BFS-PO: Best-First Search for Large Reasoning Models

Fiorenzo Parascandolo[1,2]*, Wenhui Tan[3], Enver Sangineto[1]*, Xiaoyi Yu[3], Ruihua Song[3], Rita Cucchiara[1]
[1] University of Modena and Reggio Emilia, Modena, Italy,
[2] University of Pisa, Pisa, Italy
[3] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China

Abstract

Large Reasoning Models (LRMs) have shown excellent performance in reasoning tasks using long Chains of Thought (CoTs). However, this has also led to a significant increase of computational costs and the generation of verbose output, a phenomenon known as *overthinking*. The tendency to overthink is often exacerbated by Reinforcement Learning (RL) algorithms such as GRPO/DAPO. In this paper, we propose BFS-PO, an RL algorithm which alleviates this problem using a Best-First Search exploration strategy. Specifically, the reasoning chains generated at training time by BFS-PO are organized into a search tree, within which the shortest correct sequence is selected as the best and expanded using a backtracking mechanism based on maximum entropy nodes. In this way, we bias the exploration of the solution space towards the search for increasingly shorter solutions, training the LRM to generate progressively more concise answers. Using different benchmarks and base LRMs, we show that BFS-PO can simultaneously increase the LRM accuracy and shorten its reasoning chains. Our code and our trained models are publicly available at: `https://github.com/aimagelab/BFS-PO`.

1 Introduction

Large Reasoning Models (LRMs) have recently achieved impressive accuracy results using inference-time scaling [1, 2], in which the reasoning process is decomposed into a (long) sequence of simpler steps, called Chain of Thought (CoT) [3]. However, inference-time scaling usually leads to very long reasoning chains, which often consist of thousands or even tens of thousands of tokens [4, 5], producing high computational and memory overhead. This phenomenon is called *overthinking*, and it is due to the presence of redundant steps [6], errors in reasoning [7] and the use of many words for the sole purpose of textual coherence [8]. Moreover, several recent studies have empirically demonstrated that *excessive CoT length and accuracy are strongly negatively correlated* [7, 9–13]. For instance, in DAPO [14] and in [7, 15], the authors observe that Reinforcement Learning (RL) algorithms like GRPO [16] encourage longer responses when the *advantage* is negative, due to the CoT length-based normalization contained in the objective function (more details in App. A).

A common method to reduce overthinking is based on Supervised Fine-Tuning (SFT) of a pre-trained LRM using a dataset containing a short ground truth CoT associated with each question (Sec. 2). One of the pioneering approaches of this paradigm is TokenSkip [17], which utilizes GPT-4 to associate each token in a given CoT with an importance score. Less important tokens are pruned, and the LRM is fine-tuned with the resulting dataset. Many other works [18–28] follow this direction and are based on the creation of a dataset of triplets (question, CoT, answer), where the correct answer to a question is associated with a short CoT. However, training an LRM using SFT is often suboptimal

*Emails: fiorenzo.parascandalo@unimore.it; enver.sangineto@unimore.it

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

[2], especially with a limited dataset, and can lead to overfitting. Empirical results show that most of these works succeed in reducing the length of the CoTs, but at the cost of also reducing the accuracy.

In this paper, we follow a different direction: instead of using a static dataset of pre-computed short CoTs for SFT, we propose an RL algorithm in which *exploration is biased toward correct, short reasoning paths*. In this way, it is the LRM itself that generates its dataset of concise CoTs while it is trained. Specifically, we propose Best-First Search Policy Optimization (BFS-PO), an RL algorithm in which the exploration process is guided by a Best-First Search (BFS) strategy on a search tree. Unlike GRPO-type algorithms, where multiple answers to a given question are generated independently of each other, in BFS-PO, sampling is conditioned on the best solution found so far, where "best" is defined as the *shortest correct* solution. Importantly, BFS-PO does not require external modules to evaluate partial solutions and decide which ones to expand. Indeed, tree search-based strategies have been recently adopted in other works [29–31]. However, these methods need a Process Reward Model (PRM) [32], i.e. a module which can assess the quality of the current reasoning step and guide exploration within the search tree. Unfortunately, a PRM is usually implemented as an additional Large Language Model (LLM), which needs to be trained with supervised datasets containing annotations for partial solutions. Moreover, using a PRM requires additional computational resources during training or inference. In contrast, we avoid the need for a PRM by evaluating only complete solutions. Specifically, once a full solution is generated, we use a standard answer verifier [33] to assess its correctness. Then, the solution length is used to select the shortest correct reasoning chain. Once the current best solution is selected, we use a backtracking mechanism which interprets this solution as a path in a search tree, and it identifies the most promising forking node where the full solution should be split in order to pursue a different path. Since we do not have an external PRM, we select the backtracking node using the generation uncertainty. Indeed, as empirically shown in different recent works [34–37], the highest entropy CoT tokens behave as *forking tokens* in LRM reasoning, and they correspond to *those points in which the LRM takes the most important decisions*. Intuitively, if the entropy in predicting the next token is low, it means that the token is being used by the LLM primarily to ensure linguistic coherence; conversely, if the entropy is high, it indicates a branching point, as the LLM perceives that it could take different paths to continue its reasoning [34]. Hence, we use the highest entropy token as the backtracking point, and the partial path from the question to this point to bias the exploration of alternative partial paths (Fig. 1).

Recently, Hou et al. proposed TreeRL [38], in which, similarly to BFS-PO, a search tree is constructed using the token generation uncertainty for branching. However, unlike our entropy-based selection, TreeRL uses the negative log-likelihood of the next token as the criterion to select (multiple) forking tokens in a given path. Moreover, in TreeRL, the goal of the tree search is to efficiently replace classic Monte Carlo Tree Search (MCTS) [39] methods, and it is not based on a specific search strategy like Best First. In fact, in [38] the N nodes with maximum uncertainty are expanded in parallel (where N is an empirically set hyperparameter) with the aim of estimating the *value* of a node by simulating a Monte Carlo method. Finally, the advantage proposed in [38] compares the node value with the value of the root and the value of its parent node. In contrast, we expand only the correct, shortest solution found so far, with the aim of implementing a greedy search in the discrete space of solutions that leads to a local optimum, and we compute the advantage in GRPO/DAPO style.

We evaluate BFS-PO using different reasoning benchmarks and base LRMs, and we compare it with both SFT and RL-based baselines. Unlike most existing literature on CoT shortening, where brevity is achieved at the cost of an accuracy drop with respect to the zero-shot model, BFS-PO can simultaneously reduce the average CoT length *and* increase the accuracy, both compared to the zero-shot model, and also compared to an RL fine-tuning using strong baselines such as DAPO.

2 Related work

Due to its significant practical relevance, many works have recently been published on overthinking reduction. In this section, we briefly review the main methods, organized into different categories.

Budget forcing approaches [40, 41, 24, 42–44] for CoT shortening are based on specifying a maximum number of thinking tokens (usually via a specific prompt) and forcing an early exit by adding an end-of-thought token once this number has been generated. The advantage of budget

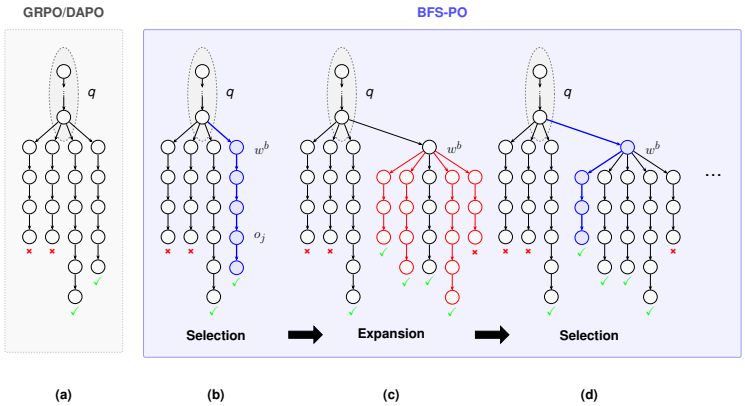

Figure 1: A schematic comparison between GRPO/DAPO (a) and BFS-PO (b-d). In (a), the sampling mechanism of GRPO/DAPO is represented as a simple tree with only one forking node (the root node, conditioned on the question q). ✓ and ✗ represent correct and incorrect answers, respectively. In (b) and (d), the current best solution is selected, while in (c) new branches are added to the tree. For simplicity, in this figure we use $G = 4$.

forcing is that it can be used as a training-free approach. However, its early exit usually leads to an accuracy drop with respect to the zero-shot base model.

SFT with short CoTs. As mentioned in Sec. 1, one of the most popular CoT shortening paradigm is based on constructing a dataset with ground truth short CoTs associated with each question, and then using SFT to train the model. Approaches in this category differ mainly in how the dataset is built and how to assess the importance of each token, e.g., using an auxiliary LLM [17, 21, 25, 28], the influence of a token on the response likelihood [18–20, 27] or other methods [22–24, 26, 28]. In contrast, we do not explicitly build a training dataset containing concise CoTs, but we let the LRM itself generate short CoTs, conditioning the exploration strategy toward the shortest correct solution.

RL with length penalty. Another common CoT shortening paradigm is based on RL algorithms which include a length penalty in their reward [45, 21, 5, 28]. For instance, in [13], the penalty is computed with respect to the current shortest positive answer, while in [46, 47], it is compared to the average length of the sampled CoTs. In [48, 28], a previous SFT phase with a compression control token is used to teach the model when to shorten the reasoning. We adopt the length penalty of DAPO [14] and we compare with DAPO in Sec. 5, showing that BFS-PO generates shorter CoTs with comparable or even slightly higher accuracy.

Steering vector. Ma et al. [49] build a dataset that associates CoTs of different lengths with each question. They then train LoRA models [50] with both long and short CoTs. The difference in parameter space between the two models indicates the direction to shorten the CoTs. A similar approach is used in [51], where the steering vector is computed in the activation space of some LRM layers rather than in the parameter space, and using PCA instead of parameter arithmetic.

Compression methods [52, 6, 53] compress the tokens of the generated CoTs, e.g., using specific attention masks which force the LRM to store information in intermediate tokens while discarding the others. Usually, this compression results in a decrease in the final accuracy.

Latent CoTs. The idea behind this category of methods [54, 55, 8, 56–63] is that human reasoning is often abstract and not always verbal, thus, rather than generating CoTs composed of discrete tokens, the LRM can generate continuous vectors in a latent space. Relaxing the textual coherence constraint, a sequence of latent tokens can potentially be shorter than a textual CoT. Although appealing, approaches belonging to this class have so far lagged behind the accuracy achieved by LRMs based on textual CoTs. One of the reasons is likely that sampling from a continuous distribution during RL training is more difficult than sampling from a distribution over a vocabulary of discrete tokens [64]. Soft Thinking [65] partially alleviates this problem using the LRM discrete distribution to softly aggregate all the vocabulary tokens. However, very recent works [66, 67] have shown that vanilla implementations of Soft Thinking often underperform their discrete counterparts, and its sampling mechanism mostly boils down to greedy decoding.

