forward search operators. **(a) Expansion:** the policy generates new steps (yellow). **(b) Combination:** beyond their shared part (P1–P2), the contents of Path A and Path B are merged into a single candidate. **(c) Deletion:** one step (P2) is removed. **(d) Translocation:** a useful step from Path B (B2) is transplanted into Path A, replacing A2, while the rest of Path A is kept. **(e) Crossover:** the beginning of Path A (A1–A2) is continued with the latter part of Path B (B3–B4). The operators are realized by prompting when direct edits are not meaningful.

better candidates. However, the value of evolution lies in generating diverse candidates: ***even if only a small fraction turn out to be improvements, that is sufficient for the search to make progress.*** In practice, we instantiate the operators either as direct edits when steps are self-contained with clear boundaries (Appendix C), or by prompting the model when direct edits are not meaningful.

At each forward search step, we select among the operators with fixed probabilities. All operators require selecting one or two parent nodes from $\mathcal{P}$. Let $\mathcal{C}_u$ be the set of eligible members of $\mathcal{P}$ at step u. For single-parent operators (expansion, deletion), we sample the parent from a Boltzmann distribution over backward score $s(n)$ (Eq. 4):

$$\Pr\big[n \mid \mathcal{C}_u\big] \;=\; \frac{\exp(\tilde{s}(n)/\tau_u)}{\sum_{n' \in \mathcal{C}_u} \exp(\tilde{s}(n')/\tau_u)}, \qquad \tilde{s}(n) \;=\; s(n) + \lambda \cdot \mathbf{1}\big[\deg(n) = 0\big], \tag{2}$$

where $\tau_u > 0$, $\deg(n)$ is the number of children of n in $\mathcal{P}$, and the indicator term adds a small constant bonus $\lambda = 0.1$ to candidates that have not yet been selected as parents.

For two-parent operators, we select a pair of parent nodes (n_a, n_b) from $\mathcal{C}_u$. We calculate a pair score $s(n_a, n_b)$ (Eq. 5) that favors complementary parents whose strengths cover different parts of the problem. The pair is then drawn from the analogous Boltzmann distribution:

$$\Pr[(n_a, n_b) \mid \mathcal{C}_u] \;=\; \frac{\exp(s(n_a, n_b)/\tau_u)}{\sum_{n_a, n_b \in \mathcal{C}_u} \exp(s(n_a, n_b)/\tau_u)}. \tag{3}$$

The temperature τ_u is annealed linearly over the search, shifting from exploration to exploitation.

We give an intuition for why evolution operators may produce candidates that expansion rarely generates. For simplicity, we study the direct-edit instantiation of the evolution operators. Fix a task x and a step horizon T. A trajectory $Y = (y_1, \ldots, y_T)$ is generated by the policy π_θ, which induces the trajectory distribution $P(Y) := \pi_\theta(Y \mid x)$. Let $H_T = H_P(Y)$ denote the trajectory-level entropy, and define, for a constant $\epsilon > 0$, the typical shell $A^{(T)} := \{y : |-\log P(y) - H_T| \le \epsilon T\}$. Under assumptions on token dependence and next-token uncertainty (Appendix D.1), we have:

Motivation 3.1 (Shell confinement and escape). ***(a)*** *A policy rollout $Y \sim P$ stays within the shell with high probability:* $\Pr[Y \notin A^{(T)}] \le \exp(-\Omega(T))$. ***(b)*** *A candidate $\widetilde{Y}$ formed by splicing segments of independent rollouts can escape it:* $\Pr[\widetilde{Y} \notin A^{(T)}] > 0$.

Escaping the shell indicates that such candidates are generated with non-negligible probability, not that they are correct. Identifying useful candidates among them is the role of the backward search. Details are illustrated in Appendix D.

3.2 Backward Search: Better Verification through Goal Decomposition

While the verifier V provides a score for each node in the forward search, this signal is relatively sparse. Backward search addresses this by decomposing the problem into a tree of fine-grained sub-goals. Each forward node is then scored against this tree: the more sub-goals a candidate trajectory has addressed, the higher its score. This gives the forward search a dense, informative signal to select promising candidates, even when none of them have fully solved the problem yet.

Below we describe how the backward search is constructed. Starting from the top-level goal $g_{\texttt{root}}$ (i.e. solving the entire problem), the policy π_θ is prompted to break each goal into finer sub-goals, producing a rooted backward goal tree. Each goal g on the tree can be recursively split into children $\mathrm{ch}(g)$ (finer sub-goals), and every g comes with a verifier $V_g(x, n) \in [0, 1]$ that tests how well a candidate node n addresses the sub-goal g on problem x. For the top-level goal, $V_{g_{\texttt{root}}} = V$ (the verifier of the original problem when available, and a model-generated verifier otherwise). This decomposition is re-invoked every K_{dec} forward search steps: at each invocation, we select a leaf sub-goal that no current candidate fully satisfies and prompt π_θ to split it into finer sub-goals. After that, all existing forward nodes are re-scored. The verifiers are task-dependent and can be instantiated as rule-based checkers, test-case code executors, embedding similarity models, or LLM judgers. We describe the verifiers used in each experiment in Appendix E.

For example, consider the problem "Compute $\frac{(4+6)\times 3}{2} - 5$." The backward search can produce:

$g_{\texttt{root}}$: compute $\frac{(4+6)\times 3}{2} - 5$
- g_1: compute $(4+6)\times 3$
  - $g_{1.1}$: compute $4+6$
  - $g_{1.2}$: #1 multiply by 3
- g_2: #1 divide by 2
- g_3: #2 subtract 5

For a candidate node n from the forward search and a sub-goal g, define the sub-goal score

$$s(n, g) \;=\; \alpha \cdot V_g(x, n) \;+\; (1-\alpha) \cdot \frac{1}{|\mathrm{ch}(g)|} \sum_{g' \in \mathrm{ch}(g)} s(n, g'), \tag{4}$$

where $\alpha \in [0, 1]$ balances the contribution of parent goals and sub-goals. For leaf sub-goals ($\mathrm{ch}(g) = \emptyset$), we set $s(n, g) = V_g(x, n)$. If a goal is already fully satisfied ($V_g(x, n) = 1$), we short-circuit to $s(n, g) = 1$ without evaluating its children. The overall score is $s(n) \triangleq s(n, g_{\texttt{root}})$.

For two candidate nodes n_a and n_b, we further define a pair score that measures their joint coverage of the goal tree. Replacing each sub-goal verifier with the maximum of the two parents' scores gives

$$s(n_a, n_b, g) \;=\; \alpha \cdot \max\{V_g(x, n_a),\, V_g(x, n_b)\} \;+\; (1-\alpha) \cdot \frac{1}{|\mathrm{ch}(g)|} \sum_{g' \in \mathrm{ch}(g)} s(n_a, n_b, g'), \tag{5}$$

with $s(n_a, n_b) \triangleq s(n_a, n_b, g_{\texttt{root}})$. A sub-goal is considered covered if either parent addresses it, so the pair score favors complementary parents that cover different parts of the goal tree.

Applying this procedure to every node in the forward search tree yields a backward score for each node and a pair score for each pair of nodes, which directly drive node selection in the forward search.

3.3 Using BES for Post-Training and Inference

Post-Training. BES replaces the sample-generation of post-training. Given a problem, BES returns a set of high-quality trajectories. These are used as training data for post-training, replacing candidates that would otherwise be obtained from i.i.d. rollouts. Because BES can find correct solutions that single rollouts rarely reach, it provides stronger training signal, especially on hard problems. Here, the verifier is available from training labels and is used to identify successful trajectories.

Inference. At inference time, BES runs with a fixed compute budget and returns the highest-scoring terminal trajectory as the final answer. For mathematical reasoning and code generation, where a ground-truth verifier is unavailable to the model, the policy writes a verifier for the top-level goal, and BES returns the terminal trajectory that scores highest under it.

4 EXPERIMENTS

We evaluate BES on both post-training and inference across LLM and agent settings. For post-training, we consider multi-hop reasoning and logical reasoning. For inference, we consider open problem solving, mathematical reasoning, and competitive programming. In each benchmark, we compare BES against commonly used baselines and show that BES consistently achieves better sampling.

4.1 BIDIRECTIONAL EVOLUTIONARY SEARCH FOR POST-TRAINING

4.1.1 MULTI-HOP REASONING

Table 1: Multi-hop reasoning post-training results. We report accuracy (%) on the MuSiQue validation set (in-distribution) and on Bamboogle (out-of-distribution); ± denotes the binomial standard error. **Bold** denotes the best method for each backbone and test set.

Method	Test Dataset	Llama-3.2-3B-Instruct	Llama-3.1-8B-Instruct
Base model	MuSiQue (ID)	4.01 ± 0.40	6.58 ± 0.50
	Bamboogle (OOD)	24.0 ± 3.8	22.4 ± 3.7
GRPO	MuSiQue (ID)	2.11 ± 0.29	5.59 ± 0.47
	Bamboogle (OOD)	16.8 ± 3.3	20.0 ± 3.6
Tree-GRPO	MuSiQue (ID)	3.93 ± 0.40	7.45 ± 0.53
	Bamboogle (OOD)	23.2 ± 3.8	28.8 ± 4.1
GRPO + MCTS	MuSiQue (ID)	4.30 ± 0.41	6.90 ± 0.52
	Bamboogle (OOD)	18.4 ± 3.5	29.6 ± 4.1
GRPO + BES (ours)	MuSiQue (ID)	**9.08** ± 0.58	**10.74** ± 0.63
	Bamboogle (OOD)	**30.4** ± 4.1	**35.5** ± 4.3

For multi-hop reasoning, we train on the MuSiQue dataset (Trivedi et al., 2022), in which the agent must answer complex questions that require retrieving and integrating information across multiple documents. We evaluate on the MuSiQue validation set (in-distribution), and Bamboogle (Press et al., 2023), a benchmark of compositional two-hop questions constructed from a different source (out-of-distribution).

