TDS: A TABLE-DRIVEN PARALLEL MULTI-AGENT SEARCH SYSTEM WITH ASYMMETRIC CREDIT GATING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic search answers a complex question by decomposing it into subproblems and gathering evidence over successive steps. Supervision comes from a single signal, the reference answer to the original question, while the intermediate subproblems carry no labels of their own. Learning from this supervision raises two questions. The first concerns how to organize intermediate results, which requires balancing sufficient evidence against concise records while maintaining a consistent search structure during training and inference. The second concerns assigning credit to intermediate decisions in multi-agent search, where value-based advantages depend on accurate state-value estimates across diverse search states, group-relative advantages do not distinguish steps, penalizing useful ones and reinforcing unnecessary ones, and dense supervision from reward models or LLM graders requires extra models or annotations and inherits the graders' biases. To address these two questions, we present **TDS** (**T**able-driven **D**ual-tier **S**earch), a multi-agent search system that pairs a table-driven search structure with asymmetric credit gating. The structure is a global table of subproblem nodes, shared by training and inference, through which a planner and parallel executors coordinate. Each node keeps a subproblem's answer and a concise evidence summary written by an executor, so the planner can build on any completed result without carrying the search history, and independent subproblems are solved in parallel. We train the executor first, then freeze it and train the planner with the gate. During training, the planner is prompted after each expansion to answer the original question from the current table, and the first answer that matches the reference marks the steps that sufficed. Successful trajectories then reinforce only these steps rather than later unnecessary expansions, and failed trajectories no longer penalize them, which yields step-level credit without a critic or a grader. Under a common training and evaluation protocol, TDS improves seven-benchmark macro-average exact match over the strongest baseline at each scale by 2.2 percentage points (5.7% relative) with Qwen2.5-3B-Instruct and by 1.9 percentage points (4.4% relative) with Qwen2.5-7B-Instruct. Ablations confirm that both the table structure and the asymmetric gate contribute to these gains.

1 INTRODUCTION

Large language model (LLM) agents trained with reinforcement learning (RL) increasingly handle multi-step, tool-assisted tasks, often in multi-agent configurations (Li et al., 2023; Wu et al., 2023; Chen et al., 2025; Jin et al., 2025; Song et al., 2025; Zheng et al., 2025; Zhang et al., 2025). Agentic search exemplifies this setting, requiring a system to decompose a complex question into subproblems, gather evidence over successive steps, and synthesize a final answer (Yao et al., 2022; Trivedi et al., 2023; Shi et al., 2024). The task provides reference answers only for the original question, which serve as the sole source of supervision. Learning from this answer supervision raises two questions (Figure 1). The first concerns how to organize search as intermediate results accumulate. What to retain, select, and aggregate remains unsettled, and the search structure often differs between training and inference. The second concerns how to assign credit to intermediate decisions in multi-agent search when subproblems have no reference answers of their own.

For the first question, prior work commonly organizes intermediate results as linear chains of subqueries and answers (Jin et al., 2025; Chen et al., 2026) or as trees with parent-child dependencies