LRM with tree search. Methods using a tree search in conjunction with an LRM usually do not aim to shorten the generated CoTs, but to extend the linear reasoning, typical of a CoT, into a structured search. The latter includes, for example, techniques such as Breadth-First Search or Depth-First Search [31]. For instance, Tree of Thoughts [29, 30] break the reasoning process into atomic thoughts (where, e.g., a thought is a sentence), organized in a tree. Then, they use auxiliary LLMs and specific prompts to evaluate the current tree frontier and expand its nodes (i.e., thoughts). This mechanism has been extended in [68] to direct graphs. Hou et al. [38] observe that MCTS [39] cannot be easily applied to LRMs because of its high computational costs (see Sec. 1). Thus, they propose TreeRL, in which, similarly to BFS-PO, the exploration is organized using a tree search. However, they do not use BFS to explore the tree nor entropy to measure uncertainty, and they compute the value of the internal nodes of the tree by cumulating the rewards on its leaves, in this way replacing the random and costly exploration in a MCTS with an efficient search. We compare BFS-PO with TreeRL in Sec. 5, showing that our method drastically outperforms TreeRL.

3 Preliminaries

Problem Setup. Let π_θ be a policy model (i.e. a base LRM) with parameters θ, and $\mathcal{D} = \{(q_j, a_j^*)\}$ the training dataset, composed of problems (q_j, a_j^*), where q_j is the question and a_j^* the corresponding ground truth answer. Given q_j, the model autoregressively generates a token sequence $o_j = (w_1, ..., w_t, ..., w_{|o_j|})$ according to $o_j \sim \pi_\theta(\cdot|q_j)$, where $w_t \in V$ is a textual token and V the vocabulary of all tokens. We indicate with $o_{j,t} = w_t$ the t-th token of the reasoning chain o_j (we will use $o_{j,t}$ and w_t interchangeably). The output o_j is often formatted using special tokens (e.g., `<think>`, `</think>`) to delimit the intermediate reasoning chain from the final answer a_j. A verifier $v(\cdot)$, usually based on simple parsing and string matching techniques, extracts a_j from o_j and compares it with a_j^*: $v(o_j, a_j^*) = 1$ if $a_j = a_j^*$, and 0 otherwise. The goal is to fine-tune π_θ to decrease the average length of its reasoning chains while keeping (or increasing) its accuracy.

GRPO and DAPO. The success of DeepSeek-R1 [2] has popularized GRPO [16], an RL algorithm which has been shown to greatly increase the accuracy of LRMs. The main idea in GRPO is to replace the costly *value model* in PPO [69] with an *advantage* computed with respect to the mean reward of a group of G different outputs $\{o_i\}_{i=1}^G$ corresponding to the same question q. Specifically, given $(q, a^*) \in \mathcal{D}$, $\pi_{\theta_{old}}$ is used to sample $\{o_i\}_{i=1}^G$, and, for each o_i, the advantage is computed using:

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}, \tag{1}$$

where R_i is the sample-level reward and $\hat{A}_{i,t}$ is the token-level advantage, same for all tokens $o_{i,t} \in o_i$. DAPO [14] is an improved version of GRPO (see App. A), in which the same advantage is used in the following objective:

$$\begin{aligned}\mathcal{J}_{DAPO}(\theta) =& \mathbb{E}_{(q,a^*)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{old}}(\cdot|q)} \\ & \left[\frac{1}{\sum_{i=1}^G |o_i|}\sum_{i=1}^G\sum_{t=1}^{|o_i|}\Big(min(r_{i,t}(\theta)\hat{A}_{i,t}, clip(r_{i,t}(\theta), 1-\epsilon_l, 1+\epsilon_h)\hat{A}_{i,t}\Big)\right] \\ & \text{s.t.: } 0 < |\{o_i|v(o_i, a^*) = 1\}| < G, \text{where: } r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}.\end{aligned} \tag{2}$$

The constraint $0 < |\{o_i|v(o_i, a^*) = 1\}| < G$ is used to filter out trivial questions (where all the answers are correct) and questions which are too difficult for the current policy model π_θ, both leading to a 0 advantage. When this constraint is violated, q is discarded and a new problem (q, a^*) is sampled from $\mathcal{D}$. Moreover, in DAPO a penalty for very long sequences is added to the reward R_i:

$$\mathcal{R}_{len}(o) = \begin{cases} 0 & |o| \leq L_{max} - L_{cache} \\ \frac{(L_{max}-L_{cache})-|o|}{L_{cache}} & L_{max} - L_{cache} < |o| \leq L_{max} \\ -1 & L_{max} < |o|, \end{cases} \tag{3}$$

and the final reward is computed using:

$$R_i = \begin{cases} 1 + \mathcal{R}_{len}(o_i) & v(o_i, a^*) = 1 \\ -1 + \mathcal{R}_{len}(o_i) & v(o_i, a^*) = 0. \end{cases} \tag{4}$$

In this paper, we adopt the objective function of DAPO, including its length penalty, and, for a fair comparison, we *use the same hyperparameter values of DAPO* (more details in App. A). In Fig. 1 (a) we schematically show the group of G outputs, which can be thought of as independent paths of a simple tree containing only one branching node (the root node).

4 Method

The G different outputs used in GRPO and DAPO (see Sec. 3 and Fig. 1) are generated independently of one another and are used to explore the solution space, providing potentially different solutions for the same question q. However, classical optimization methods for planning and problem-solving usually organize the exploration of a discrete state-space of solutions using a search tree and strategies such as BFS, while employing mechanisms like backtracking to re-examine and modify a partial solution [70]. For instance, in our case, the benefit of a BFS strategy over multiple independent random sampling is that we can steer the exploration towards a local optimum of the solution space by progressively refining the current best solution to make it increasingly shorter. To implement this idea, we first select the best solution generated so far, where "best", in our case, means correct and shortest. Specifically, if $\mathcal{T} = \{o_i\}_{i=1}^{G}$, then let $o_j \in \mathcal{T}$ be such that:

$$v(o_j, a^*) = 1, \;\; |o_j| \leq min_{o_i \in \mathcal{T}} |o_i|. \tag{5}$$

Note that, since we adopt the resampling strategy of DAPO, i.e., we resample (q, a^*) if $|\{o_i | v(o_i, a^*) = 1\}| = 0$ or $|\{o_i | v(o_i, a^*) = 1\}| = G$ (Sec. 3), then there always exists at least one o_j such that $v(o_j, a^*) = 1$. For a given q, o_j, selected using Eq. (5), is the best solution generated so far. We now want to condition the exploration process using o_j. More specifically, we want to explore the solution space in the neighbourhood of o_j, searching for a shorter, correct solution. To do so, we need a backtracking mechanism, i.e., if we interpret o_j as a path in a search tree ($\mathcal{T}$), we need to identify the most promising forking node where the reasoning process should follow a different path. As mentioned in Sec. 1, and following the empirical findings in [34], we use the highest entropy tokens for backtracking. Specifically, we select the *backtracking* token $w^b \in o_j$ (Fig. 1 (b)) using:

$$w^b = o_{j,t^*} = \arg\max_{o_{j,t} \in o_j} H(o_{j,t}), \tag{6}$$

where:

$$H(o_{j,t}) = -\sum_{n=1}^{|V|} \pi_\theta(w_n | q, o_{j,\leq t}), log(\pi_\theta(w_n | q, o_{j,\leq t})). \tag{7}$$

Using o_{j,t^*}, we expand the current tree by adding new branches. In more detail, we use $(o_{j,1}, ..., o_{j,t^*})$ as a suffix which is concatenated to the question q to condition the generations of G new outputs, sampled using $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot | q, o_{j,1}, ..., o_{j,t^*})$. The new partial solutions $\{o_i\}_{i=1}^{G}$ are added to the search tree $\mathcal{T}$ (Fig. 1 (c)) and the process is iterated. The full algorithm is shown in Alg. 1, where B is the set of backtracking tokens. At iteration k, the new best solution o_j is selected (line 5) from all complete paths from the root to a leaf using Eq. (5). Then, a backtracking token (o_{j,t^*}) is chosen (line 6) in the current shortest correct solution o_j, where we exclude from the selection the previously chosen backtracking tokens ($o_{j,t^*} \notin B$) to increase the exploration diversity. To avoid an exponential tree growth, we use a fixed number of expansion iterations, K ($K = 3$, see Sec. 5.2), which plays a role analogous to that of G in GRPO/DAPO, limiting the exploration of the solution space for the sake of efficiency.

4.1 Optimization

Once $\mathcal{T}$ is built, we associate a reward and an advantage to each token which are then used to compute the objective function. Specifically, if w_n is the token associated to a generic node in $\mathcal{T}$, we indicate with $w_m = P(w_n)$ the parent of w_n. Similarly, we denote with $C(w_n) = \{w_{m_1}, ..., w_{m_k}\}$

the set of children of w_n. If $w_n \in B$, then $k = G + 1$; for leaf nodes, $k = 0$; and $k = 1$ for all the other nodes. Moreover, from each leaf node w_i, we use $P(\cdot)$ to extract a full reasoning path recursively until we reach the root (q), obtaining the output sequence $o_i = (w_1, ..., w_i) = (P(...P(w_i)...), ..., P(P(w_i)), P(w_i), w_i)$. Note that, while $\mathcal{T}$ is only partially ordered, each o_i is totally ordered; it is therefore consistent with the standard causal attention mechanism of a decoder-only transformer. If $\mathcal{T}$ has N leaves, we extract N complete reasoning chains $CoTs = \{o_i\}_{i=1}^{N}$. For each $o_i = (w_1, ..., w_i) \in CoTs$, we use Eq. (4) to compute a **reward** R_i, which is assigned to the *leaf* w_i. The rewards for the internal nodes are recursively computed averaging the rewards of their children. Specifically, if w_n is the token of a non-leaf node, its reward $R(w_n)$ is:

$$R(w_n) = \frac{1}{|C(w_n)|} \sum_{w_m \in C(w_n)} R(w_m). \tag{8}$$

Note that, if $w_n \notin B$, then $R(w_n) = R(C(w_n))$. This recursive reward computation is similar to the "value backpropagation" in MCTS [39], and it is different from the sample-level reward of GRPO/DAPO, in which all the tokens of a CoT receive the same reward (Sec. 3). The benefits of our token-level reward approach include both a partial reduction of the credit assignment problem and a decrease in the variance of the estimated values, thanks to the averaging in Eq. (8).

After a reward has been assigned to each token, for each $o_i \in CoTs$ and each $o_{i,t} \in o_i$, we compute the **advantage**, using:

$$\hat{A}_{i,t} = \frac{R(o_{i,t}) - \text{mean}(\{R_j | o_j \in CoTs\})}{\text{std}(\{R_j | o_j \in CoTs\})}. \tag{9}$$

Finally, our surrogate **objective function** is:

$$\mathcal{J}_{BFS-PO}(\theta) = \mathbb{E}_{(q,a^*)\sim\mathcal{D}, \mathcal{T}\sim\mu_{BFS}(\cdot|q;\pi_{\theta_{old}})} \left[\frac{1}{\sum_{i=1}^{N} |o_i|} \sum_{i=1}^{N} \sum_{t=1}^{|o_i|} \left(min(r_{i,t}(\theta)\hat{A}_{i,t}, clip(r_{i,t}(\theta), 1-\epsilon_l, 1+\epsilon_h)\hat{A}_{i,t} \right) \right], \tag{10}$$

where the outer sum iterates over the N sequences of $CoTs = \{o_i\}_{i=1}^{N}$ extracted from $\mathcal{T}$, and $\mu_{BFS}(\cdot|q;\pi_{\theta_{old}})$ represents the entire sampling procedure, which depends on both the old policy $\pi_{\theta_{old}}$ and Alg. 1. However, since the dependence of $\mu_{BFS}(\cdot|q;\pi_{\theta_{old}})$ on the procedure described in Alg. 1 makes the estimation of this likelihood non straightforward, we simplify the importance sampling ratio in Eq. (10) using only $\pi_{\theta_{old}}$, and we set: $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}$.