Baselines. We compare against GRPO (Guo et al., 2025) and Tree-GRPO (Ji et al., 2026), with the latter being the current state-of-the-art method for search agent post-training. We apply BES on top of GRPO. Since BES is a sampling algorithm, we additionally compare against GRPO with MCTS sampling, which replaces the rollouts of GRPO with trajectories found by Monte Carlo Tree Search (Qi et al., 2025; Guan et al., 2025).

Training Setup. We adopt the training setup of Tree-GRPO (Ji et al., 2026), using Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as base models. During the search process, the agent can perform multiple search actions before producing a final answer. Search results are provided by an offline Wikipedia server. We use the solvable subset of the MuSiQue training set as our training data and train for 2 epochs. We evaluate on the full official MuSiQue validation set. Detailed configurations are provided in Appendix E.1.

As shown in Table 1, GRPO falls below the base model on both backbones, while Tree-GRPO and MCTS yield limited changes. We attribute the degradation of GRPO to reward sparsity: on MuSiQue, 93% of GRPO groups receive zero reward for all rollouts, so the group-normalized advantage vanishes and these problems provide no learning signal. The GRPO-trained policy also learns to search less, reducing the number of valid <search> calls per question from 1.62 to 0.84 (Table 4), which hurts on questions that require multiple hops. In contrast, GRPO + BES improves MuSiQue accuracy from 4.01% to 9.08% (3B) and from 6.58% to 10.74% (8B), and Bamboogle accuracy by 6.4 and 13.1 points, respectively. See Section 4.4 for a compute comparison.

4.1.2 LOGICAL REASONING

For logical reasoning, we use the Knights-and-Knaves dataset (Xie et al., 2024a). In this task, each puzzle involves a group of people who are either a knight or a knave, and the solver must deduce each one's identity.

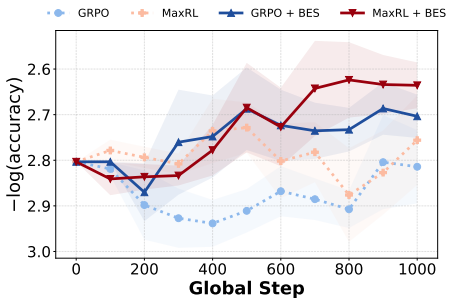

Figure 3: EMA-smoothed validation accuracy on logical reasoning (Knights-and-Knaves).

Baselines. We select two post-training algorithms, GRPO (Guo et al., 2025) and MaxRL (Tajwar et al., 2026), as baselines, as we apply BES on top of both GRPO and MaxRL. We note that BES is a sampling algorithm that is agnostic to the training method and can be applied on top of any post-training algorithm.

Training Setup. We use Gemma-3-1B-it (Team et al., 2025) as the base model. To adapt the model to Knights-and-Knaves, we first perform a cold start by fine-tuning on 1K SFT examples generated using the official data generation pipeline for 3 epochs, followed by 4 epochs of post-training on 5K problems, during which we compare the different methods on a 1.3K validation set. Detailed experimental configurations are provided in Appendix E.2.

As shown in Figure 3, due to the difficulty of the training set, GRPO and MaxRL show little to no improvement during training, while BES improves throughout training, demonstrating that BES is more effective at discovering high-quality training samples.

4.2 BIDIRECTIONAL EVOLUTIONARY SEARCH FOR INFERENCE

4.2.1 OPEN PROBLEM SOLVING

Here, we consider three benchmarks: Circle Packing (Square), which seeks to pack N circles into a unit square and maximize the sum of radii; Circle Packing (Rect), the analogous problem for a rectangular container; and Heilbronn (Convex), which places 13 points in a convex region of unit area to maximize the minimum area of any triangle formed by three of the points.

Table 2: Open problem solving benchmarks with GPT-5 as the backbone model. All open-source frameworks are run under the same backbone model, API budget, and evaluation protocol. We report mean ± standard deviation and best objective values (↑) over 3 runs. **Bold** denotes the best performing framework; human and AlphaEvolve results are listed for reference only.

Strategy	**Circle Packing (Square)**		**Circle Packing (Rect.)**		**Heilbronn (Convex)**	
	Avg.	Best	Avg.	Best	Avg.	Best
Human & High-compute closed-source framework						
Human	–	2.634	–	2.364	–	0.0306
AlphaEvolve	–	2.635	–	2.3658	–	0.0309
Open-source frameworks						
OpenEvolve	2.418 ± .123	2.541	2.270 ± .025	2.313	0.020 ± .002	0.022
GEPA	2.537 ± .004	2.541	2.274 ± .046	2.351	0.025 ± .001	0.026
ShinkaEvolve	2.478 ± .076	2.565	2.332 ± .023	2.357	0.023 ± .001	0.024
ShinkaEvolve + BES	**2.623** ± .014	**2.632**	**2.349** ± .012	**2.360**	**0.026** ± .001	**0.027**

Baselines. We compare against three open-source frameworks: OpenEvolve (Sharma, 2025), GEPA (Agrawal et al., 2026), and ShinkaEvolve (Lange et al., 2025). All baseline results use the same backbone model, compute budget, and configuration as ours. We apply BES on top of

Table 3: Inference-time results on AIME 2024–2025 and LiveCodeBench v6 with Qwen3 backbones. We report accuracy (%) as mean ± standard error over 3 runs. All search-based methods use the same compute budget per problem. **Bold** denotes the best method per column.

Method	AIME 2024–2025			LiveCodeBench v6		
	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Zero-shot	15.6 ± 1.5	13.9 ± 1.5	33.9 ± 0.6	26.3 ± 0.1	24.9 ± 0.9	28.3 ± 1.4
Majority-of-N	28.3 ± 1.7	38.3 ± 1.7	46.7 ± 0.0	27.8 ± 1.2	26.3 ± 0.6	29.4 ± 0.3
MCTS + PRM	**30.0** ± 1.0	38.3 ± 2.5	43.3 ± 1.0	30.6 ± 0.2	30.3 ± 0.1	34.0 ± 0.3
BES (ours)	**30.0** ± 1.7	**39.4** ± 0.6	**49.4** ± 0.6	**31.9** ± 0.4	**33.1** ± 0.6	**37.1** ± 1.7

ShinkaEvolve. For reference, we also include the results of human experts and AlphaEvolve (Novikov et al., 2025), which is closed-source and uses significantly more compute than all other frameworks.

Setup. ShinkaEvolve maintains a population of candidate programs and iteratively proposes modifications. Since directly concatenating model outputs is not meaningful on this benchmark (where candidates are executable programs), we implement the evolution operators by prompting LLMs. We use GPT-5 as the backbone model. Detailed configurations are provided in Appendix E.3.

As shown in Table 2, BES outperforms all open-source frameworks on average. Notably, BES also exhibits the lowest run-to-run variance on average across the three benchmarks, indicating that the search is more stable. The advantage of BES also holds with open-weight backbones (Appendix G).

4.2.2 Mathematical Reasoning & Competitive Programming

For mathematical reasoning, we use problems of AIME 2024 and AIME 2025 (Zhang & Team, 2025). These competition problems require long chains of tightly coupled reasoning steps. For competitive programming, we use LiveCodeBench v6 (Jain et al., 2025), which consists of recent problems from LeetCode, AtCoder, and Codeforces, and we report pass@1 on the hidden test cases.

Baselines. We compare against zero-shot generation, majority voting, MCTS (Kocsis & Szepesvari, 2006) + PRM (Lightman et al., 2024). The PRM is instantiated by prompting the same language model used for answer generation.

Setup. Since AIME solutions are tightly coupled, the evolution operators act on complete solutions through prompting. The backward search asks the model to write executable self-check predicates for the problem. Detailed configurations are provided in Appendix E.4. Similarly, for competitive programming, candidates are complete programs, and all four evolution operators are implemented by prompting. The backward search decomposes the problem into executable program checks. Detailed configurations are provided in Appendix E.5.

As shown in Table 3, most search methods substantially improve over zero-shot generation. BES achieves the best accuracy on average.

4.3 Ablation Study

We conduct an ablation study on the Knights-and-Knaves benchmark. BES consists of two components: the evolution operators in the forward search and the backward search. We therefore consider two ablations: (1) removing the evolution operators; and (2) removing the backward search. As shown in Figure 4, removing the evolution operators degrades performance relative to the full BES, while removing the backward search performs close to or even worse than the MaxRL baseline. Without the verification provided by the backward search, the forward search reduces to random exploration, which shows that the components of BES are complementary.

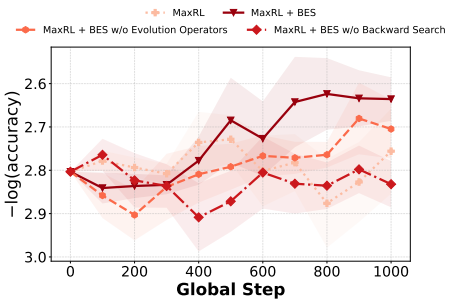

Figure 4: Ablation study on logical reasoning.

4.4 Cost Analysis

To evaluate the cost-effectiveness of BES, we compare the per-step wall-clock time and accuracy across GRPO, Tree-GRPO, GRPO + MCTS, and GRPO + BES when training Llama-3.2-3B-Instruct on the agentic search task. We report the median wall-clock time per step throughout the training.

Table 4: Cost analysis for multi-hop reasoning post-training on Llama-3.2-3B-Instruct. Accuracy (%) on MuSiQue and Bamboogle is reported from Table 1. # Valid Search is the average number of valid <search> calls per question by the agent. Walltime is the median wall-clock time per training step.