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

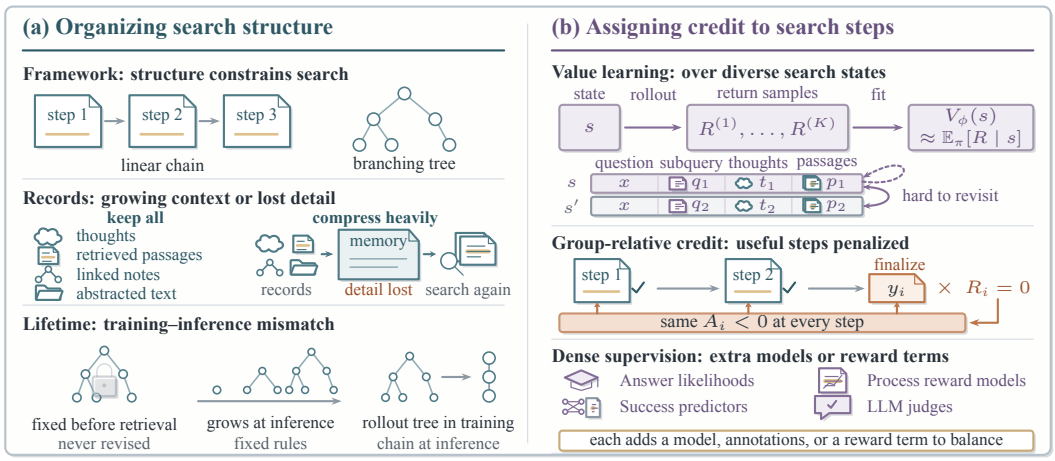


Figure 1: Two main questions in agentic search with a single source of supervision. (a) Organizing search structure. (b) Assigning credit to search steps.

(Shi et al., 2026). In both settings, the information retained in each record varies widely, leaving the balance between sufficiency and brevity unresolved. Some systems keep extensive histories, including intermediate thoughts (Yao et al., 2023; Hao et al., 2023), retrieved passages (Jiapeng et al., 2024), linked notes (Xu et al., 2026), and text organized at different levels of abstraction (Rezazadeh et al., 2025), so the stored context grows rapidly with the search. Others condense each turn into a compressed memory (Yuan et al., 2026), so details dropped at one turn must be retrieved again for later decisions. Structures also differ in when they are built and how long they last. Some remain fixed once retrieval begins (Shi et al., 2026). Others grow during inference under fixed rules without policy training (Jiapeng et al., 2024), while training-time trees organize sampled rollouts but are absent at inference (Ji et al., 2026). Later planning draws on this record for two things, what has been established so far and which earlier result a follow-up subproblem should build on. When this dependency is explicit, completed subproblems become starting points for further search. This calls for a representation that keeps intermediate results concise and reusable while preserving their dependencies, and that is shared by training and inference.

For the second question, a trajectory's outcome arrives only at its end, while the decisions behind it span many planner steps, which makes it difficult to assign credit to individual steps. Proximal policy optimization (PPO), adopted in recent agentic RL for variable rollout segments and single-rollout updates (Li et al., 2026a; Hou et al., 2026), estimates step-level advantages from learned state values, so its credit is only as accurate as a critic that must generalize across the diverse and evolving states of agentic search. Group-relative policy optimization (GRPO) removes the critic by comparing the terminal outcomes of searches for the same question (Shao et al., 2024), but broadcasts one advantage to every step, penalizing useful steps in failed trajectories (Deng et al., 2025) and reinforcing unnecessary expansions in successful ones. Dense supervision from success-probability models trained on annotated pivot steps (Liu et al., 2026), process reward models (Lightman et al., 2024), and LLM-as-a-judge evaluators (Zheng et al., 2023) provides step-level signals but requires models or annotations beyond the original question-answer pairs, and LLM grading carries grader biases (Zheng et al., 2023). Likelihood-based rewards stay within these pairs by scoring per-turn changes in reference-answer likelihood (Xie et al., 2026; Wang et al., 2025a), but add a second reward term whose shaping weight or normalization must be tuned against the outcome reward. Existing step-level credit thus relies on a critic, a grader, or an added reward term. It remains open whether such credit can be derived from the reference answer alone, with the terminal outcome as the sole reward.

To address these two questions, we present **TDS** (**T**able-driven **D**ual-tier **S**earch), a parallel multi-agent search system that pairs a table-driven search structure with asymmetric credit gating (Figure 2). TDS organizes search as a growing subproblem tree stored in a global table, which is the same during training and inference. Each record holds a subproblem, its parent, its answer, and an evidence summary, so records stay concise and reusable, with their dependencies explicit. A high-tier planner extends existing nodes with new subproblems grounded in recorded results or produces the final

answer. Low-tier executors solve the new subproblems in parallel and write their answers and summaries back to the table. TDS first trains the executor and freezes it, then trains the planner with GRPO (Shao et al., 2024), using the terminal answer check as the only reward. After each expansion during training, the planner answers the original question from the updated table, and the first answer that passes the reference check marks the search prefix that sufficed. Asymmetric credit gating reinforces only this prefix in successful trajectories and shields it from the penalty in failed ones, so step-level credit comes from the reference answer alone, without a critic or a grader.