At **inference time** we use standard, single-chain, autoregressive sampling (without tree): given a question q, we sample $o \sim \pi_\theta(\cdot|q)$.

Algorithm 1 Tree Construction

Input: sample problem $(q, a^*) \sim \mathcal{D}$, policy model $\pi_{\theta_{old}}$, number of expansion iterations K.
$\mathcal{T} = \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot|q)$ s.t. $0 < |\{o_i | v(o_i, a^*) = 1\}| < G$ ▷ Initialization
$B = \{\}$
for $k = 1$ **to** K **do**
  Select a complete path o_j using Eq. (5) ▷ Selection
  $o_{j,t^*} = \arg\max_{o_{j,t} \in o_j, o_{j,t} \notin B} H(o_{j,t})$
  $\mathcal{N} = \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot|q, o_{j,1}, ..., o_{j,t^*})$ ▷ Expansion
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{N}$
  $B \leftarrow B \cup \{o_{j,t^*}\}$
end for

4.2 Implementation details

Computational cost. Most of the computational cost in both DAPO and BFS-PO comes from generating the reasoning chains, while the optimization step is the same and has the same computational cost for both methods (Sec. 4.1) and the overhead of tree construction is negligible. To make the

comparison fair, we use the same number of generated sequences per question and the same number of questions per generation batch. Specifically, for each question q, let G_D denote the number of output sequences generated by DAPO, and G_B denote the number of outputs generated by BFS-PO at each expansion step k ($k = 1, ..., K$, see Alg. 1). Since BFS-PO performs $K + 1$ tree expansions (including the initial generation at step 2), it generates $N = (K + 1)G_B$ sentences for each question (where $N = |CoTs|$, see Sec. 4.1). Thus, we set $G_D = N$, which implies $G_B = G_D/(K + 1)$ (we use $G_D = 32$ in all our experiments). Moreover, for both methods we use the same number of questions Q per generation batch. However, while in DAPO the QG_D sentences are generated in parallel, in BFS-PO we need $K + 1$ sequential steps to generate the same number QN of outputs. Nevertheless, even though BFS-PO requires K more generation steps than DAPO to generate the same number of sentences, the overall wall-clock time is comparable for two reasons. First, after the initial expansion in step 2, each generation in BFS-PO starts from a backtracking point, thus the tokens from the root to the backtracking point have already been generated and do not need to be recomputed. Second, as shown in Fig. 2 (App. D), during training, BFS-PO reduces the average length of the generated sequences much faster than DAPO, which leads to a large computational advantage. In all our experiments we use the same fixed training time budget for all the methods, and, as shown in App. D, in this period BFS-PO and DAPO complete almost the same number of training steps. To further validate this observation, in Sec. E we compare the wall-clock training time of BFS-PO with its closest competitors, DAPO and TreeRL, showing that there is no significant difference between the three approaches.

Discussion. Although we recursively compute a *token-level* reward (Eq. (8)), the advantage is based on a *sample-level* mean and standard deviation, using all the elements in $CoTs$ (Eq. (9)). In our preliminary experiments, we extended the local reward computation to the advantage as well, by replacing the mean and standard deviation in Eq. (9) with values calculated at each backtracking node. However, our empirical results show that the difference is not significant. More importantly, this forking-node based advantage has a computational issue when, for instance, all k branches of w_n yield a positive reward, which can result in a token-level advantage of zero. Although this issue could be resolved by adopting a resampling mechanism—similar to the one used in DAPO (Sec. 3)—*at each backtracking node*, doing so significantly increases computation time.

As mentioned above, in BFS-PO some output sequences share part of their tokens with other sequences because they have common sub-paths from the root to a backtracking node (see Fig. 1). Hence, since on average BFS-PO observes fewer tokens in a batch than DAPO, the question is: does this lead to a decrease in the diversity of solutions explored during training? In Sec. 5 we show that BFS-PO, in most experiments, achieves an accuracy equal to or higher than that of DAPO (as well as shorter responses), which empirically shows that BFS-PO can generalize as well as or better than DAPO. We believe that the main reason for this is that BFS-PO generates solutions that differ more in their "reasoning logic", compared to DAPO. In fact, as shown in [34], the tokens with low entropy are mainly used by an LRM to ensure the linguistic coherence of the sentence, while the highest entropy tokens are the actual decision points which determine the logical trajectory of the generated solution. Thus, the textual difference of the independently generated chains in DAPO does not necessarily correspond to a deep semantic difference. In contrast, at each expansion step of BFS-PO (Alg. 1), we branch the highest entropy node of the current best solution, and in this way we allow the LRM to explore different trajectories by making decisions "logically" different (Sec. 5.2).

5 Experiments

5.1 Experimental setup

We adopt two experimental protocols taken from TokenSkip [17] and Extra-CoT [28]. Extra-CoT [28] is a recently proposed work which combines SFT with short CoTs and RL with a length penalty (Sec. 2), where the variant in which RL is used is called CHRPO. Specifically, we use the In-Domain (**ID**) experimental setup taken from TokenSkip [17] and adopted also in Extra-CoT [28], in which training is done on the training split of GSM8K and testing on the testing split of the same dataset. In this case, following [17], the *base models* are: Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct [71] Llama3.2-3BInstruct and LLaMA-3.1-8B-Instruct [72], and the *token budget* is 512 tokens. Moreover, we use the experimental setup proposed in Extra-CoT [28], which is based on Out-of-Domain (**OOD**) training and testing, where the *training* phase uses a subset of 60k problem

samples randomly extracted from MetaMathQA-395K [73][2]. On the other hand, *testing* is based on different *benchmarks*: GSM8K [75], MATH-500 [76], AMC2023 [77] and MMLU-STEM [78]. GSM8K, MATH-500 and AMC2023 contain mathematical problems, while MMLU-STEM is based on Science, Technology, Engineering, and Mathematics (STEM) subjects. Following [28], in the OOD setup we use Qwen3-1.7B [79] as the *base model* with a *token budget* of 4k tokens.

Metrics. We use the evaluation script adopted in [28] for answer extraction and correctness verification. Besides the standard *Accuracy*, used to measure the answer correctness, we use *Average Length* to report the average number of tokens in the reasoning chains. Moreover, since a drastic reduction in CoT length usually also results in a reduction in accuracy (Sec. 1 and 2), we use the *Accuracy Efficiency Score (AES)*, proposed in [47], which combines the two metrics and evaluates the trade-off between improving accuracy and reducing computational cost (see App. C for more details).

Baselines. We compare BFS-PO with different baselines representing most of the paradigms mentioned in Sec. 2: 1) Zero-shot and standard SFT; 2) Training-free budget forcing methods, including length-control prompts (LC-Prompt [44]) and hard truncation of the output sequence (Truncation [44]) ; 3) SFT with short CoTs, including TokenSkip [17] and Extra-CoT [28], both with different compression ratios; 4) RL with length penalty, including Thinkless (DeGRPO) [48] and Extra-CoT (CHRPO) [28]; 5) DAPO, which is a state-of-the-art RL method for general-purpose LRM training; and 6) LRM with tree search, including Tree of Thought [29] and TreeRL [38].

5.2 Ablation

In our ablation experiments we use the ID scenario (Sec. 5.1), with training and testing on GSM8K, and Llama3.2-3B-Instruct as the base LRM (Tab. 1). The results of *Zero-shot* are taken from [28], while *DAPO* refers to our reproduction of DAPO [14] using the publicly available code and the hyperparameter values suggested by the authors. Note that *DAPO* in Tab. 1 includes the length penalty ($\mathcal{R}_{len}(\cdot)$) defined in Eq. (3). The only hyperparameter specific to BFS-PO is the number of expansions K (Alg. 1). If $K = 0$, then BFS-PO reduces to DAPO. Tab. 1 shows that, by varying the value of K from 0 (*DAPO*) to 3, both AES and average length significantly improve, while accuracy with $K = 3$ is slightly higher than DAPO. In all the other experiments of this paper, we use $K = 3$.

Table 1: Ablation experiments using the ID protocol and Llama3.2-3B-Instruct. Best results are shown in **bold**, and second-best results are underlined.

Methods	K	Len. Penalty	GSM8K		
			Ave. Len.↓	Acc↑	AES↑
Zero-shot	–	–	214	79.5	0.0
DAPO	0	✓	200	85.6	0.3
BFS-PO	1	✓	189	85.5	0.34
BFS-PO	2	✓	169	85.3	0.43
BFS-PO (full method)	3	✓	151	85.8	**0.53**
w/o BFS	3	✓	204	85.6	0.28
w/o Entropy	3	✓	**145**	82.7	0.44
w/o $\mathcal{R}_{len}(\cdot)$	3	✗	167	**86.0**	0.46

Table 2: Model comparison with in-domain experiments on GSM8K.

Methods	Ratio	Qwen2.5-3B-Instruct			Qwen2.5-7B-Instruct			Qwen2.5-14B-Instruct			Llama3.2-3B-Instruct			LLaMA-3.1-8B-Instruct			Average		
		Ave. Len.↓	Acc↑	AES↑	Ave. Len.↓	Acc↑	AES↑	Ave. Len.↓	Acc↑	AES↑	Ave. Len.↓	Acc↑	AES↑	Ave. Len.↓	Acc↑	AES↑	Ave. Len.↓	Acc↑	AES↑
Zero-shot	–	315	83.7	0.00	297	91.4	0.00	313	93.1	0.00	214	79.5	0.00	213	86.2	0.00	270	86.8	0.00
TokenSkip [17]	1.0	319	83.4	-0.03	296	91.7	0.01	315	93.0	-0.01	199	80.5	0.11	214	86.7	0.02	269	87.1	0.02
	0.9	263	83.2	0.13	255	91.1	0.12	269	93.3	0.15	182	78.0	0.05	198	86.1	0.06	233	86.3	0.10
	0.8	251	81.6	0.07	237	90.1	0.13	247	93.2	0.21	162	77.0	0.08	170	84.3	0.09	213	85.2	0.12
	0.7	233	80.1	0.04	217	89.9	0.18	219	93.4	0.31	153	74.1	-0.05	150	82.5	0.08	194	84.0	0.11
	0.6	200	77.3	-0.01	178	87.9	0.20	181	92.7	0.40	141	73.9	-0.01	129	81.1	0.10	166	82.6	0.14
	0.5	**171**	74.4	-0.09	**151**	86.0	0.19	**157**	91.4	0.41	123	72.5	-0.01	**113**	78.2	0.01	**143**	80.5	0.10
Extra-CoT [28]	1.0	–	–	–	298	92.3	0.02	–	–	–	196	81.5	0.15	–	–	–	247	86.9	0.09
	0.9	–	–	–	283	92.1	0.07	–	–	–	181	79.3	0.14	–	–	–	232	85.7	0.11
	0.8	–	–	–	261	91.7	0.13	–	–	–	163	79.2	0.21	–	–	–	212	85.4	0.17
	0.7	–	–	–	236	90.5	0.15	–	–	–	145	78.5	0.25	–	–	–	191	84.5	0.20
	0.6	–	–	–	222	89.9	0.17	–	–	–	132	77.0	0.22	–	–	–	177	83.5	0.20
	0.5	–	–	–	189	89.4	0.25	–	–	–	**115**	73.4	0.07	–	–	–	152	81.4	0.16
Tree Of Thoughts [29]	–	–	–	–	–	–	–	–	–	–	–	–	–	385	74.0	-1.51	385	74.0	-1.51
TreeRL	–	293	85.8	0.15	258	92.3	0.16	297	94.4	0.09	212	85.4	0.23	198	88.8	0.16	252	89.3	0.16
DAPO [14]	–	250	**87.4**	0.34	235	**92.8**	0.25	244	94.7	0.27	200	85.6	0.30	194	**89.6**	0.21	225	**90.0**	0.27
BFS-PO	–	198	86.3	**0.46**	200	**92.8**	**0.37**	193	**95.0**	**0.45**	151	**85.8**	**0.53**	125	89.5	**0.53**	173	89.9	**0.47**

[2]In [28], the authors use an additional dataset, CAMEL [74], for training their Compressor and additional data from MetaMathQA in the RL phase, which we do not use for simplicity.