Method	MuSiQue Acc. (%)	Bamboogle Acc. (%)	# Valid Search	Walltime (s)
Base model	4.01 ± 0.40	24.0 ± 3.8	1.62	–
GRPO	2.11 ± 0.29	16.8 ± 3.3	0.84	64
Tree-GRPO	3.93 ± 0.40	23.2 ± 3.8	1.50	240
GRPO + MCTS	4.30 ± 0.41	18.4 ± 3.5	1.88	382
GRPO + BES (ours)	**9.08** ± 0.58	**30.4** ± 4.1	**2.31**	309

Search is more expensive than i.i.d. rollouts: BES takes about $4.8\times$ the per-step time of GRPO. However, GRPO wastes most of its compute, since 93% of its groups carry no learning signal. Among the search-based samplers, BES takes 19% less time per step than GRPO + MCTS while more than doubling its MuSiQue accuracy (9.08% vs. 4.30%) and improving Bamboogle accuracy by 12.0 points. Compared with Tree-GRPO, BES adds 29% to the per-step time but improves MuSiQue accuracy by $2.3\times$ and Bamboogle accuracy by 7.2 points. Thus, the gains of BES cannot be attributed to extra compute alone. Instead, they stem from how the search constructs and selects candidates.

5 Related Work

Self-Improvement for LLM and Agent. A growing line of work studies how language models and agents can improve using their own outputs to self-evolve (Huang et al., 2025; Kadlčík & Štefánik, 2024). STaR (Zelikman et al., 2022) bootstraps reasoning by filtering correct rationales for fine-tuning, while Self-Refine (Madaan et al., 2023) revises outputs at inference time. Self-Rewarding Language Models (Yuan et al., 2024) use the model itself as a judge for preference optimization. Reflexion (Shinn et al., 2023) converts environmental feedback into verbal reflections, and Voyager (Wang et al., 2024) accumulates reusable skills through continual exploration. These methods typically refine individual trajectories or rely on the model to judge its outputs. In contrast, BES treats self-improvement as a search problem, discovering solutions that facilitate self-improvement.

Classical Search Methods. The recent surge of search methods in LLMs draws on a long history of classical search. In graph search, algorithms such as A* (Hart et al., 1968) and bidirectional search introduced principles including heuristic guidance, frontier expansion, and search-space reduction. Branch-and-bound methods (Lawler & Wood, 1966), which prune subtrees whose bounds certify suboptimality, offer a natural analogy for methods that use verifiers to discard unpromising candidates. Evolutionary methods, including genetic search (Holland, 1992; Banzhaf et al., 2000) and differential evolution (Storn & Price, 1997), optimize by iteratively refining candidate populations.

Additional related work is discussed in Appendix A.

6 Conclusion

In this paper, we present BES, a bidirectional evolutionary search framework that addresses two fundamental limitations of existing search methods for LLMs and agents: sparse verification signals and confined candidate generation. BES couples forward search, which evolves candidates through expansion and evolution operators, with backward search, which recursively decomposes problems into verifiable sub-goals to provide dense feedback. We illustrate that candidates generated by expansion-only search may be confined to a narrow entropy shell while evolution operators may escape it. Experiments demonstrate that BES consistently improves over baselines in both post-training and inference. We hope BES provides useful insights for self-improving language models.

AI USE STATEMENT

In this work, we used generative AI tools to assist in checking the correctness of the derivations, to assist with grammar corrections, and we used GPT-5.6-luna as an annotator in Appendix J. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, or interpret results, and generating synthetic data sets and cleaning or reformatting data sets are not applicable to this work. We have reviewed all AI-assisted work. AI-assisted modifications were reviewed and edited by the authors to ensure they faithfully convey the intended meaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work proposes a general-purpose search framework for improving the quality of LLM and agent outputs. On the positive side, BES can help models achieve stronger reasoning performance, potentially reducing the need for larger models and their associated computational and environmental costs. The backward search component also improves interpretability by explicitly decomposing problems into verifiable sub-goals, making the search process more transparent. However, more effective search methods could also enable stronger performance on tasks that could be misused.

REPRODUCIBILITY STATEMENT

We provide detailed information to facilitate reproduction of our results. The main BES algorithm and its forward and backward search procedures are specified in pseudocode. Appendix gives the assumptions and derivations underlying the motivating analysis of shell confinement and escape. Complete experimental configurations are provided, including datasets, model backbones, and other hyperparameters for each task.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r. 2, 7

W. Banzhaf, J.R. Koza, C. Ryan, L. Spector, and C. Jacob. Genetic programming. *IEEE Intelligent Systems and their Applications*, 15(3):74–84, 2000. doi: 10.1109/5254.846288. 9

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720. 20

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787. 2

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang,

Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348. 32

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=9ZxnPZGmPU. 20

Ronald Aylmer Fisher. *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press, 1999. 2

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai, 2025. URL https://arxiv.org/abs/2411.04872. 1

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu,

Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763. 32

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao

Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783. 6

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.

net/forum?id=5zwF1GizFa. 1, 6, 20

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z. 1, 2, 6, 7

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZG3RaNIsO8. 20

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/. 20

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136. 9

John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 04 1992. ISBN 9780262275552. doi: 10.7551/mitpress/1090.001.0001. URL https://doi.org/10.7551/mitpress/1090.001.0001. 9

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12355–12369, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.604. URL https://aclanthology.org/2025.acl-long.604/. 20

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WJaUkwci9o. 9

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, Mar 2026. ISSN 1476-4687. doi: 10.1038/s41586-025-09833-y. URL https://doi.org/10.1038/s41586-025-09833-y. 1

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL. 2, 8

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZpQwAFhU13. 1, 2, 3, 6, 20

Marek Kadlčík and Michal Štefánik. Self-training language models for arithmetic reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12378–12386, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.721. URL https://aclanthology.org/2024.findings-emnlp.721/. 9

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=FPVCb0WMuN. J2C Certification. 20

Levente Kocsis and Csaba Szepesvari. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, 2006. URL https://api.semanticscholar.org/CorpusID:15184765. 1, 2, 3, 8

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution, 2025. URL https://arxiv.org/abs/2509.19349. 1, 2, 7, 20

Eugene L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Oper. Res.*, 14:699–719, 1966. URL https://api.semanticscholar.org/CorpusID:36099120. 9

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking, 2025. URL https://arxiv.org/abs/2501.09891. 20

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158. 1

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi. 8, 20, 33

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB. 9

Elliot Meyerson, Mark J. Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K. Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting. *ACM Trans. Evol. Learn. Optim.*, 4(4), November 2024. ISSN 2688-299X. doi: 10.1145/3694791. URL https://doi.org/10.1145/3694791. 20

Kou Misaki, Yuichi Inoue, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling LLM inference-time compute with adaptive branching tree search. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. URL https://openreview.net/forum?id=3HF6yogDEm. 20

Hermann Joseph Muller. Some genetic aspects of sex. *The American Naturalist*, 66(703):118–138, 1932. 2

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131. 1, 8, 20

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. Self-improving language models for evolutionary program synthesis: A case study on arc-agi, 2026. URL https://arxiv.org/abs/2507.14172. 20

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/. 6

Zhenting Qi, Mingyuan MA, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller LLMs stronger problem-solver. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6aHUmotXaw. 1, 6

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. 20

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve. 2, 7

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6. 9

Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, December 1997. ISSN 0925-5001. doi: 10.1023/A:1008202821328. URL https://doi.org/10.1023/A:1008202821328. 9

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning, 2026. URL https://arxiv.org/abs/2602.02710. 2, 7

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786. 7

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/. 6

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a. 9

Haorui Wang, Marta Skreta, Cher Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=awWiNvQwf3. 20

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Thetaevolve: Test-time learning on open problems, 2025b. URL https://arxiv.org/abs/2511.23473. 20

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. Large language interpolators can learn logical reasoning: A study

on knights and knaves puzzles. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024a. URL https://openreview.net/forum?id=mxX8WdPCx9. 7

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*, 2024b. URL https://openreview.net/forum?id=s004OmYP2P. 20

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h. 1, 3, 20

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1354. URL https://aclanthology.org/2025.acl-long.1354/. 20

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. 1

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=0NphYCmgua. 1, 9

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=4OsgYD7em5. 1

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175. 1

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI. 1, 9

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=46Zgqo4QIU. 1

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=8rcFOqEud5. 20

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pUpzQZTvGY. 20

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25. 2, 8

APPENDIX - TABLE OF CONTENTS

A Additional Related Work **20**

B Pseudo Code **20**

C Formal Definitions of Evolution Operators **21**

D Intuition for Evolution Operators **22**

D.1 Discussion of Assumptions 22

D.2 Shell Confinement of Expansion 23

D.3 Shell Escape via Evolution 25

E Detailed Experimental Setup **28**

E.1 Multi-Hop Reasoning 28

E.2 Logical Reasoning 29

E.3 Open Problem Solving 29

E.4 Mathematical Reasoning 30

E.5 Competitive Programming 30

F Case Study **31**

G Open Problem Solving with Open-Weight Models **32**

H Hyperparameter Sensitivity Analysis **33**

I Empirical Verification of Shell Confinement and Escape **33**

J Quality of the Embedding-Based Verifier on MuSiQue **33**

K Potential Limitations **34**

A Additional Related Work

Search in LLM and Agent. Recent work applies search to both training and inference of LLMs and agents (Wang et al., 2025a; Romera-Paredes et al., 2024). On the training side, tree-structured exploration provides richer supervision and finer-grained credit assignment than standard rollout-based methods. Tree-GRPO (Ji et al., 2026) and TreeRL (Hou et al., 2025) integrate tree search directly into reinforcement learning, while ReST-MCTS* (Zhang et al., 2024), MCTS-DPO (Xie et al., 2024b), and rStar-Math (Guan et al., 2025) use search to bootstrap higher-quality training data for self-improvement. On the inference side, Tree of Thoughts (Yao et al., 2023), Graph of Thoughts (Besta et al., 2024), and RAP (Hao et al., 2023) established reasoning as an explicit search problem by expanding chain-of-thought into tree-structured exploration (Misaki et al., 2025). Since these methods build each candidate by extending a single lineage, they struggle to explore beyond the policy's own distribution. A complementary line of work uses LLMs in evolutionary search. Language Model Crossover (Meyerson et al., 2024) prompts a model with several parents to produce offspring, EvoPrompt (Guo et al., 2024) and Promptbreeder (Fernando et al., 2024) evolve prompts through LLM-driven crossover, and Mind Evolution (Lee et al., 2025) recombines and refines natural-language solutions at inference time. Evolutionary frameworks such as AlphaEvolve (Novikov et al., 2025), ShinkaEvolve (Lange et al., 2025), and ThetaEvolve (Wang et al., 2025b) further tackle hard and even open problems (Pourcel et al., 2026). Besides this, Gödel Agent (Yin et al., 2025) and the Darwin Gödel Machine (Zhang et al., 2026) achieve evolution by letting an agent rewrite its own code or scaffold, i.e., they search over agent designs. However, these methods guide selection with a task-level fitness function and are mainly used at inference time. Process reward models (PRMs) provide intermediate feedback by learning to evaluate individual reasoning steps (Lightman et al., 2024); generative PRMs such as ThinkPRM additionally produce reasoning to verify these steps (Khalifa et al., 2026). In BES, backward search generates executable sub-goal checks that can be reused across candidates, eliminating the need to train a separate PRM.