We evaluate TDS on seven open-domain question answering benchmarks spanning single-hop and multi-hop settings (Kwiatkowski et al., 2019; Joshi et al., 2017; Mallen et al., 2023; Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Press et al., 2023), using Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct backbones (Yang et al., 2024). TDS exceeds the strongest baseline in seven-benchmark macro-average EM by 2.2 percentage points (5.7% relative) at 3B and 1.9 percentage points (4.4% relative) at 7B, and ablations confirm the effectiveness of the two components.

We make three contributions.

- We introduce a table-driven search structure that combines parallel execution of independent subproblems within each round with dependency-aware expansion across rounds, using the same search interface during training and inference.
- We introduce asymmetric credit gating, which checks the planner's interim answers from the same table and uses the first passing check and the final outcome to decide which planner steps receive the outcome advantage.
- We introduce TDS, a parallel multi-agent search system built on the table-driven structure and trained with asymmetric credit gating. It outperforms the strongest baseline on seven QA benchmarks at two model scales, with larger gains on multi-hop tasks.

2 Related Work

2.1 Agentic Search Architectures

Search agents organize intermediate evidence through interaction histories, compact memories, or branching structures. ReAct and Search-R1 interleave reasoning and retrieval along a trajectory (Yao et al., 2022; Jin et al., 2025); MemSearcher rewrites a single memory each turn, merging new retrievals with content kept from earlier turns (Yuan et al., 2026). Tree of Reviews expands and prunes retrieval paths during inference, whereas Tree-GRPO organizes training rollouts (Jiapeng et al., 2024; Ji et al., 2026). In Tree-GRPO, each node represents a complete interaction step, and branches contain alternative continuations sampled from a shared history. TDS stores a growing subproblem tree in a global table, preserving answers, evidence summaries, and parent links for reuse and bounded parallel expansion. Here, the tree records dependencies within one search; parent links identify the earlier subproblems on which later ones build.

2.2 Learning and Credit Assignment

Training methods differ in which roles they update and how they assign credit. Dr. MAS jointly optimizes collaborating roles, while AgentFlow trains the planner with other components fixed (Feng et al., 2026; Li et al., 2026b). DeSA separates search and answering objectives across training stages (Wang et al., 2025b). Search-R1's PPO variant uses learned value baselines, while GRPO compares outcomes for the same question (Jin et al., 2025; Shao et al., 2024); Tree-GRPO further compares branches within and across rollout trees (Ji et al., 2026). MemSearcher applies a shared trajectory-level advantage across turns with changing memory inputs (Yuan et al., 2026). For intermediate supervision, TIPS uses reference-answer likelihood changes under a policy checkpoint, PiCA learns success estimators from process annotations, and M-ASK uses an answering role to provide turn-level feedback (Xie et al., 2026; Liu et al., 2026; Chen et al., 2026). IGPO combines outcome rewards with turn-level rewards derived from changes in reference-answer log probability (Wang et al., 2025a). TDS freezes the pretrained executor and asymmetrically gates the planner's group-relative outcome advantage using the first passing check of a generated interim answer and the final outcome.