Table 3: Model comparison with out-of-domain experiments and Qwen3-1.7B.

Method	Ratio	GSM8K			MATH-500			AMC2023			MMLU (STEM)			Average		
		Ave. Len.↓	Acc.↑	AES↑	Ave. Len.↓	Acc.↑	AES↑	Ave. Len.↓	Acc.↑	AES↑	Ave. Len.↓	Acc.↑	AES↑	Ave. Len.↓	Acc.↑	AES↑
Zero-shot	–	873	86.8	0.00	1675	64.2	0.00	2092	47.5	0.00	1081	74.1	0.00	1430	68.2	0.00
LC-Prompt [44]	0.6	763	86.2	0.09	1466	60.4	-0.17	2015	45.0	-0.23	-	-	-	1415	63.9	-0.10
Truncation [44]	0.6	692	81.7	-0.09	1271	49.8	-0.88	1334	25.0	-2.01	-	-	-	1099	52.2	-0.99
TokenSkip [17]	1.0	916	83.7	-0.23	1696	54.4	-0.78	2333	37.5	-1.17	1556	56.0	-1.66	1625	57.9	-0.96
	0.8	794	84.3	-0.05	1527	53.0	-0.78	2302	27.5	-2.21	1481	54.5	-1.69	1526	54.8	-1.18
	0.6	770	78.5	-0.36	1517	47.4	-1.21	2077	20.0	-2.89	1310	47.5	-2.01	1418	48.4	-1.62
	0.4	516	74.2	-0.32	1259	34.4	-2.07	1743	12.5	-3.52	877	29.8	-2.80	1099	37.7	-2.18
	0.2	273	59.1	-0.91	660	23.4	-2.57	911	10.0	-3.38	446	18.1	-3.19	572	27.7	-2.51
	<POLICY>	473	70.0	-0.51	1182	35.8	-1.92	1437	12.5	-3.37	1035	32.6	-2.76	1032	37.7	-2.14
Extra-CoT [28]	1.0	902	86.7	-0.04	1698	64.0	-0.03	2153	42.5	-0.56	1252	70.5	-0.40	1501	65.9	-0.26
	0.8	807	86.7	0.07	1520	64.2	0.09	1845	42.5	-0.41	1267	71.3	-0.36	1360	66.2	-0.15
	0.6	641	85.5	0.19	1304	59.6	-0.14	1624	40.0	-0.57	1012	60.6	-0.85	1145	61.4	-0.34
	0.4	469	82.3	0.20	920	54.2	-0.33	945	25.0	-1.82	730	61.3	-0.54	766	55.7	-0.62
	0.2	303	80.2	0.27	481	47.8	-0.56	782	17.5	-2.53	458	54.3	-0.76	506	49.9	-0.90
	<POLICY>	569	86.3	0.32	1312	63.0	0.12	1515	42.5	-0.25	1026	67.5	-0.39	1106	64.8	-0.05
Thinkless (DeGRPO) [48]	–	356	85.5	0.52	888	63.6	0.42	1369	50.0	0.50	-	-	-	871	66.4	0.47
Extra-CoT (CHRPO) [28]	<POLICY>	**210**	85.8	**0.70**	**452**	64.8	0.76	**675**	50.0	0.84	**330**	67.1	0.22	**417**	66.9	0.63
TreeRL [38]	–	786±9	87.1±0.4	0.11±0.03	1618±42	76.2±0.8	0.59±0.05	2050±95	59.2±3.6	0.76±0.23	932±9	76.8±0.4	0.25±0.03	1347±28	74.8±0.9	0.43±0.06
DAPO [14]	–	606±7	89.2±0.4	0.39±0.02	1307±31	80.7±0.8	1.00±0.04	2011±92	60.7±3.4	0.87±0.22	663±7	79.0±0.4	0.58±0.02	1147±24	77.4±0.8	0.71±0.05
BFS-PO	–	437±2	**89.7±0.4**	0.60±0.01	968±16	**81.6±0.7**	**1.24±0.04**	1558±76	**64.7±4.4**	**1.34±0.26**	459±2	**79.3±0.4**	**0.79±0.02**	856±19	**78.8±0.9**	**0.99±0.05**

In Tab. 1, *w/o BFS* refers to our tree search algorithm without BFS. Specifically, in this case, the backtracking point is *not* selected using the current best solution but only using the entropy. In more detail, in *w/o BFS*, we remove line 5 from Alg. 1 and we replace line 6 with:

$$o_{j,t^*} = \arg \max_{o_{j,t} \in \mathcal{T}, o_{j,t} \notin B} H(o_{j,t}). \tag{11}$$

We use *w/o BFS* to evaluate the contribution of a BFS strategy in which only the current best solution is expanded. Since in *w/o BFS* we expand the highest entropy node selected from all the current complete solutions (i.e., from all the paths from the root to a leaf), it can be considered similar to a breadth-first search strategy. The results in Tab. 1 show that *w/o BFS* can boost the accuracy with respect to *Zero-shot*, but the reduction of the average number of generated tokens is much more limited compared to our full method (BFS-PO with $K = 3$), which shows the effectiveness of a BFS strategy in reducing overthinking. We believe this is because our BFS strategy biases the LRM exploration towards progressively shorter solutions, thereby self-generating a training dataset of concise CoTs (Sec. 1).

The next entry in Tab. 1 is *w/o Entropy*, in which we select the backtracking point randomly instead of using entropy. Specifically, *w/o Entropy* corresponds to replacing step 6 in Alg. 1 with a random selection of o_{j,t^*} from all tokens of o_j, leaving everything else unchanged (*including the BFS strategy*, i.e., the selection of o_j using Eq. (5) in step 5). The results in Tab. 1 show that both accuracy and AES are much lower in *w/o Entropy* than in our full method, despite the average length being shorter. This shows that, although the BFS strategy in *w/o Entropy* leads to an effective CoT shortening, using entropy to select backtracking points is crucial to increasing accuracy. This is likely due to the fact that, since these high entropy tokens are the points where LRMs make the main decisions [34], our branching approach allows the model to better semantically differentiate the solutions explored at training time and, consequently, better generalize at inference time (Sec. 4.2).

Finally, in *w/o* $\mathcal{R}_{len}(\cdot)$, we remove the length penalty of DAPO (Eq. (3)) from Eq. (4) and we use a (leaf) reward based only on the sequence correctness ($R_i = 1$ if $v(o_i, a^*) = 1$ and $R_i = -1$ if $v(o_i, a^*) = 0$), keeping all the rest of our method unchanged. Overall, the results of *w/o* $\mathcal{R}_{len}(\cdot)$ are slightly worse than our full method, however they are much better than *DAPO*, *in which* $\mathcal{R}_{len}(\cdot)$ *is used*, which shows that the benefits of BFS-PO depend only marginally from the length penalty.

5.3 Main results

In Tab. 2 we use the ID experimental protocol (Sec. 5.1) to compare BFS-PO with other methods. The results of Zero-shot (i.e., the base models) and TokenSkip are taken from [17]. Specifically, TokenSkip with compression ratio = 1 corresponds to SFT (without CoT shortening and using the TokenSkip prompt). Moreover, the results of Extra-CoT are taken from [28], while the results of Tree of Thought [29] (Sec. 2) are taken from [80], and DAPO [14] is our reproduction. In Tab. 3 we use the OOD experimental protocol (Sec. 5.1). The results of Zero-shot, LC-Prompt [44], Truncation [44], TokenSkip [17], Extra-CoT [28], Thinkless (DeGRPO) [48] and Extra-CoT with RL (CHRPO) [28] are taken from [28]. Additionally, in the same table we include our reproduction of DAPO and TreeRL [38] using the publicly available code and the hyperparameter values suggested by

Table 4: Model comparison with Qwen3-4B-Base.

	GSM8K		
Method	Ave. Len. ↓	Acc. ↑	AES ↑
Zero-shot	301	91.0	0.00
TreeRL	304	92.8	0.05
DAPO	<u>298</u>	<u>93.1</u>	<u>0.08</u>
BFS-PO	**196**	**93.6**	**0.43**

the corresponding authors. For those methods for which we have access to the trained parameters (BFS-PO, DAPO and TreeRL), we repeated the testing phase using 16 different seeds, and we report in Tab. 3 the mean and the standard deviation[3].

Tab. 2 and Tab. 3 show that BFS-PO *consistently achieves the best AES score*, with both protocols and with all models and datasets, and often by a wide margin over other methods. Furthermore, in all experiments, BFS-PO consistently achieves either the best or the second-best result in terms of accuracy. Specifically, the average accuracy in Tab. 2 is nearly equal to that of DAPO, while in Tab. 3 it outperforms DAPO by about one and a half points. When compared with Zero-shot, BFS-PO significantly decreases the CoT length *and* increases the accuracy. For instance, considering the results averaged over the four datasets in Tab. 3, the mean CoT length of BFS-PO is 574 tokens shorter than Zero-shot (-40.1%), while accuracy improved by 10.6 points. Comparing the average length of the CoTs produced by BFS-PO with those of TreeRL and DAPO in Tab. 3, BFS-PO shortens the generated sequences by 36.5% and 25.4%, respectively. TokenSkip and Extra-CoT can compress more than BFS-PO, but only at the cost of a drastic reduction in accuracy compared to Zero-shot.

Finally, in Tab. 4 we use the ID experimental protocol with a much larger budget of 4096 tokens and Qwen3-4B-Base. The latter is a "Base" model—that is, a model lacking a post-training phase—and, consequently, it is characterized by a relatively high level of output uncertainty. The results in Tab. 4 show that, even when using a model with a high average token entropy, BFS-PO can largely decrease the CoT length (-34.9% with respect to Zero-shot) while simultaneously increasing the accuracy. We believe that this is due to the fact that, as shown in [34], token entropy is a signal of logical forking even for models without post-training, and is therefore a fairly universal indicator of those points in an LLM's reasoning trajectory where the model makes decisions and chooses the main paths.

Overall, the results reported in Tab. 2 to 4 show that BFS-PO significantly reduces the average CoT length with respect to Zero-shot while still achieving an accuracy equal to or better than DAPO.