B Pseudo Code

We provide the full pseudocode for BES below. Algorithm 1 describes the main loop, which alternates between forward search steps (Algorithm 2) and backward scoring (Algorithm 3). Every K_{dec} steps, the backward goal tree is refined by decomposing an unsolved leaf into finer sub-goals (Algorithm 4), after which all pool scores are recomputed under the updated tree. When the task provides no ground-truth verifier, the root-goal verifier is written by the policy.

Algorithm 1 BES: Bidirectional Evolutionary Search

Require: problem x, policy π_θ, verifier V, budget B, decomposition interval K_{dec}
Initialize $\mathcal{G}$ with root goal g_0; prompt π_θ to decompose g_0 into initial sub-goals
$\mathcal{P} \leftarrow \{n_0\}$ with $n_0 = ()$ ▷ Empty root node
for $u = 0, 1, \ldots$ **while** calls$(u) < B$ **do**
  $n' \leftarrow \textsc{ForwardStep}(\mathcal{P}, \mathcal{C}_u, \mathcal{G}, \tau_u)$ ▷ Algorithm 2
  $s(n') \leftarrow \textsc{BackwardScore}(n', g_0, \mathcal{G})$ ▷ Algorithm 3
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{n'\}$
  if $(u+1) \bmod K_{\text{dec}} = 0$ **then**
    $\mathcal{G} \leftarrow \textsc{BackwardDecompose}(x, \mathcal{G}, \mathcal{P})$ ▷ Algorithm 4
    for $n \in \mathcal{P}$ **do** ▷ Recompute all scores under refined tree
      $s(n) \leftarrow \textsc{BackwardScore}(n, g_0, \mathcal{G})$
    end for
  end if
end for
return $\arg\max_{n \in \mathcal{P}:\, \text{terminal}(n)} V(x, n)$

Algorithm 2 details a single forward search step. An operator is sampled from the fixed distribution over the five operators. In Apply, operators are realized either as direct edits or by prompting the policy.

Algorithm 2 FORWARDSTEP: Forward Search Step

Require: pool $\mathcal{P}$, eligible set $\mathcal{C}_u$, goal tree $\mathcal{G}$, temperature τ_u
Sample operator $o \in \{\textit{expand}, \textit{combine}, \textit{delete}, \textit{translocate}, \textit{crossover}\}$
if $o \in \{\textit{expand}, \textit{delete}\}$ **then** ▷ Single-parent operators
  Sample parent n from $\mathcal{C}_u$ via Boltzmann selection (Eq. 2)
  $n' \leftarrow \textsc{Apply}(o, n)$
else ▷ Two-parent operators
  Sample pair (n_a, n_b) from $\mathcal{C}_u$ via pair Boltzmann selection (Eq. 3)
  $n' \leftarrow \textsc{Apply}(o, n_a, n_b)$
end if
return n'

Algorithm 3 computes the backward score for a candidate node by recursively traversing the sub-goal tree. If a sub-goal is fully satisfied, the score is 1; otherwise, the score blends the local verifier output with the average score over its children.

Algorithm 3 BACKWARDSCORE: Backward Search Scoring

Require: problem x, candidate node n, sub-goal g, goal tree $\mathcal{G}$
if $V_g(x, n) = 1$ **then**
  return 1
else if $\mathrm{ch}(g) \neq \emptyset$ **then**
  return $\alpha \cdot V_g(x, n) + (1 - \alpha) \cdot \frac{1}{|\mathrm{ch}(g)|} \sum_{g' \in \mathrm{ch}(g)} \textsc{BackwardScore}(n, g', \mathcal{G})$
else
  return $V_g(x, n)$
end if

Algorithm 4 refines the goal tree by selecting a random unsolved leaf sub-goal and prompting the policy to decompose it into finer children, each equipped with its own local verifier.

Algorithm 4 BACKWARDDECOMPOSE: Goal Tree Decomposition

Require: problem x, current goal tree $\mathcal{G}$, pool $\mathcal{P}$
$\mathcal{L} \leftarrow \{g \in \mathcal{G} : \mathrm{ch}(g) = \emptyset \wedge \max_{n \in \mathcal{P}} V_g(x, n) < 1\}$ ▷ Unsolved leaves
Sample g^* uniformly from $\mathcal{L}$
Prompt π_θ to decompose g^* into children $\{g'_1, \ldots, g'_c\}$ with local verifiers $\{V_{g'_1}, \ldots, V_{g'_c}\}$
$\mathrm{ch}(g^*) \leftarrow \{g'_1, \ldots, g'_c\}$; add to $\mathcal{G}$
return $\mathcal{G}$

C FORMAL DEFINITIONS OF EVOLUTION OPERATORS

The definitions below are for the direct-edit instantiation of the evolution operators, in which a candidate is a sequence of self-contained steps with clear boundaries (e.g., paragraph-level reasoning steps in Knights-and-Knaves and action triples in MuSiQue). When direct edits are not meaningful, e.g., for executable programs or tightly coupled derivations, the same operators are instantiated by prompting the policy model.

For two parents n_a, n_b, let $s = \max\{\ell : y_{a,1:\ell} = y_{b,1:\ell}\}$ denote their common-prefix length, and let $\sigma_a = (y_{a,s+1}, \ldots, y_{a,t_a})$ and $\sigma_b = (y_{b,s+1}, \ldots, y_{b,t_b})$ be their respective suffixes.

(i) Combination. $n' = y_{a,1:s} \oplus \sigma_a \oplus \sigma_b$.

(ii) Deletion. $n' = (y_1, \ldots, y_{\ell-1}, y_{\ell+1}, \ldots, y_t)$, where $\ell \sim \mathrm{Uniform}\{2, \ldots, t-1\}$.

(iii) Translocation. $n' = y_{a,1:s} \oplus \sigma_a[1 : r-1] \oplus (\sigma_b)_q \oplus \sigma_a[r+1 : m_a]$, where $r \sim \mathrm{Uniform}\{1, \ldots, m_a\}$ and $q \sim \mathrm{Uniform}\{1, \ldots, m_b\}$.

(iv) Crossover. $n' = y_{a,1:s} \oplus \sigma_a[1 : i] \oplus \sigma_b[j+1 : m_b]$, where $i \in \{0, \ldots, m_a\}$ and $j \in \{0, \ldots, m_b - 1\}$ are sampled uniformly.

D Intuition for Evolution Operators

In this section, we provide an intuition for the evolution operators. This is not a formal theory.

Expansion generates a continuation conditioned on an existing prefix, whereas evolution assembles blocks from different trajectories. For a fixed partition of trajectory positions, independently selecting a block from each corresponding library yields a Cartesian product of block choices.

Fix a task x and a positive integer horizon T. We use token-level steps in the analysis: each step takes values in a finite vocabulary $\mathcal{V}$. Let $Y = (Y_1, \ldots, Y_T)$ denote a random trajectory and $y = (y_1, \ldots, y_T)$ a realization. For each position $t \in \{1, \ldots, T\}$, write $Y_{<t} = (Y_1, \ldots, Y_{t-1})$ for the random prefix and $y_{<t}$ for its realization; the prefix at $t = 1$ is empty. The policy π_θ, parameterized by θ, induces the trajectory distribution

$$P(y) := \pi_\theta(y \mid x) = \prod_{t=1}^{T} P(y_t \mid y_{<t}). \tag{6}$$

Here, $P(y_t \mid y_{<t})$ is the policy's next-token probability. We suppress conditioning on the fixed task x throughout. All logarithms are natural, so information quantities are measured in nats.

To describe next-token uncertainty, define the *surprisal* of the generated token and the *conditional entropy* at a given prefix:

$$Z_t := -\log P(Y_t \mid Y_{<t}), \qquad h_t(y_{<t}) := H_P(Y_t \mid Y_{<t} = y_{<t}), \tag{7}$$

where H_P denotes Shannon entropy under P. Thus, Z_t measures how surprising the sampled token is, whereas $h_t(y_{<t})$ is its expected surprisal when the prefix is fixed. We also define the position-wise mean conditional entropy

$$\bar{h}_t := \mathbb{E}_P[h_t(Y_{<t})]\,, \tag{8}$$

where $\mathbb{E}_P$ denotes expectation under the native trajectory distribution P. Unlike $h_t(Y_{<t})$, which varies with the sampled prefix, $\bar{h}_t$ is deterministic for a fixed task and position.

The total surprisal of a trajectory is

$$S_T := -\log P(Y) = \sum_{t=1}^{T} Z_t. \tag{9}$$

Its expectation is the trajectory entropy:

$$H_T := H_P(Y) = \mathbb{E}_P[S_T] = \sum_{t=1}^{T} \mathbb{E}_P[h_t(Y_{<t})] = \sum_{t=1}^{T} \bar{h}_t, \tag{10}$$

where the decomposition follows from the entropy chain rule. In particular, H_T is a fixed center determined by the task, policy, and horizon.