6 Conclusions

We proposed BFS-PO, an RL algorithm for LRM in which the solution space is explored using a Best-First Search. Specifically, we avoid the use of a costly PRM by selecting the best solution from the complete solutions of the current search tree, where a standard verifier can be applied and the path length measured. We then use the highest entropy tokens for backtracking and tree expansion, following recent literature showing the importance of these tokens in LRM decisions. Using different benchmarks and models, we showed that BFS-PO can *simultaneously* increase accuracy *and* shorten the average CoT length, in stark contrast to state-of-the-art CoT shortening methods that typically sacrifice accuracy for length reduction.

Limitations. In our experiments we strictly followed the experimental protocols proposed in [28, 17], and we used LRMs with 1.7B to 14B parameters, which is one of the most common settings for CoT shortening methods. While extending the experiments using larger models would be interesting, training larger models using RL is very computationally expensive and beyond our computational resources, and we believe that the experimental protocols we have adopted can be used as a basis for comparison by a greater number of competitors.

[3]The other results in Tab. 3 are taken from [28], where the authors have not released the parameters of the tested methods, while the ID protocol used in Tab. 2 and 4 is based on greedy decoding, thus we cannot repeat testing using multiple seeds.

Acknowledgments

This work was supported by the PNRR project "Italian Strengthening of Esfri RI Resilience (IT-SERR)", funded by the European Union – NextGenerationEU (CUP B53C22001770006) and by the EU Horizon project ELIAS (No. 101120237). We acknowledge EuroHPC JU for awarding the project ID EHPC-AIF-2026SC01-043 access to Leonardo Booster hosted by CINECA, and the CINECA award under the ISCRA initiative for the availability of high-performance computing resources and support.

References

[1] OpenAI, "Openai o1 system card," *arXiv:22412.16720*, 2024.

[2] DeepSeek-AI, "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning," *arXiv:2501.12948*, 2025.

[3] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *NeurIPS*, 2022.

[4] M. Ballon, A. Algaba, and V. Ginis, "The relationship between reasoning and performance in large language models – o3 (mini) thinks harder, not longer," *arXiv:2502.15631*, 2025.

[5] B. Hou, Y. Zhang, J. Ji, Y. Liu, K. Qian, J. Andreas, and S. Chang, "Thinkprune: Pruning long chain-of-thought of LLMs via reinforcement learning," *arXiv:2504.01296*, 2025.

[6] J. Song, D. Jo, Y. Kim, and J.-J. Kim, "Reasoning path compression: Compressing generation trajectories for efficient LLM reasoning," in *NeurIPS*, 2025.

[7] M. Fatemi, B. Rafiee, M. Tang, and K. Talamadupula, "Concise reasoning via reinforcement learning," *arXiv:2504.05185*, 2025.

[8] S. Hao, S. Sukhbaatar, D. Su, X. Li, Z. Hu, J. Weston, and Y. Tian, "Training large language models to reason in a continuous latent space," in *COLM*, 2025.

[9] J. Su, J. Healey, P. Nakov, and C. Cardie, "Between underthinking and overthinking: An empirical study of reasoning length and correctness in LLMs," *arXiv:2505.00127*, 2025.

[10] W. Yang, S. Ma, Y. Lin, and F. Wei, "Towards thinking-optimal scaling of test-time compute for LLM reasoning," *arXiv:2502.18080*, 2025.

[11] M. Hassid, G. Synnaeve, Y. Adi, and R. Schwartz, "Don't overthink it. Preferring shorter thinking chains for improved LLM reasoning," *arXiv:2505.17813*, 2025.

[12] A. Agarwal, A. Sengupta, and T. Chakraborty, "First finish search: Efficient test-time scaling in large language models," *arXiv:2505.18149*, 2025.

[13] J. Yi, J. Wang, and S. Li, "ShorterBetter: guiding reasoning models to find optimal inference length for efficient reasoning," in *NeurIPS*, 2025.

[14] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, YuYue, W. Dai, T. Fan, G. Liu, J. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, R. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, Y. Song, X. Wei, H. Zhou, J. Liu, W.-Y. Ma, Y.-Q. Zhang, L. Yan, Y. Wu, and M. Wang, "DAPO: An open-source LLM reinforcement learning system at scale," in *NeurIPS*, 2025.

[15] Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, and M. Lin, "Understanding r1-zero-like training: A critical perspective," *arXiv:2503.20783*, 2025.

[16] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv:2402.03300*, 2024.

[17] H. Xia, C. T. Leong, W. Wang, Y. Li, and W. Li, "TokenSkip: Controllable chain-of-thought compression in LLMs," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

[18] X. Xu, S. Wang, X. Han, Z. Liu, H. Wu, P. Li, Z. Liu, M. Sun, and Z. He, "A*-Thought: efficient reasoning via bidirectional compression for low-resource settings," in *NeurIPS*, 2025.

[19] J. Jang, J. Kim, W. Kweon, S. Lee, and H. Yu, "Verbosity-aware rationale reduction: Sentence-level rationale reduction for efficient and effective reasoning," in *ACL*, 2025.

[20] Y. Xiao, J. Wang, R. Yuan, C. Xu, K. Xu, W. Li, and P. Liu, "LIMOPro: Reasoning refinement for efficient and effective test-time scaling," in *NeurIPS*, 2025.

[21] X. Cheng, J. Li, Z. Zhang, X. Tang, W. X. Zhao, X. Kong, and Z. Zhang, "Incentivizing dual process thinking for efficient large language model reasoning," in *NeurIPS*, 2025.

[22] Z. Chen, X. Ma, G. Fang, R. Yu, and X. Wang, "VeriThinker: learning to verify makes reasoning model efficient," in *NeurIPS*, 2025.

[23] T. Munkhbat, N. Ho, S. H. Kim, Y. Yang, Y. Kim, and S.-Y. Yun, "Self-training elicits concise reasoning in large language models," in *ACL*, 2025.

[24] T. Han, Z. Wang, C. Fang, S. Zhao, S. Ma, and Z. Chen, "Token-budget-aware LLM reasoning," in *ACL*, 2025.

[25] Y. Kang, X. Sun, L. Chen, and W. Zou, "C3oT: Generating shorter chain-of-thought without compromising effectiveness," in *AAAI*, AAAI Press, 2025.

[26] H. Zhao, Y. Yan, Y. Shen, H. Xu, W. Zhang, K. Song, J. Shao, W. Lu, J. Xiao, and Y. Zhuang, "Let LRMs break free from overthinking via self-braking tuning," in *NeurIPS*, 2025.

[27] Z. Qiao, Y. Deng, J. Zeng, D. Wang, L. Wei, F. Meng, J. Zhou, J. Ren, and Y. Zhang, "Concise: Confidence-guided compression in step-by-step efficient reasoning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2025.

[28] Y. Tang, B. Jia, W. Huang, L. Zhang, J. Xie, W. Li, R. Ji, and S. Lin, "Towards efficient large language reasoning models via extreme-ratio chain-of-thought compression," *arXiv:2602.08324*, 2026.

[29] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. R. Narasimhan, "Tree of thoughts: Deliberate problem solving with Large Language Models," in *NeurIPS*, 2023.

[30] J. Long, "Large Language Model guided tree-of-thought," *arXiv:2305.08291*, 2023.

[31] M. Besta, F. Memedi, Z. Zhang, R. Gerstenberger, G. Piao, N. Blach, P. Nyczyk, M. Copik, G. Kwaśniewski, J. Müller, L. Gianinazzi, A. Kubicek, H. Niewiadomski, A. O'Mahony, O. Mutlu, and T. Hoefler, "Demystifying chains, trees, and graphs of thoughts," *IEEE Trans. on PAMI*, vol. 47, no. 12, pp. 10967–10989, 2025.

[32] J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, and I. Higgins, "Solving math word problems with process- and outcome-based feedback," *arXiv:2211.14275*, 2022.

[33] N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, X. Lyu, Y. Gu, S. Malik, V. Graf, J. D. Hwang, J. Yang, R. L. Bras, O. Tafjord, C. Wilhelm, L. Soldaini, N. A. Smith, Y. Wang, P. Dasigi, and H. Hajishirzi, "Tulu 3: Pushing frontiers in open language model post-training," in *Second Conference on Language Modeling*, 2025.

[34] S. Wang, L. Yu, C. Gao, C. Zheng, S. Liu, R. Lu, K. Dang, X.-H. Chen, J. Yang, Z. Zhang, Y. Liu, A. Yang, A. Zhao, Y. Yue, S. Song, B. Yu, G. Huang, and J. Lin, "Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning," in *NeurIPS*, 2025.

[35] D. Shi, A. Asi, K. Li, X. Yuan, L. Pan, W. Lee, and W. Xiao, "Swireasoning: Switch-thinking in latent and explicit for pareto-superior reasoning LLMs," *arXiv:2510.05069*, 2026.

[36] Y. Fu, X. Wang, Y. Tian, and J. Zhao, "Deep think with confidence," *arXiv:2508.15260*, 2025.

[37] D. Cheng, S. Huang, X. Zhu, B. Dai, W. X. Zhao, Z. Zhang, and F. Wei, "Reasoning with exploration: An entropy perspective," in *AAAI*, 2026.

[38] Z. Hou, Z. Hu, Y. Li, R. Lu, J. Tang, and Y. Dong, "TreeRL: LLM Reinforcement learning with on-policy tree search," in *ACL*, 2025.

[39] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[40] N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. Candès, and T. Hashimoto, "s1: Simple test-time scaling," *arXiv:2501.19393*, 2025.

[41] G. Li, Y. Gao, Y. Li, and Y. Wu, "Thinkless: A training-free inference-efficient method for reducing reasoning redundancy," *arXiv:2505.15684*, 2025.

[42] Y. Fu, J. Chen, S. Zhu, Z. Fu, Z. Dai, Y. Zhuang, Y. Ma, A. Qiao, T. Rosing, I. Stoica, and H. Zhang, "Efficiently scaling LLM reasoning with certaindex," *arXiv:2412.20993*, 2025.

[43] G. Li, W. Cai, Y. Gao, and Y. Wu, "Syncthink: A training-free strategy to align inference termination with reasoning saturation," *arXiv:2601.03649*, 2026.

[44] A. Lee, E. Che, and T. Peng, "How well do LLMs compress their own chain-of-thought? a token complexity approach," in *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025.

[45] B. Sel, L. Huang, N. Ramakrishnan, R. Jia, and M. Jin, "LLMs can reason faster only if we let them," in *ICML*, 2025.

[46] D. Arora and A. Zanette, "Training language models to reason efficiently," in *NeurIPS*, 2025.

[47] H. Luo, L. Shen, H. He, Y. Wang, S. Liu, W. Li, N. Tan, X. Cao, and D. Tao, "O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning," *arXiv:2501.12570*, 2025.

[48] G. Fang, X. Ma, and X. Wang, "Thinkless: LLM learns when to think," in *NeurIPS*, 2026.

[49] X. Ma, G. Wan, R. Yu, G. Fang, and X. Wang, "Cot-valve: Length-compressible chain-of-thought tuning," in *ACL*, 2025.

[50] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank adaptation of large language models," in *ICLR*, 2022.

[51] Z. Lin, Z. Fu, Z. Chen, C. Chen, L. Xie, W. Wang, D. Cai, Z. Wang, and J. Ye, "Controlling thinking speed in reasoning models," in *NeurIPS*, 2025.

[52] J. Pang, F. Ye, D. F. Wong, X. He, W. Chen, and L. Wang, "Anchor-based large language models," in *ACL*, 2024.

[53] J. Zhang, Y. Zhu, M. Sun, Y. Luo, S. Qiao, L. Du, D. Zheng, H. Chen, and N. Zhang, "LightThinker: Thinking step-by-step compression," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[54] Y. Deng, K. Prasad, R. Fernandez, P. Smolensky, V. Chaudhary, and S. Shieber, "Implicit chain of thought reasoning via knowledge distillation," *arXiv:2311.01460*, 2023.

[55] Y. Deng, Y. Choi, and S. Shieber, "From explicit cot to implicit cot: Learning to internalize cot step by step," *arXiv:2405.14838*, 2024.

[56] N. Saunshi, N. Dikkala, Z. Li, S. Kumar, and S. J. Reddi, "Reasoning with latent thoughts: On the power of looped transformers," in *ICLR*, 2025.

[57] Z. Shen, H. Yan, L. Zhang, Z. Hu, Y. Du, and Y. He, "CODI: Compressing chain-of-thought into continuous space via self-distillation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2025.

[58] D. Su, H. Zhu, Y. Xu, J. Jiao, Y. Tian, and Q. Zheng, "Token assorted: Mixing latent and text tokens for improved language model reasoning," in *ICML*, 2025.

[59] J. Geiping, S. M. McLeish, N. Jain, J. Kirchenbauer, S. Singh, B. R. Bartoldson, B. Kailkhura, A. Bhatele, and T. Goldstein, "Scaling up test-time compute with latent reasoning: A recurrent depth approach," in *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025.

[60] D. Kong, M. Zhao, D. Xu, B. Pang, S. Wang, E. Honig, Z. Si, C. Li, J. Xie, S. Xie, and Y. N. Wu, "Latent thought models with variational bayes inference-time computation," in *ICML*, 2025.

[61] T. Lin, J. Xie, S. Yuan, and D. Yang, "Implicit reasoning in transformers is reasoning through shortcuts," in *ACL*, 2025.

[62] W. Tan, J. Li, J. Ju, Z. Luo, R. Song, and J. Luan, "Think silently, think fast: Dynamic latent compression of LLM reasoning chains," in *NeurIPS*, 2025.

[63] W. Tan, F. Parascandolo, E. Sangineto, J. Ju, Z. Luo, Q. Cao, R. Cucchiara, R. Song, and J. Luan, "Restoring exploration after post-training: Latent exploration decoding for large reasoning models," in *ICML*, 2026.

[64] LCM team, L. Barrault, P.-A. Duquenne, M. Elbayad, A. Kozhevnikov, B. Alastruey, P. Andrews, M. Coria, G. Couairon, M. R. Costa-jussà, D. Dale, H. Elsahar, K. Heffernan, J. M. Janeiro, T. Tran, C. Ropers, E. Sánchez, R. S. Roman, A. Mourachko, S. Saleem, and H. Schwenk, "Large concept models: Language modeling in a sentence representation space," *arXiv:2412.08821*, 2024.

[65] Z. Zhang, X. He, W. Yan, A. Shen, C. Zhao, and X. E. Wang, "Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space," in *NeurIPS*, 2025.

[66] J. Wu, J. Lu, Z. Ren, G. Hu, Z. Wu, D. Dai, and H. Wu, "LLMs are single-threaded reasoners: Demystifying the working mechanism of soft thinking," *arXiv:2508.03440*, 2025.

[67] N. Butt, A. Kwiatkowski, I. Labiad, J. Kempe, and Y. Ollivier, "Soft tokens, hard truths," *arXiv:2509.19170*, 2025.

[68] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, and T. Hoefler, "Graph of Thoughts: Solving Elaborate Problems with Large Language Models," in *AAAI*, 2024.

[69] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[70] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd ed., 2010.

[71] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu, "Qwen2.5 technical report," *arXiv:2412.15115*, 2025.

[72] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, *et al.*, "The Llama 3 herd of models," *arXiv:2407.21783*, 2024.

[73] L. Yu, W. Jiang, H. Shi, J. YU, Z. Liu, Y. Zhang, J. Kwok, Z. Li, A. Weller, and W. Liu, "MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models," in *ICLR*, 2024.

[74] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for "mind" exploration of large language model society," in *NeurIPS*, 2023.

[75] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *arXiv:2110.14168*, 2021.

[76] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, "Let's verify step by step," in *ICLR*, 2024.

[77] AI-MO Team, "Aimo validation AMC (AMC 2023 subset). Derived from AMC12 2022–2023 problems. This work uses the 2023 subset." `https://huggingface.co/datasets/AI-MO/aimo-validation-amc`, 2024.

[78] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *ICLR*, 2021.

[79] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu, "Qwen3 technical report," *arXiv:2505.09388*, 2025.

[80] T. McDonald, A. Colosimo, Y. Li, and A. Emami, "Can we afford the perfect prompt? balancing cost and accuracy with the economical prompting index," in *ACL*, 2025.

[81] L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, S. Huang, K. Rasul, and Q. Gallouédec, "TRL: Transformers Reinforcement Learning." `https://github.com/huggingface/trl`, 2020.

A GRPO and DAPO

The GRPO objective is [16]:

$$\mathcal{J}_{GRPO} = \mathbb{E}_{(q,a^*)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\min(r_{i,t}(\theta)\hat{A}_{i,t}, clip(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t} - \beta D_{KL}(\pi_\theta||\pi_{ref}))\Big)\right], \tag{12}$$

where:

$$\hat{A}_{i,t} = \frac{R_i - mean(\{R_i\}_{i=1}^{G})}{std(\{R_i\}_{i=1}^{G})}, \;\; r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}. \tag{13}$$

In Eq. (12), the Kullback-Leibler divergence $D_{KL}(\pi_\theta||\pi_{ref})$ is used to regularize the difference between the online policy (π_θ) and the frozen reference policy (π_{ref}), where the latter is usually the base LRM model before the RL fine-tuning starts. β is an hyperparameter which weights this KL penalty, while ϵ is the clipping-related hyperparameter introduced in PPO for stabilizing training [69].

In DAPO [14], the KL penalty is removed, while the clipping operation is bounded below by ϵ_l and above by ϵ_u. Moreover, Yu et al. [14] observe that, if all the G answers are correct or if they are all wrong, then the advantage is 0. Therefore, they resample those groups of outputs which do not contain at least one correct and one incorrect answer (see Sec. 3). Finally, Yu et al. [14] show that the normalization term $\frac{1}{|o_i|}$ in Eq. (12), which divides the objective function value of a given sample o_i by its length, leads to penalizing long correct answers ($v(o_i, a^*) = 1$) but also has the opposite effect, i.e. it decreases the penalty for long incorrect answers ($v(o_i, a^*) = 0$) [7, 62]. The authors empirically show that this leads to longer CoTs. Therefore, they replace this term by a group-based normalization formulated as $\frac{1}{\sum_{i=1}^{G}|o_i|}$:

$$\begin{aligned} &\mathcal{J}_{DAPO}(\theta) = \mathbb{E}_{(q,a^*)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(\cdot|q)} \\ &\left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\Big(min(r_{i,t}(\theta)\hat{A}_{i,t}, clip(r_{i,t}(\theta), 1-\epsilon_l, 1+\epsilon_h)\hat{A}_{i,t}\Big)\right] \\ &s.t. \;\; 0 < |\{o_i|v(o_i, a^*) = 1\}| < G. \end{aligned} \tag{14}$$

As mentioned in Sec. 3, the constraint $0 < |\{o_i|v(o_i, a^*) = 1\}| < G$ is used to ensure a non-zero advantage, and it is implemented by resampling the training problems which violate this constraint (Sec. 3). Finally, very long sequences are penalized using Soft Overlong Punishment, defined as:

$$\mathcal{R}_{len}(o) = \begin{cases} 0 & |o| \leq L_{max} - L_{cache} \\ \frac{(L_{max}-L_{cache})-|o|}{L_{cache}} & L_{max} - L_{cache} < |o| \leq L_{max} \\ -1 & L_{max} < |o|, \end{cases} \tag{15}$$

and the final reward is computed using:

$$R_i = \begin{cases} 1 + \mathcal{R}_{len}(o_i) & v(o_i, a^*) = 1 \\ -1 + \mathcal{R}_{len}(o_i) & v(o_i, a^*) = 0. \end{cases} \tag{16}$$

In all our experiments with DAPO we adopt the hyperparameter values suggested in [14]: $\epsilon_l = 0.2$, $\epsilon_h = 0.28$, $L_{cache} = \frac{1}{5}L_{budget}$ and $L_{max} = L_{budget} - L_{cache}$, where L_{budget} is the token budget, i.e., the maximum number of tokens which can be generated by the LRM (and depends on the specific computational resources). For a fair comparison, in all experiments with BFS-PO, *we use the same hyperparameter values* ($\epsilon_l = 0.2$, $\epsilon_h = 0.28$, $L_{cache} = \frac{1}{5}L_{budget}$, $L_{max} = L_{budget} - L_{cache}$), which are kept fixed without any variation.

B Experimental details

Our implementation is based on the TRL framework [81]. We fine-tune all models using LoRA [50], with rank $r = 8$ and $\alpha = 16$. We keep the optimization and training hyperparameters fixed across all experiments. Specifically, we use AdamW with learning rate $1e{-}6$ and 10 warmup steps. We use a generation batch size of 2048, except for Qwen2.5-14B-Instruct in Tab. 2 and Qwen3-1.7B in Tab. 3, for which we use 4096. In all experiments, we use a training batch size of 512, with a minibatch size of 32, corresponding to 16 training steps for each training batch. During training, we use a temperature of 1.0 and top-p of 1.0. We monitor the AES score during training and, for each method, we select the checkpoint corresponding to its highest AES score. For Qwen3-1.7B, which is evaluated on multiple datasets (Tab. 3), we monitor AES on MATH-500 (only) and use the selected checkpoint for testing on all datasets. As mentioned in Sec. 3 and App. A, for both BFS-PO and DAPO we adopt the hyperparameter values used in DAPO [14]: $\epsilon_h = 0.28$ and $\epsilon_l = 0.2$ for clipping, and $L_{cache} = \frac{1}{5}L_{budget}$, $L_{max} = L_{budget} - L_{cache}$ for $\mathcal{R}_{len}(\cdot)$. Following the TokenSkip [17] protocol, when training on GSM8K at inference time we use greedy decoding, and we set the maximum token budget $L_{\text{budget}} = 512$, except for the additional experiments with Qwen3-4B-Base (Tab. 4), for which we use $L_{\text{budget}} = 4096$. For the experiments in Tab. 3, following the Extra-CoT protocol, we use a temperature of 0.7 and top-p of 0.9 at inference time with $L_{\text{budget}} = 4096$. As mentioned in Sec. 4.2, for each base LRM we use the same training time budget for all methods. Specifically, for the experiments reported in Tab. 2, we use a training time budget of 24 hours for Qwen2.5-3B-Instruct and Llama3.2-3B-Instruct, and 72 hours for Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct, with 4 NVIDIA A100 GPUs in all four cases. For Qwen2.5-14B-Instruct (Tab. 2) and Qwen3-1.7B (Tab. 3), we use a training time budget of 168 hours on 16 NVIDIA A100 GPUs. Finally, for Qwen3-4B-Base (Tab. 4), we use a training time budget of 24 hours on 8 NVIDIA A100 GPUs.