Finally, let $\mathcal{F}_t = \sigma(Y_1, \ldots, Y_t)$ be the natural filtration, where $\sigma(\cdot)$ denotes the sigma-algebra generated by the listed random variables. Thus, $\mathcal{F}_t$ represents the information available after the first t tokens, and $\mathcal{F}_0$ is the trivial sigma-algebra. By construction,

$$\mathbb{E}_P[Z_t \mid \mathcal{F}_{t-1}] = h_t(Y_{<t}). \tag{11}$$

Consequently, $Z_t - h_t(Y_{<t})$ is a martingale difference. This identity separates token-sampling fluctuations from prefix-dependent fluctuations in conditional entropy and forms the basis of the concentration analysis below.

D.1 Discussion of Assumptions

We make two assumptions about the process.

Assumption D.1 (Local context correlation)**.** *There exists a constant $\rho > 0$ such that, for every $t \in \{2, \ldots, T\}$,*

$$I_P(Y_t; Y_{t-1}) \geq \rho, \tag{12}$$

where I_P denotes mutual information under P.

This requires the preceding token to provide predictive information about the current token: observing Y_{t-1} reduces the uncertainty about Y_t by at least ρ nats on average. It captures the local sequential structure of language and reasoning, where nearby tokens are connected through grammatical constraints, semantic relations, and the progression of an argument. The constant ρ quantifies the minimum strength of this local dependence.

Assumption D.2 (Limited next-token uncertainty)**.** *There exists a constant $\kappa \geq 0$ such that, for every $t \in \{2, \ldots, T\}$ and every prefix $y_{<t}$ with $P(y_{<t}) > 0$,*

$$h_t(y_{<t}) \leq \kappa. \tag{13}$$

This requires the policy's next-token uncertainty to remain bounded after observing the full prefix. It formalizes the intuition that context narrows the range of plausible continuations, with κ controlling the remaining uncertainty. Different prefixes may favor entirely different tokens and induce different probability distributions while satisfying the same entropy bound. Together, the two assumptions describe a generation process in which context informs the prediction and the uncertainty remaining after conditioning on that context is limited.

D.2 Shell Confinement of Expansion

The deviation $S_T - H_T$ between a trajectory's actual information content and its expected entropy governs how tightly ordinary rollouts cluster around the typical set. We decompose this deviation into token-sampling fluctuations and conditional-entropy drift, and bound each separately.

Observation D.3 (Martingale decomposition)**.** *Define*

$$D_t := Z_t - h_t(Y_{<t}), \qquad M_T := \sum_{t=1}^{T} D_t, \qquad R_T := \sum_{t=1}^{T} \big(h_t(Y_{<t}) - \bar{h}_t\big), \tag{14}$$

where $\bar{h}_t = \mathbb{E}_P[h_t(Y_{<t})]$. Then $(D_t, \mathcal{F}_t)$ is a martingale difference sequence, and

$$S_T - H_T = M_T + R_T. \tag{15}$$

Here M_T captures token-sampling fluctuations, while R_T captures the accumulated deviation of conditional entropy from its position-wise mean.

Derivation. By definition,

$$\mathbb{E}_P[Z_t \mid \mathcal{F}_{t-1}] = h_t(Y_{<t}),$$

so $\mathbb{E}_P[D_t \mid \mathcal{F}_{t-1}] = 0$. Moreover,

$$S_T = \sum_{t=1}^{T} D_t + \sum_{t=1}^{T} h_t(Y_{<t}), \qquad H_T = \sum_{t=1}^{T} \bar{h}_t.$$

Subtracting these identities gives Eq. equation 15. □

Observation D.4 (Bound on entropy drift)**.** *Under Assumption D.2,*

$$|R_T| \leq (T-1)\kappa \qquad P\text{-almost surely}. \tag{16}$$

Derivation. The first-step prefix is fixed, so $h_1(Y_{<1}) = \bar{h}_1$. For every $t \geq 2$, Assumption D.2 gives

$$0 \leq h_t(Y_{<t}) \leq \kappa, \qquad 0 \leq \bar{h}_t \leq \kappa.$$

Consequently,

$$|R_T| \leq \sum_{t=2}^{T} |h_t(Y_{<t}) - \bar{h}_t| \leq (T-1)\kappa.$$

□

Observation D.5 (Concentration of M_T)**.** *Let*

$$L := 1 + \log |\mathcal{V}|. \tag{17}$$

Then, for every $u > 0$,

$$\Pr(|M_T| \geq u) \leq 2 \exp\left(-\frac{u^2}{4TL^2 + 2Lu} \right). \tag{18}$$

Derivation. For every $r \geq 0$, the finite vocabulary gives

$$\Pr(Z_t \geq \log|\mathcal{V}| + r \,|\, \mathcal{F}_{t-1}) \leq e^{-r}. \tag{19}$$

Indeed, this event contains at most $|\mathcal{V}|$ tokens, each with conditional probability at most $e^{-r}/|\mathcal{V}|$. Since $0 \leq h_t(Y_{<t}) \leq \log|\mathcal{V}|$, we have

$$|D_t| \leq \max\{Z_t, \log|\mathcal{V}|\}.$$

The same tail bound therefore holds with Z_t replaced by $|D_t|$. Integrating this bound gives, for every integer $q \geq 2$,

$$\mathbb{E}_P[|D_t|^q \mid \mathcal{F}_{t-1}] \leq q! \sum_{j=0}^{q} \frac{(\log|\mathcal{V}|)^j}{j!} \leq q! L^q. \tag{20}$$

Using $\mathbb{E}_P[D_t \mid \mathcal{F}_{t-1}] = 0$, for every real λ with $|\lambda| < 1/L$,

$$\mathbb{E}_P[e^{\lambda D_t} \mid \mathcal{F}_{t-1}] \leq 1 + \sum_{q=2}^{\infty} (|\lambda| L)^q \tag{21}$$

$$\leq \exp\left(\frac{\lambda^2 L^2}{1 - |\lambda| L}\right). \tag{22}$$

Iterating conditional expectations yields

$$\mathbb{E}_P[e^{\lambda M_T}] \leq \exp\left(\frac{T\lambda^2 L^2}{1 - |\lambda| L}\right). \tag{23}$$

Applying the exponential Markov inequality with

$$\lambda = \frac{u}{2TL^2 + Lu}$$

gives

$$\Pr(M_T \geq u) \leq \exp\left(-\frac{u^2}{4TL^2 + 2Lu}\right).$$

Applying the same argument to $-M_T$ and combining the two tails gives Eq. equation 18. □

Combining the entropy-drift bound with the martingale concentration bound yields exponential confinement around the fixed trajectory entropy H_T.

Observation D.6 (Finite-sample concentration for policy rollouts)**.** *Under Assumption D.2, for every* $u > (T-1)\kappa$*,*

$$\Pr(|S_T - H_T| \geq u) \leq 2\exp\left(-\frac{\big(u - (T-1)\kappa\big)^2}{4TL^2 + 2L\big(u - (T-1)\kappa\big)}\right), \tag{24}$$

where L is defined in Observation D.5. In particular, for a fixed vocabulary, fixed κ, and every fixed $\epsilon > \kappa$,

$$\Pr\left(\left|\frac{S_T}{T} - \frac{H_T}{T}\right| \geq \epsilon\right) \leq \exp(-\Omega(T)). \tag{25}$$

Derivation. By Observation D.3, $S_T - H_T = M_T + R_T$. Observation D.4 therefore gives

$$\Pr(|S_T - H_T| \geq u) \leq \Pr(|M_T| \geq u - (T-1)\kappa).$$

Applying Observation D.5 gives Eq. equation 24. For $u = \epsilon T$ with fixed $\epsilon > \kappa$,

$$u - (T-1)\kappa = (\epsilon - \kappa)T + \kappa.$$

The numerator in the exponent is then of order T^2, while the denominator is of order T, giving the exponential concentration rate. □

Observation D.7 (Shell confinement)**.** *For a fixed $\epsilon > \kappa$, define the typical set*

$$A_\epsilon^{(T)} := \{y : |-\log P(y) - H_T| \le \epsilon T\}. \tag{26}$$

Under Observation D.6,

$$P(A_\epsilon^{(T)}) \ge 1 - \exp(-\Omega(T)).$$

Moreover,

$$|A_\epsilon^{(T)}| \le \exp(H_T + \epsilon T). \tag{27}$$

Thus, expansion through native policy rollouts remains within this shell with exponentially high probability.

Derivation. The probability statement follows from Observation D.6. Every $y \in A_\epsilon^{(T)}$ satisfies

$$P(y) \ge \exp\big(-(H_T + \epsilon T)\big).$$

Summing probabilities over the typical set gives

$$1 \ge |A_\epsilon^{(T)}| \exp\big(-(H_T + \epsilon T)\big),$$

which gives Eq. equation 27. □

D.3 Shell Escape via Evolution

We first relate the four evolution operators to a common recombination model. The reductions use independent native policy rollouts as donors, fixed edit locations, and transferred tokens placed at their original positions. We use the convention $-\log 0 = +\infty$.

Partition the coordinates of a trajectory Y into $k \ge 2$ nonempty, disjoint groups $U_1, \dots, U_k$ that together cover all T positions. Each group records its tokens in their original order and may contain several separated fragments. Let P_j be the native marginal distribution of U_j. A k-way evolution takes group j from an independent donor and returns its tokens to their assigned positions. Identifying the grouped coordinates with the full trajectory, its law is

$$Q = \bigotimes_{j=1}^{k} P_j. \tag{28}$$

Thus, k counts independent donor groups.

(i) Combination. Combination appends σ_b to $y_{a,1:s} \oplus \sigma_a$, giving a concatenation of two parent fragments. This moves the appended continuation to new positions and changes the trajectory length. We use combination as an intuitive extension of the recombination mechanism and do not establish an escape guarantee for its general variable-length form.

(ii) Deletion. Deletion removes a paragraph and reconnects the retained fragments. It motivates exploration through changes to the context, but is not included in the illustration.

(iii) Translocation. For translocation at a fixed position $t \in \{2, \dots, T\}$, retain all coordinates except Y_t from one donor and replace Y_t with the same-position token from another independent donor. Writing Y_{-t} for the subvector containing all coordinates except t, this gives

$$U_1 = Y_{-t}, \qquad U_2 = Y_t, \qquad Q = P_{Y_{-t}} \otimes P_{Y_t}. \tag{29}$$

The retained left and right fragments together form one group. Consequently, this is a $k = 2$ construction, with

$$H(Q) - H_T = I_P(Y_t; Y_{-t}) \ge \rho. \tag{30}$$

Replacing $k-1$ distinct positions using fresh independent donors similarly produces k groups: one retained group and $k-1$ singleton groups. Transfers between different positions are treated as intuitive extensions here.

(iv) Crossover. At a fixed, aligned cut $s \in \{1, \dots, T-1\}$, crossover takes $Y_{1:s}$ from one donor and $Y_{s+1:T}$ from another. It therefore gives

$$U_1 = Y_{1:s}, \qquad U_2 = Y_{s+1:T}, \qquad Q = P_{Y_{1:s}} \otimes P_{Y_{s+1:T}}, \tag{31}$$

with

$$H(Q) - H_T = I_P(Y_{1:s}; Y_{s+1:T}) \geq \rho. \tag{32}$$

Successive crossovers at fixed boundaries $0 = s_0 < s_1 < \cdots < s_k = T$, each introducing a fresh independent donor, yield the contiguous groups $U_j = Y_{s_{j-1}+1:s_j}$ and hence the k-way product law in Eq. equation 28.

We now illustrate these information gains and the resulting shell escape, beginning with a single crossover and then extending to independent donor groups.

Observation D.8 (Splice-point KL identity)**.** *Split a trajectory at position $s \in \{1, \ldots, T-1\}$ and write $V = Y_{1:s}$ and $U = Y_{s+1:T}$. Let (V, U) and (V', U') be two independent policy rollouts, and form the crossover trajectory $\widetilde{Y} = (V, U')$. The expected increase in native surprisal is*

$$\mathbb{E}[-\log P(V, U')] - H_T = \mathbb{E}_{V,V'}\left[D_{\mathrm{KL}}\big(P_{U|V'} \| P_{U|V}\big)\right]. \tag{33}$$

Equivalently,

$$\mathbb{E}[-\log P(V, U')] - H_T = I_P(V; U) + D_{\mathrm{KL}}\big(P_V \otimes P_U \| P_{V,U}\big) \geq I_P(V; U). \tag{34}$$

The gap is strictly positive whenever the suffix distribution depends on the prefix. Under Assumption D.1, the gap is at least ρ.

Derivation. Using $P(V, U) = P_V(V) P_{U|V}(U)$,

$$\mathbb{E}[-\log P(V, U')] = H_P(V) + \mathbb{E}_{V,V'}\mathbb{E}_{U \sim P_{U|V'}}[-\log P_{U|V}(U)], \tag{35}$$

$$H_T = H_P(V) + \mathbb{E}_{V'} H(P_{U|V'}). \tag{36}$$

Subtracting gives Eq. equation 33.

The marginal law of (V, U') is $P_V \otimes P_U$, so its expected native surprisal is

$$H(P_V \otimes P_U) + D_{\mathrm{KL}}\big(P_V \otimes P_U \| P_{V,U}\big).$$

Since

$$H(P_V \otimes P_U) - H_T = I_P(V; U),$$

Eq. equation 34 follows. Finally, data processing and Assumption D.1 give

$$I_P(V; U) \geq I_P(Y_s; Y_{s+1}) \geq \rho.$$

□

Observation D.8 analyzes crossover at a single splice point. We now generalize to k-way block evolution, where each block is drawn from an independent seed trajectory.

Observation D.9 (Block evolution cross-entropy gap)**.** *For $\widetilde{Y} \sim Q$,*

$$\mathbb{E}_Q[-\log P(\widetilde{Y})] - H_T = \mathrm{TC}_P(U_1, \ldots, U_k) + D_{\mathrm{KL}}(Q \| P) \geq \mathrm{TC}_P(U_1, \ldots, U_k), \tag{37}$$

where

$$\mathrm{TC}_P(U_1, \ldots, U_k) := \sum_{j=1}^{k} H_P(U_j) - H_T$$

is the total correlation between the donor groups. Under Assumption D.1,

$$\mathrm{TC}_P(U_1, \ldots, U_k) \geq (k-1)\rho. \tag{38}$$

Derivation. Since Q is the product of the native group marginals,

$$H(Q) = \sum_{j=1}^{k} H_P(U_j).$$

The cross-entropy identity

$$\mathbb{E}_Q[-\log P(\widetilde{Y})] = H(Q) + D_{\mathrm{KL}}(Q \| P)$$

therefore gives Eq. equation 37.

To derive Eq. equation 38, order the groups by their first token positions. Let t_j be the first position belonging to U_j, so

$$1 = t_1 < t_2 < \cdots < t_k \leq T.$$

For every $j \geq 2$, position $t_j - 1$ belongs to an earlier group, because its group's first position is smaller than t_j. Thus, writing $U_{<j} = (U_1, \ldots, U_{j-1})$, data processing gives

$$I_P(U_j; U_{<j}) \geq I_P(Y_{t_j}; Y_{t_j-1}) \geq \rho.$$

The entropy chain rule then yields

$$\mathrm{TC}_P(U_1, \ldots, U_k) = \sum_{j=2}^{k} I_P(U_j; U_{<j}) \geq (k-1)\rho. \tag{39}$$

□

Putting these together gives the following illustration.

Motivation 3.1 (Shell confinement and escape)**.**

Under Assumptions D.1 and D.2, fix $\epsilon > \kappa$ and define the shell

$$A_\epsilon^{(T)} := \{y : |-\log P(y) - H_T| \leq \epsilon T\}.$$

(a) Generation and tree search stay within the shell. For a trajectory $Y \sim P$ generated by the policy,

$$\Pr[Y \notin A_\epsilon^{(T)}] \leq \exp(-\Omega(T)).$$

This shell contains at most $\exp(H_T + \epsilon T)$ trajectories.

(b) Evolution Operators can escape the shell. Construct $\widetilde{Y}$ by cutting independent policy rollouts at the same m fixed positions and joining the resulting $m+1$ segments, taking each segment from a different rollout at its original positions. The m joins yield an expected surprisal increase of at least $m\rho$:

$$\mathbb{E}[-\log P(\widetilde{Y})] \geq H_T + m\rho. \tag{40}$$

Whenever $m\rho > \epsilon T + \log 2$, the expected surprisal exceeds the upper shell boundary, and

$$\Pr[\widetilde{Y} \notin A_\epsilon^{(T)}] \geq \frac{m\rho - \epsilon T - \log 2}{T \log |\mathcal{V}| - H_T - \epsilon T} > 0. \tag{41}$$

If $m\rho/T$ exceeds ϵ by a fixed positive margin as T grows, the recombined candidate escapes with probability bounded below by a positive constant, whereas a native rollout escapes only with exponentially small probability.

Derivation. Part (a) follows from Observation D.7, since $\epsilon > \kappa$.

For part (b), let $U_1, \ldots, U_{m+1}$ denote the segments of a native trajectory at the chosen cut positions. The recombined trajectory $\widetilde{Y}$ contains independent segments with these same marginal distributions. Hence, by Observation D.9,

$$H(\widetilde{Y}) = \sum_{j=1}^{m+1} H_P(U_j) \geq H_T + m\rho, \tag{42}$$

where $H(\widetilde{Y})$ denotes the entropy of the recombined candidate. The cross-entropy inequality gives

$$\mathbb{E}[-\log P(\widetilde{Y})] \geq H(\widetilde{Y}) \geq H_T + m\rho.$$

Let

$$p := \Pr[\widetilde{Y} \notin A_\epsilon^{(T)}].$$

Splitting the entropy according to shell membership contributes at most $\log 2$ nats for the membership indicator. Inside the shell, the conditional entropy is at most $\log |A_\epsilon^{(T)}| \le H_T + \epsilon T$. Outside the shell, it is at most $T \log |\mathcal{V}|$. Therefore,

$$H_T + m\rho \le H(\widetilde{Y}) \le \log 2 + (1-p)(H_T + \epsilon T) + pT \log |\mathcal{V}|. \tag{43}$$

Rearranging yields

$$p\big(T \log |\mathcal{V}| - H_T - \epsilon T\big) \ge m\rho - \epsilon T - \log 2.$$

Moreover, since $H(\widetilde{Y}) \le T \log |\mathcal{V}|$,

$$T \log |\mathcal{V}| - H_T - \epsilon T \ge m\rho - \epsilon T > 0.$$

Dividing by this positive quantity gives

$$\Pr[\widetilde{Y} \notin A_\epsilon^{(T)}] \ge \frac{m\rho - \epsilon T - \log 2}{T \log |\mathcal{V}| - H_T - \epsilon T},$$

which is strictly positive whenever $m\rho > \epsilon T + \log 2$.

Finally, the denominator is at most $T \log |\mathcal{V}|$, so under the same condition,

$$\Pr[\widetilde{Y} \notin A_\epsilon^{(T)}] \ge \frac{m\rho/T - \epsilon - (\log 2)/T}{\log |\mathcal{V}|}.$$

If $m\rho/T - \epsilon$ remains bounded below by a positive constant as T grows, this lower bound remains positive for all sufficiently large T. Part (a) gives the corresponding exponentially small escape probability for native policy rollouts. □

Remarks. A candidate with high surprise under the policy is not necessarily correct. Identifying the useful candidates among them is the role of the backward search, which is why the two components are coupled; without it, the forward search reduces to random exploration.