C Metrics

Average Length. Following [28], in the OOD setup, when computing the length of an output sequence $|o|$, the token count is calculated using only the CoT tokens contained within the `<think>` ...`</think>` block. On the other hand, following [17, 28], in the ID protocol, $|o|$ includes also the final answer tokens[4].

Accuracy Efficiency Score (AES). The AES metric proposed in [47] combines the improvement in accuracy and the reduction in computational cost compared to the baseline model. It weights and sums the differences between the evaluated model and the baseline, taking into account both the average length of the reasoning chains and their accuracy. In our case, the baseline corresponds to the zero-shot performance of the model. The terms ΔLength and ΔAcc represent these relative changes, and they are computed as follows [47]:

$$\Delta\text{Length} = \frac{\text{Length}_{\text{baseline}} - \text{Length}_{\text{model}}}{\text{Length}_{\text{baseline}}}, \quad \Delta\text{Acc} = \frac{\text{Acc}_{\text{model}} - \text{Acc}_{\text{baseline}}}{\text{Acc}_{\text{baseline}}}. \tag{17}$$

Using these terms, the AES score is defined as:

$$\text{AES} = \begin{cases} \alpha \cdot \Delta\text{Length} + \beta \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} \geq 0 \\ \alpha \cdot \Delta\text{Length} - \gamma \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} < 0. \end{cases} \tag{18}$$

The coefficients α, β and γ are used to weight the different components of the score. In our experiments we use the default hyperparameters values suggested in [47], which are $\alpha = 1, \beta = 3,$ and $\gamma = 5$.

D Training curves

Fig. 2 shows the training curves of BFS-PO and DAPO for both the ID and OOD experimental protocols (Sec. 5). Specifically, rows (a)-(e) correspond to the ID experiments on GSM8K in Tab. 2,

[4]This distinction in the adopted protocols is presumably due to the fact that the *instruct* models used in the ID scenario do not separate the answer from the CoT using special formatting.

Table 5: Wall-clock training time and VRAM usage of BFS-PO, DAPO, and TreeRL.

	Model	Method	Training time budget (h)	Training time to best (h)	VRAM usage (GB)
Tab. 2	Qwen2.5-3B-Instruct	BFS-PO	24	22	40
		DAPO		22	
		TreeRL		23	
	Qwen2.5-7B-Instruct	BFS-PO	72	72	42
		DAPO		71	
		TreeRL		71	
	Qwen2.5-14B-Instruct	BFS-PO	168	166	62
		DAPO		158	
		TreeRL		166	
	Llama3.2-3B-Instruct	BFS-PO	24	21	40
		DAPO		22	
		TreeRL		23	
	Llama3.1-8B-Instruct	BFS-PO	72	71	42
		DAPO		69	
		TreeRL		70	
Tab. 3	Qwen3-1.7B	BFS-PO	168	162	61
		DAPO		150	
		TreeRL		164	
Tab. 4	Qwen3-4B-Base	BFS-PO	24	20	52
		DAPO		18	
		TreeRL		21	

row (f) refers to the OOD experiments with Qwen3-1.7B in Tab. 3, and row (g) corresponds to the additional ID experiments with Qwen3-4B-Base in Tab. 4. Note that, in all plots, each training step on the horizontal axis corresponds to 16 parameter updates (App. B). As mentioned in Sec. 4.2 and App. B, both DAPO and BFS-PO are trained with the same training time budget and the two methods complete approximately the same number of training steps. Independently of the model, BFS-PO has always a higher AES than DAPO during the entire training. Moreover, BFS-PO largely outperforms both TokenSkip and Extra-CoT, whose best AES values are significantly lower than the AES achieved by BFS-PO. The plots also show that the average length of BFS-PO is consistently lower than that of DAPO throughout training, which confirms that our method learns shorter reasoning chains from the early stages of training. Finally, the accuracy curves of BFS-PO and DAPO stay close to each other, showing that the reduction in reasoning length does not come at the cost of a loss in accuracy.

E Training time comparison

In Tab. 5, we compare the wall-clock training time of BFS-PO with DAPO and TreeRL. Specifically, "Training Time to Best" refers to the time needed to reach the highest AES score, which is the criterion used to select the best checkpoint (Sec. B). Moreover, "Tab. X" refers to the experiments in the X-th table of this paper. Tab. 5 shows that the three methods require comparable training times on average. Specifically, the average "Training Time to Best" is 76.3 hours for BFS-PO, 72.9 hours for DAPO, and 76.9 hours for TreeRL. Compared with DAPO, BFS-PO requires about 4.7% more time on average, while it requires about 0.8% less time than TreeRL. Finally, since we keep the hardware, batch size, token budget, and memory allocation fixed across the three methods, BFS-PO, DAPO and TreeRL use the same amount of VRAM in each experiment. The VRAM and wall-clock time results show no evidence of memory or communication bottlenecks for BFS-PO.

F Qualitative analysis

F.1 Solution space exploration during training

In this section, we show how BFS-PO progressively generates shorter correct CoTs during the tree construction using the example in Fig. 3. Specifically, given a question, the LRM initially generates G answers (line 2 of Alg. 1). Line 5 of Alg. 1 selects the shortest correct answer o_1, which, in this example, has 277 tokens (see Fig. 3). Line 6 selects the first backtracking token w_1^b, which corresponds to pieces. Then, line 7 samples G partial output sequences starting from w_1^b (i.e., using

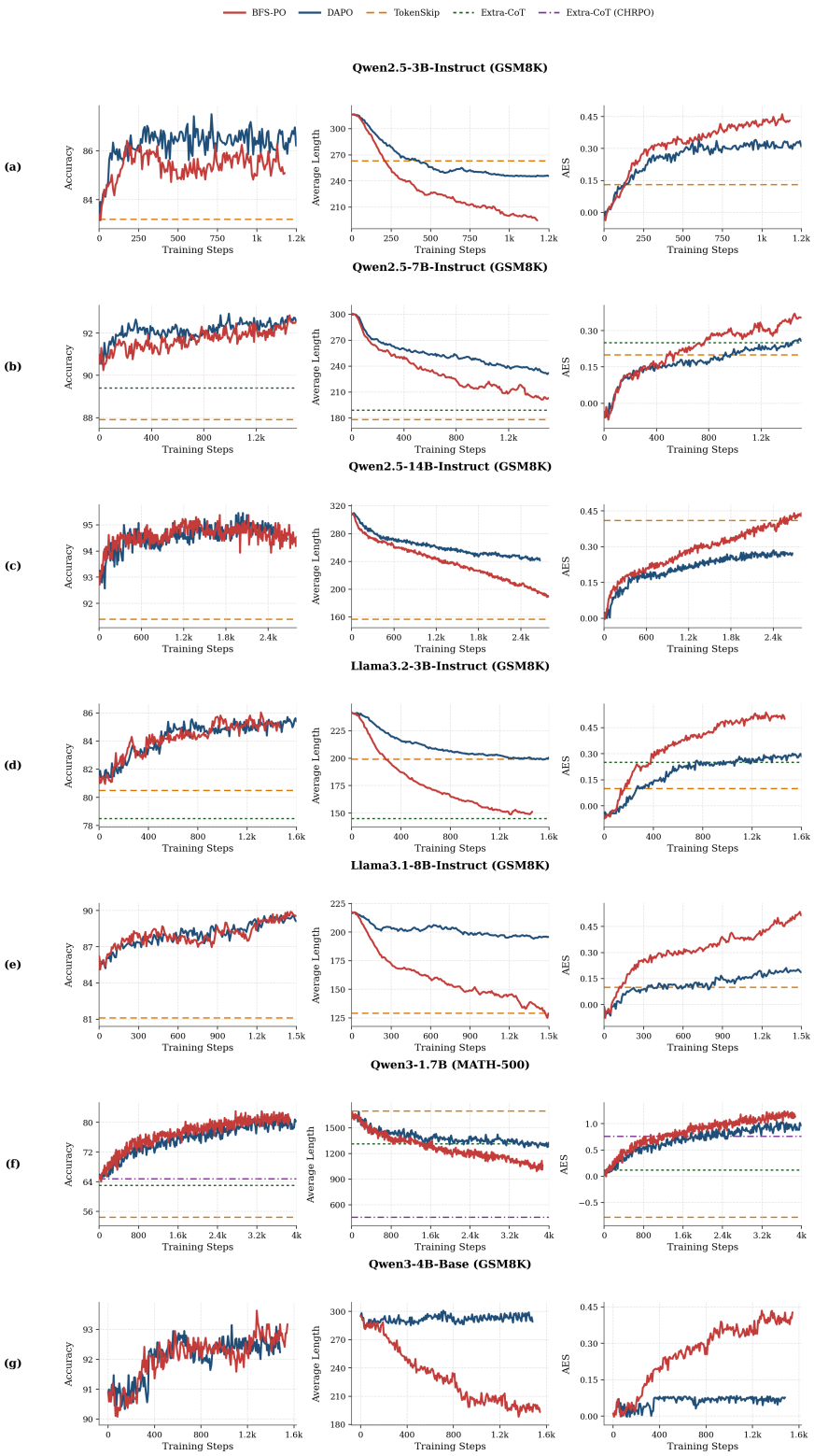

Figure 2: Training curves of BFS-PO and DAPO for the experiments in Tab. 2 to 4. Evaluation during training is performed on GSM8K for all models except Qwen3-1.7B, which is evaluated on MATH-500 (see App. B). Additionally, for TokenSkip, Extra-CoT and Extra-CoT (CHRPO) we show with dashed lines the values corresponding to the best AES results reported in Tab. 2 and 3 and taken from [17, 28].

the text "`Let ... pieces`" as a suffix to the question). In the second iteration, BFS-PO selects the new shortest correct solution, which, in this example, is o_2, that differs from o_1 starting from the token her. In o_2, the new selected backtracking point (w_2^b) is pies, which is used for the second expansion. In the third iteration, the new shortest correct output is o_3, with only 198 tokens. Note that o_3 is 28% shorter than o_1.

Exploration of the solution space in BFS-PO

Question (q): Grace baked 4 whole pumpkin pies. She sold 1 whole pumpkin pie and gave 1 whole pumpkin pie to her friend. The remaining whole pumpkin pies were each sliced into 6 pieces. Her family ate 2/3 pieces. How many pieces of pumpkin pie were left?