E Detailed Experimental Setup

E.1 Multi-Hop Reasoning

The training set is the solvable split of the MuSiQue training data; the validation set is the official MuSiQue validation set. The agent emits actions as `<think>...</think>` followed by either `<search>q</search>` or `<answer>...</answer>`. Each `<search>` call goes to the offline retriever which returns the top 3 passages back to the agent inside `<information>...</information>` tokens. The trainer uses 2 H200 GPUs. The retriever server (E5 encoder + FAISS over the 2018 Wikipedia dump) runs independently. We conduct two epochs of post-training. Tokens inside `<information>` are masked from the loss. Trajectories returned by `BES` are used as the rollouts for each training problem, and their log-probabilities are computed under the original prompt x by a forward pass of the policy.

Forward search. The forward search maintains a candidate pool of trajectories per question, where every trajectory is a sequence of $(\langle\texttt{think}\rangle, \langle\texttt{search}\rangle, \langle\texttt{information}\rangle)$ triples optionally followed by a terminal $(\langle\texttt{think}\rangle, \langle\texttt{answer}\rangle)$ pair. The evolution operators here operate at the triple boundary. The Boltzmann temperature τ is annealed linearly from 1.5 to 0.3. We pick $\alpha = 0.7$.

Backward search. During backward search, the policy model decomposes each question into a list of sub-questions. The original question together with these sub-questions defines the per-question goal tree. The local verifier V_{g_i} is queried by the forward search for every candidate trajectory at every step, so a per-call LLM verifier would be impractical. We therefore instantiate V_{g_i} as a fast embedding model: every $\langle\texttt{search}\rangle$ query that a trajectory n has emitted is embedded into the `all-MiniLM-L6-v2` sentence-embedding space, and the same is done for each sub-question. A sub-question g_i is declared *covered* by trajectory n when

$$\max_{q \in \text{searches}(n)} \cos\big(\text{emb}(q), \text{emb}(g_i)\big) \;\ge\; 0.6, \tag{44}$$

i.e. when at least one of the trajectory's search queries semantically aligns with the sub-question. The reliability of the local verifiers is quantified in Appendix J.

Hyperparameters are listed in Table 5.

Table 5: Hyperparameters for the multi-hop reasoning experiment (MuSiQue).

Category	Hyperparameter = Value

Optimization	epochs = 2; optimizer = AdamW; learning_rate = 1×10^{-6};
	lr_warmup_ratio = 0.285; group_size = 8; max_turns = 3;
	train_batch_size = 128; ppo_mini_batch_size = 16; ppo_micro_batch_size = 8;
	kl_loss_coef = 1×10^{-3}; kl_loss_type = low_var_kl.

Generation	max_prompt_length = 4096; max_response_length = 2048; temperature = 1.0.
BES search	action_types = {expand, delete, combine, translocate, crossover} with probs $\{0.70, 0.05, 0.10, 0.075, 0.075\}$.

E.2 Logical Reasoning

We generate the K&K corpus with the official sampling pipeline. The SFT cold-start corpus contains 1,000 problems. The post-training corpus contains 5,000 problems. Validation contains 1,287 problems. We conduct a 3-epoch SFT to teach the output format, followed by a 4-epoch post-training. Trajectories returned by `BES` are used as the rollouts for each training problem, and their log-probabilities are computed under the original prompt x by a forward pass of the policy.

Forward search. The forward search maintains a pool of partial reasoning trajectories partitioned at paragraph granularity (`\n\n`-separated reasoning steps). The Boltzmann temperature τ is annealed linearly from 2.0 to 1.0. A trajectory becomes *terminal* when it contains a "`### Final Answer`" marker followed by a parseable JSON map `{name: 0/1}`; the rule-based KK scorer then assigns reward $r \in \{0, 1\}$ from exact match against the ground-truth assignment. We pick $\alpha = 0.3$.

Backward search. At the start of every `BES` search on a puzzle, we instantiate a goal tree whose root goal is to identify every person's role correctly. The rest of the tree is produced through decomposition: given a goal, the model is asked to split it into finer sub-goals. As guidance, the prompt suggests that the root can be decomposed into one sub-goal per person, and that each per-person sub-goal can be further decomposed into elementary verification strategies, e.g. assume the opposite role and seek a contradiction. Every ten search steps, an unsolved leaf sub-goal is selected and the model is prompted to decompose it further. Each generated sub-goal is equipped with a lightweight syntactic verifier produced by the model.

Hyperparameters are listed in Table 6.

Table 6: Hyperparameters for the logical reasoning experiment (Knights-and-Knaves).

Category	Hyperparameter = Value

Optimization	optimizer = AdamW; learning_rate = 1×10^{-6}; group_size = 8;
	train_batch_size = 32; ppo_mini_batch_size = 32;
	ppo_micro_batch_size_per_gpu = 16; ppo_epochs = 1;
	clip_ratio = 0.2; grad_clip = 0.3; kl_coef = 0.0.

Generation	max_prompt_length = 1024; max_response_length = 4096; temperature = 1.0.
BES search	action_types = {expand, delete, combine, translocate, crossover} with probs $\{0.70, 0.05, 0.10, 0.075, 0.075\}$.

E.3 Open Problem Solving

We choose three open optimization tasks: (i) *Circle Packing (Square)*: pack $n = 26$ non-overlapping circles in the unit square to maximize the sum of radii; (ii) *Circle Packing (Rect)*: pack $n = 21$ non-overlapping circles in a rectangle of perimeter 4 with free aspect ratio to maximize the sum of radii; (iii) *Heilbronn (Convex)*: place 13 points in a convex region of unit area, i.e. the minimum

triangle area is normalized by the area of the points' convex hull, to maximize the minimum area over all $\binom{13}{3}$ triangles. We report mean and best objective value across 3 runs per benchmark. We use `gpt-5` with `reasoning_effort = high`. Each run is given 100 generations and $50 of GPT-5 API usage; both budgets include the calls used for backward decomposition.

Forward search. We adopt ShinkaEvolve as the base program-evolution framework. The four evolution operators are realized at the program level. We pass both parents to the proposer with an operator-specific instruction (e.g. "combine the structural decisions of program A with the parameter choices of program B") and ask it to return a single new program implementing the requested edit. We pick $\alpha = 0.3$.

Backward search. In the backward goal tree, each leaf carries a Python verifier expression that returns a partial-progress score in $[0, 1]$. Tree growth is performed adaptively during the run. A decomposition pass is triggered whenever the population has failed to improve on the raw objective for five consecutive generations. We pick a leaf that no program in the archive has yet fully satisfied, and ask the model to propose sub-goals that together cover the leaf, each with its own natural-language description and Python verifier.

Hyperparameters are listed in Table 7.

Table 7: Hyperparameters for the open problem solving experiment.

Category	**Hyperparameter = Value**
Generation	backbone = gpt-5; reasoning_effort = high.
BES Search	action_types = {expand, delete, combine, translocate, crossover} with probs $\{0.80, 0.05, 0.05, 0.05, 0.05\}$.

E.4 Mathematical Reasoning

We use the problems of AIME 2024 and AIME 2025. A trajectory is correct if the integer parsed from its final `\boxed{}` expression matches the reference answer. We report accuracy as the mean $\pm$ standard error over 3 independent runs. Models are served with one B200 GPU with a context length of 32768. Each problem is given a budget of 64 policy calls at 8,192 max tokens each, which includes the calls used for backward decomposition. Qwen3 models are run in non-thinking mode.

Forward search. All operators act on complete solutions through prompting. Expansion generates a fresh solution. Deletion re-solves the problem given the checks that the parent failed and it is asked to remove something in the original solution. Combination re-solves the problem given a summary of candidates from the pool. We pick $\alpha = 0.3$.

Backward search. The decomposition call asks the model to write executable Python self-check predicates for the problem, e.g. parity, divisibility, bound, or digit constraints on the answer.

Hyperparameters are listed in Table 8.

Table 8: Hyperparameters for the mathematical reasoning experiment (AIME 2024–2025).

Category	**Hyperparameter = Value**
Generation	temperature = 0.7; max_tokens = 8192.
BES search	action_types = {expand, delete, combine, translocate, crossover} with probs $\{0.80, 0.05, 0.05, 0.05, 0.05\}$;

E.5 Competitive Programming

We use LiveCodeBench v6: all problems from contests held between 2025-01-04 and 2025-04-06. Accuracy is computed with the official `lcb_runner`. Models are served with one B200 GPU with a

context length of 32768. Each problem is given a budget of 64 policy calls at 4,096 max tokens each, which includes the calls used for backward decomposition. Qwen3 models are run in non-thinking mode.

Forward search. Candidates are complete programs, and the evolution operators use prompting. Crossover merges two parent programs, translocation transplants exactly one technique from one parent into the other, combination merges inspiration from both programs, and deletion identifies the part of a program that is unnecessary and removes it. Parents are drawn by Boltzmann sampling with τ annealed linearly from 0.30 to 0.05 over the budget. We pick $\alpha = 0.3$.

Backward search. The goal tree is generated by the same policy model. Each sub-goal is an executable check with an expected value.

Hyperparameters are listed in Table 9.

Table 9: Hyperparameters for the competitive programming experiment (LiveCodeBench v6).

Category	**Hyperparameter = Value**
Generation	temperature = 0.7; top_p = 0.8; top_k = 20; max_tokens = 4096.
BES search	action_types = {expand, delete, combine, translocate, crossover} with probs $\{0.80, 0.05, 0.05, 0.05, 0.05\}$;

F Case Study

We illustrate the BES search process on a multi-hop question answering example from the MuSiQue dataset. The question is: *"What is the record label of the artist who originally recorded Back to Bedlam?"* The correct answer is Custard Records. Figure 5 shows the full search trace.

Backward search. The backward search decomposes the original question into two sub-goals: (1) *"Which artist originally recorded Back to Bedlam?"* and (2) *"What is the record label of #1?"*. Each sub-goal is equipped with a local verifier based on embedding similarity (Section E.1). This decomposition allows the search to track partial progress: a candidate that correctly identifies the artist (James Blunt) but retrieves the wrong label still receives a non-zero score, providing a useful signal for parent selection.

Forward search: expansion. The search begins with two expansion branches from the root. The left branch searches for the record label of "Back to Bedlam" directly and concludes that the album was released through Atlantic Records, which is incorrect. The right branch takes a broader approach, searching for James Blunt's career and music releases, and discovers that both Custard Records and Atlantic Records were involved. However, it also fails to produce the correct final answer, arriving at a wrong conclusion.

Forward search: translocation. At this point, the backward search scores both branches. The right branch receives a score of 0.3, indicating partial progress: it has identified relevant information about Custard Records but has not resolved the question correctly. The key moment occurs when the translocation operator replaces the reasoning step in the left branch (which concluded Atlantic Records) with the more nuanced reasoning step from the right branch (which noted both Custard Records and Atlantic Records). The resulting candidate inherits the left branch's identification of James Blunt as the artist and the right branch's awareness of the Custard Records connection. This recombined trajectory correctly concludes that the original record label is Custard Records, achieving a score of 1.0.

This example highlights two core advantages of BES. First, the backward search provides dense intermediate feedback: even though both initial branches produce wrong final answers, the sub-goal scores distinguish the branch that has made more relevant progress (score 0.3), enabling informed parent selection. Second, the translocation operator constructs a correct trajectory by transplanting a

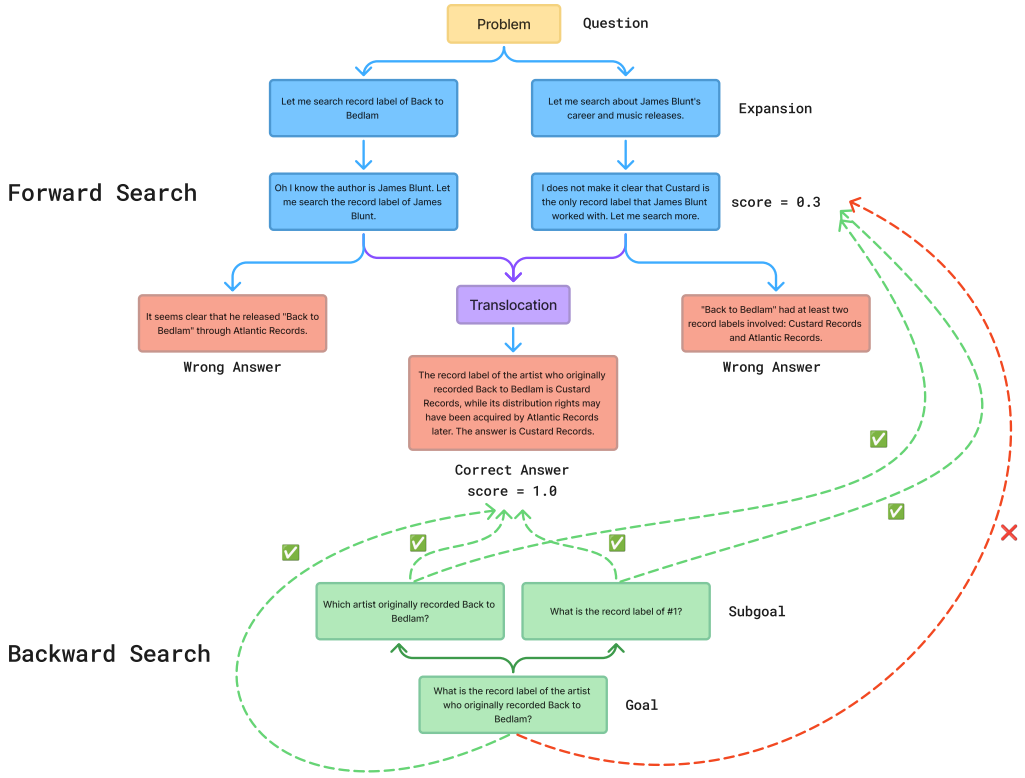

Figure 5: Case study of BES on a multi-hop reasoning problem. The forward search (top) explores two branches via expansion, both of which lead to wrong answers. A translocation operator then combines a reasoning step from the right branch into the left branch, producing a correct answer. The backward search (bottom) decomposes the original question into two sub-goals and provides dense verification feedback (green/red arrows) to guide parent selection.

single useful reasoning step from one branch into another. Neither branch alone would have reached the correct answer through further expansion, but their combination via translocation succeeds. This illustrates how evolution operators can discover solutions beyond what any single policy rollout can produce.

G Open Problem Solving with Open-Weight Models

The inference experiments in Section 4.2 use GPT-5 as the backbone model. To test whether the advantage of BES depends on a closed-source backbone, we repeat the Circle Packing (Square) experiment with two open-weight models, GLM-5.2 (GLM-5-Team et al., 2026) and DeepSeek-V4-Pro (DeepSeek-AI et al., 2026), keeping all other configurations unchanged. For each backbone, all frameworks are run under the same backbone model, API budget, and evaluation protocol as in Table 2, and we report the mean $\pm$ standard deviation and the best objective value over 3 runs.

As shown in Table 10, BES achieves the highest average and best objective values with every backbone. These results indicate that the gains of BES come from the search procedure rather than from a particular backbone model.

Table 10: Circle Packing (Square) with different backbone models. We report mean ± standard deviation and best objective values (↑) over 3 runs. **Bold** denotes the best performing framework per backbone.

Strategy	GPT-5		GLM-5.2		DeepSeek-V4-Pro	
	Avg.	Best	Avg.	Best	Avg.	Best
OpenEvolve	2.418 ± .123	2.541	2.534 ± .012	2.541	2.526 ± .027	2.542
GEPA	2.537 ± .004	2.541	2.541 ± .000	2.541	2.499 ± .059	2.541
ShinkaEvolve	2.478 ± .076	2.565	2.558 ± .026	2.595	2.458 ± .070	2.536
ShinkaEvolve + BES	**2.623** ± .014	**2.632**	**2.617** ± .008	**2.623**	**2.565** ± .042	**2.600**

H Hyperparameter Sensitivity Analysis

We study the sensitivity of BES to the blending weight α of the backward score (Eq. 4). All experiments are conducted on Circle Packing (Square) with GPT-5, keeping every other configuration as in Table 7, and we report the best objective value over 3 runs.
In Eq. 4, α controls how much the score of an unsolved sub-goal relies on its own verifier versus the scores of its children. Performance is stable for $\alpha \in [0, 0.7]$.

Table 11: Blending weight α of the backward score.

α	Best
0.0	2.630
0.3 (default)	**2.632**
0.7	2.612

I Empirical Verification of Shell Confinement and Escape

Motivation 3.1 predicts that trajectories produced by expansion-only search concentrate in a narrow entropy shell, while evolution operators can produce trajectories outside it. We provide empirical evidence consistent with both predictions using operators on real model outputs. We sample 4,000 trajectories from Llama-3.2-3B-Instruct (temperature 0.7, top-p 0.95) on MATH-500 (Lightman et al., 2024) and compute the per-token surprisal of the continuation after a shared prefix under the policy, $-\frac{1}{T}\log P(Y \mid x)$, where T is the number of continuation tokens. Following the definition of the typical set in Motivation 3.1, we define the empirical shell as the interval between the 10th and 90th percentiles of the expansion samples, $[0.079, 0.301]$ nats/token; its width of only 0.22 nats/token around a median of 0.16 shows how tightly expansion candidates are concentrated.

We then apply the forward operators to these trajectories and score the resulting candidates under the same policy: expansion continues a trajectory prefix with new policy samples (1,000 candidates), and the evolution operators inject a segment from a donor trajectory either by direct splicing at a step boundary or by prompting the policy to rewrite the continuation. Only 10.2% of the expansion operations produce a candidate above the upper edge of the shell. In contrast, on the cells with evolution operators, 33.0% of the direct-splice and 27.3% of the prompted-rewrite operations do so, roughly three times the rate of expansion, and the escaping candidates lie on the high-surprisal side of the shell. The fraction of candidates below the lower edge drops from 8.0% under expansion to 0.6% and 1.7%. These results show that evolution increases the frequency of candidates above the empirical shell boundary, while selecting the useful ones among them is the role of the backward search.

J Quality of the Embedding-Based Verifier on MuSiQue

The verifiers used in the multi-hop reasoning experiment (Appendix E.1) declare a sub-question covered when one of the trajectory's search queries has cosine similarity at least $\sigma_{\text{cov}} = 0.6$ with the sub-question in the embedding space. This is slightly different from whether the retrieved evidence actually *answers* it. To quantify how well the two agree, we collect the search logs of BES runs

with Llama-3.1-8B-Instruct from the MuSiQue training data and sample 192 (question, sub-question, trajectory node) triples from those problems. For each sampled triple, we compute the verifier output and ask GPT-5.6-luna to annotate whether the sub-question is answered by the passages retrieved along the trajectory.

On these triples the verifier attains a precision of 0.78, a recall of 0.74, a specificity of 0.92 and a negative predictive value of 0.90. The verifier is therefore a reasonable proxy: it rarely withholds credit incorrectly and captures most of the true progress, which is sufficient for its role in `BES`. A per-call LLM verifier would be more accurate but is too expensive here.

K Potential Limitations

We acknowledge the following limitations of the paper:

(1) `BES` requires an objective reward signal to guide the search. It has not been tested on subjective evaluation tasks where such signals are difficult to obtain, such as academic writing.

(2) The backward search relies on the policy's ability to decompose problems into meaningful sub-goals. For very weak models, this decomposition capability is limited.

(3) Due to resource constraints, our post-training experiments use relatively small models up to 8B parameters.