Answer: 4

---

o_1 **(277 tokens):**

```
Let's break down the problem step-by-step:\n \n 1.  Grace baked 4 whole pumpkin pies.\n 2.  She sold
1 whole pumpkin pie.\n 3.  She gave 1 whole pumpkin pie to her friend.\n 4.  Therefore, the number of
pumpkin pies remaining is:\n \\[\n 4 - 1 - 1 = 2 \\text{ whole pumpkin pies}\n \\]\n \n 5.  Each of
the remaining 2 whole pumpkin pies was sliced into 6 pieces:\n \\[\n 2 \\text{ pies} \\times 6 \\text{
pieces per pie} = 12 \\text{ pieces}\n \\]\n \n 6.  Her family ate \\(\\frac{2}{3}\\) of the total
pieces of pie.  To find out how many pieces were eaten:\n \\[\n \\frac{2}{3} \\text{ of } 12 \\text{
pieces} = \\frac{2}{3} \\times 12 = 8 \\text{ pieces}\n \\]\n \n 7.  To find out how many  pieces
are  left after the family ate:\n \\[\n 12 \\text{ pieces} - 8 \\text{ pieces} = 4 \\text{ pieces}\n
\\]\n \nTherefore , the number of pieces of pumpkin pie left is:\n \\[\n \\boxed{4}\n \\]
```

---

o_2 **(241 tokens):**

```
Let's break down the problem step-by-step:\n \n 1.  Grace baked 4 whole pumpkin pies.\n 2.  She sold
1 whole pumpkin pie.\n 3.  She gave 1 whole pumpkin pie to her friend.\n 4.  Therefore, the number of
pumpkin pies remaining is:\n \\[\n 4 - 1 - 1 = 2 \\text{ whole pumpkin pies}\n \\]\n \n 5.  Each of
the remaining 2 whole pumpkin pies was sliced into 6 pieces:\n \\[\n 2 \\text{ pies}\\times 6 \\text{
pieces per pie} = 12 \\text{ pieces}\n \\]\n \n 6.  Her family ate \\(\\frac{2}{3}\\) of the total
pieces of pie.  To find out how many  pieces  her  family ate:\n \\[\n \\frac{2}{3} \\times 12 = 8
\\text{ pieces}\n \\]\n \n 7.  To find the number of pieces left:\n \\[\n 12 - 8 = 4 \\text{ pieces}\n
\\]\n \nTherefore , the number of pieces of pumpkin pie that were left is \\(\\boxed{4}\\).
```

---

o_2 **(241 tokens):**

```
Let's break down the problem step-by-step:\n \n 1.  Grace baked 4 whole pumpkin pies.\n 2.  She sold
1 whole pumpkin pie.\n 3.  She gave 1 whole pumpkin pie to her friend.\n 4.  Therefore, the number
of pumpkin  pies  remaining  is:\n \\[\n 4 - 1 - 1 = 2 \\text{ whole pumpkin pies}\n \\]\n \n 5.
Each of the remaining 2 whole pumpkin pies was sliced into 6 pieces:\n \\[\n 2 \\text{ pies}\\times 6
\\text{ pieces per pie} = 12 \\text{ pieces}\n \\]\n \n 6.  Her family ate \\(\\frac{2}{3}\\) of the
total pieces of pie.  To find out how many pieces her family ate:\n \\[\n \\frac{2}{3} \\times 12 = 8
\\text{ pieces}\n \\]\n \n 7.  To find the number of pieces left:\n \\[\n 12 - 8 = 4 \\text{ pieces}\n
\\]\n \nTherefore , the number of pieces of pumpkin pie that were left is \\(\\boxed{4}\\).
```

---

o_3 **(198 tokens):**

```
Let's break down the problem step-by-step:\n \n 1.  Grace baked 4 whole pumpkin pies.\n 2.  She sold
1 whole pumpkin pie.\n 3.  She gave 1 whole pumpkin pie to her friend.\n 4.  Therefore, the number
of whole pumpkin  pies  left  is \\(4 - 1 - 1 = 2\\).\n 5.  Each of the remaining 2 whole pumpkin
pies is sliced into 6 pieces.\n 6.  Thus, the total number of pie pieces is \\(2 \\times 6 = 12\\)
pieces.\n 7.  Her family ate \\(\\frac{2}{3}\\) of the total pieces.\n 8.  The number of pieces her
family ate is \\(\\frac{2}{3} \\times 12 = 8\\) pieces.\n 9.  Therefore, the number of pieces left is
\\(12 - 8 = 4\\).\n \nPutting the final answer within the \\boxed{}:\n \\[\n \\boxed{4}\n \\]
```

Figure 3: A qualitative example of the Best-First Search strategy using Qwen2.5-3B-Instruct during training on GSM8K.

F.2 Inference-time comparison with the base model

Fig. 4 compares BFS-PO with the base model *at inference time*, and shows that BFS-PO generates shorter reasoning chains without skipping essential reasoning steps. Specifically, both the base model (Zero-shot) and BFS-PO generate reasoning sequences which can be partitioned into five logical steps, highlighted in Fig. 4 using the same color. The main difference concerns the way the reasoning step is expressed. In the first block, the base model introduces the first subproblem with: "`Calculate the amount of dog food used in the first 180 days`", while BFS-PO uses a shorter sentence: "`First 180 days`". However, both intermediate reasoning chains address the same subproblem and reach the same result. In the second block, the base model generates: "`Subtract the first 180 days from the total days in a year`",

while BFS-PO expresses the same operation more directly as: "`Remaining days = 365 - 180 = 185 days`". The main compression appears in the third block and fourth block. The base model first introduces the second-period computation in the third block with: "`Therefore, the total amount of dog food used in the remaining 185 days is`" and then combines the two quantities in the fourth block with: "`Add the dog food used in the first 180 days and the remaining 185 days`". BFS-PO, instead, keeps in the third block only the premise: "`For the first year, we only need to consider the first 185 days`", and then performs both the multiplication and the aggregation in the fourth block through: "`Next 185 days: 185 * 2 = 370 cups`" and "`Total for the first year = 180 + 370 = 550 cups`". This comparison shows that BFS-PO does not skip the computation of 370 cups. It simply moves it to the next block and expresses it more compactly. In the fifth block, the base model generates: "`Divide the total amount of dog food by the number of cups per bag`", while BFS-PO generates only the final calculation: "`Number of bags = 550 / 110 = 5 bags`".

Reasoning Chain Comparison: Zero-shot vs BFS-PO (Qwen2.5-3B-Instruct)

Question (q): Cecilia just bought a new puppy. According to her veterinarian, she has to feed the puppy 1 cup of dog food every day for the first 180 days. Then she has to feed the puppy 2 cups of dog food every day for the rest of its life. If one bag of dog food contains 110 cups, how many bags of dog food will Cecilia use in the first year?

Answer: 5

Zero-shot (504 tokens):

```
To determine how many bags of dog food Cecilia will use in the first year, we need to break down the problem into two parts: the first 180 days and the rest of the puppy's life.\n \n
1. **Calculate the amount of dog food used in the first 180 days:**\n - The puppy eats 1 cup of dog food per day for the first 180 days.\n - Therefore, the total amount of dog food used in the first 180 days is:\n \\[\n 1 \\text{ cup/day} \\times 180 \\text{ days} = 180 \\text{ cups}\n \\]\n \n
2. **Determine the number of days remaining in the first year:**\n - A year has 365 days.\n - Subtract the first 180 days from the total days in a year:\n \\[\n 365 \\text{ days} - 180 \\text{ days} = 185 \\text{ days}\n \\]\n \n
3. **Calculate the amount of dog food used in the remaining 185 days:**\n - After the first 180 days, the puppy eats 2 cups of dog food per day.\n - Therefore, the total amount of dog food used in the remaining 185 days is:\n \\[\n 2 \\text{ cups/day} \\times 185 \\text{ days} = 370 \\text{ cups}\n \\]\n \n
4. **Calculate the total amount of dog food used in the first year:**\n - Add the dog food used in the first 180 days and the remaining 185 days:\n \\[\n 180 \\text{ cups} + 370 \\text{ cups} = 550 \\text{ cups}\n \\]\n \n
5. **Determine the number of bags of dog food needed:**\n - Each bag contains 110 cups of dog food.\n - Divide the total amount of dog food by the number of cups per bag:\n \\[\n \\frac{550 \\text{ cups}}{110 \\text{ cups/bag}} = 5 \\text{ bags}\n \\]\n \n
Thus, Cecilia will use \\boxed{5} bags of dog food in the first year.
```

BFS-PO (320 tokens):

```
To determine how many bags of dog food Cecilia will use in the first year, we need to calculate the total amount of dog food required for the first 180 days and then for the rest of the puppy's life within the first year.\n \n
1. **First 180 days:**\n - The puppy needs 1 cup of dog food per day.\n - Total for 180 days = 180 cups.\n \n
2. **Remaining days in the first year:**\n - A year has 365 days.\n - Remaining days = 365 - 180 = 185 days.\n \n
3. **After the first year:**\n - The puppy will need 2 cups of dog food per day for the rest of its life.\n - For the first year, we only need to consider the first 185 days.\n \n
4. **Total dog food needed for the first year:**\n - First 180 days: 180 cups.\n - Next 185 days: 185 * 2 = 370 cups.\n - Total for the first year = 180 + 370 = 550 cups.\n \n
5. **Number of bags required:**\n - Each bag contains 110 cups.\n - Number of bags = 550 / 110 = 5 bags.\n \n
Thus, Cecilia will use \\boxed{5} bags of dog food in the first year.
```

Figure 4: A qualitative comparison of reasoning chains for a GSM8K problem using Qwen2.5-3B-Instruct, which shows the inference-time differences between Zero-shot and BFS-PO.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction we claim that (1) BFS-PO can alleviate the overthinking problem and (2) we sketch the main methodological aspects of our proposal. The first claim is empirically validated in Sec. 5, where BFS-PO is compared to different SOTA baselines, while the second is detailed in Sec. 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: We discussed the limitations of our experiments in Sec. 6, in an ad hoc paragraph called "Limitations", in which we say that, for computational reasons, we cannot use models larger than 14B parameters.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: This paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: We provide our code, the network parameters and the evaluation procedure with the Supplementary Material. Moreover, our method is explained in Sec. 4 and the experimental protocols (including the public datasets we used) in Sec. 5.1. Finally, the Appendix sections A, B and C provide the details and all the hyperparameter values, thus our method and our experiments are fully-reproducible.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: In the Supplementary Material we provided our code and all the instructions to execute it and reproduce the main results (our code will be published after the acceptance of this paper). Moreover, we do not use private data but only public datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: We followed two training/testing protocols, respectively taken from [28] and [17]. These are clearly explained in Sec. 5.1, with additional details in Sec. B and Sec. C. In Sec. A and Sec. B we provide all the hyperparameter values, type of optimizer used and all the the training and testing details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

Justification: Unfortunately, error bars or other types of statistical measures which need to repeat the training phase more than once are computationally infeasible for the LRM domain, especially when using RL-based algorithms. As far as we know, no paper reports these kinds of statistics. However, testing is much more computationally efficient, thus we repeated the experiments in Tab. 3 using 16 different seeds and we reported the means and the standard deviations. Please, note that the experiments in Tab. 2 and 4 are based on the greedy decoding protocol proposed in [17], thus we cannot repeat testing using multiple seeds. Importantly, our experiments are based on multiple datasets, and Tab. 2 to 4 show that BFS-PO consistently beats SOTA on all the datasets and settings, demonstrating the scientific robustness of our results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The computer resources for all our experiments are provided in Sec. B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: Our research respects all the details of NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No].

    Justification: Our contribution is entirely methodological and application agnostic: it provides a generic method for limiting overthinking in LRMs, without targeting any particular domain or use case, and therefore does not raise any specific social or ethical concerns that need to be highlighted in this context.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: We do not release any dataset or pre-trained model, but just a procedure to fine-tune an existing, pre-trained LRM.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: We use only public datasets and public codes. We have cited all works, datasets, codes and assets that are mentioned in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: No new assets are released in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: This article does not contain any crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: This article does not contain any participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: The method we propose in this article is an LLM fine-tuning approach, thus the entire article is based on the detailed description of this method and how it is used with standard, publicly available LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.