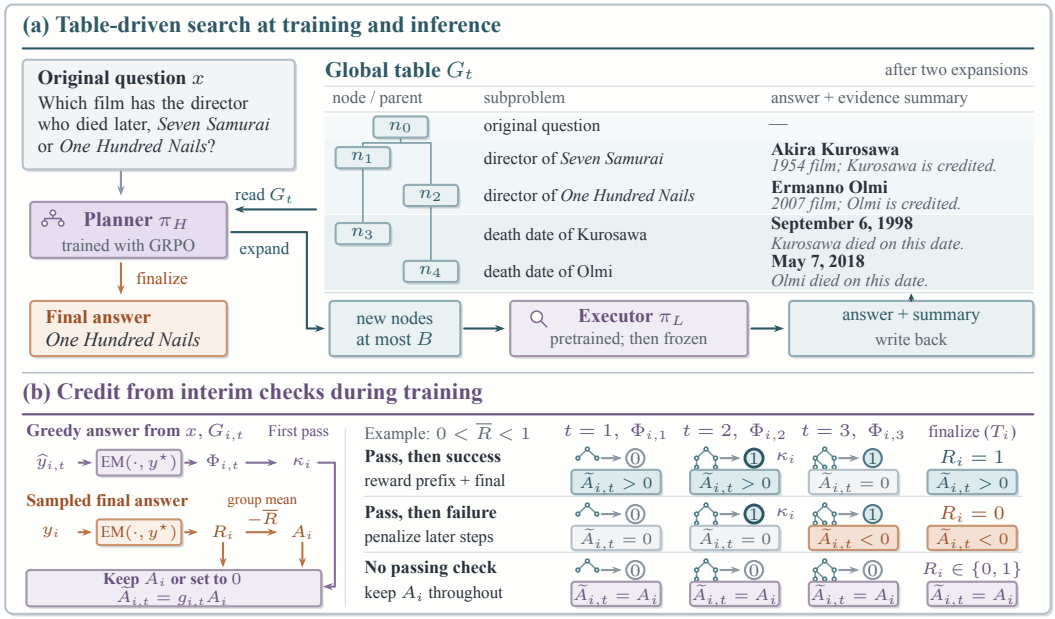


Figure 2: TDS overview. (a) Table-driven parallel search with cross-round subproblem dependencies. (b) Asymmetric credit gating based on the first passing interim check and final outcome.

3 METHOD

3.1 PRELIMINARIES

We study agentic search for question answering in a multi-agent setting. Given a question x, a planner coordinates executors to retrieve evidence and solve subproblems. A trajectory τ_i records the search decisions, retrieved passages, and intermediate outputs. expand steps occur at $1 \le t < T_i$, and the finalize step at T_i produces the final answer y_i.

We denote the reference answer for x by $y^\star$. The exact-match indicator $\mathrm{EM}(y, y^\star)$ equals 1 if the prediction matches $y^\star$ after normalization (Appendix A) and 0 otherwise. The terminal return is $R_i = \mathrm{EM}(y_i, y^\star)$, and benchmark EM averages this indicator over questions.

We train the planner with group-relative policy optimization (GRPO) (Shao et al., 2024). For each question, we sample M trajectories $\{\tau_i\}_{i=1}^{M}$ and compute their mean-centered outcome advantages

$$\overline{R} = \frac{1}{M} \sum_{j=1}^{M} R_j, \qquad A_i = R_i - \overline{R}. \tag{1}$$

We maximize the entropy-regularized clipped objective

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{\mathcal{B}}[\ell_{\mathrm{clip}}(\rho_\theta(z,h), A_i) - \beta_{\mathrm{KL}}\, d_\theta(z,h) + \lambda_{\mathrm{ent}}\, \mathcal{H}(\pi_\theta(\cdot \mid h))]\,. \tag{2}$$

Here, $\rho_\theta(z,h) = \pi_\theta(z \mid h)/\pi_{\mathrm{old}}(z \mid h)$ is the token probability ratio given context h, and ℓ_{clip} is the clipped policy surrogate. The KL penalty d_θ relative to the reference policy and the token entropy $\mathcal{H}$ are weighted by β_{KL} and λ_{ent}, respectively. We average over the planner response tokens included in batch $\mathcal{B}$; clipping and masking details are given in Appendix B. Section 3.3 defines the step-specific advantages used in the clipped term.

3.2 TABLE-DRIVEN SEARCH

TDS stores its growing subproblem tree in a global table G_t, with one row per node. Each node n_v records its unique identifier id_v, parent identifier p_v, subproblem q_v, answer a_v, and evidence summary s_v,

$$G_t = \{n_v\}_{v=0}^{N_t}, \qquad n_v = (\mathrm{id}_v, p_v, q_v, a_v, s_v). \tag{3}$$

The root stores the original question, $q_0 = x$, and N_t denotes the number of subproblem nodes. Each subproblem node stores a short answer in a_v and a one- or two-sentence summary of the retrieved passages in s_v. This summary keeps key facts, including relevant candidates and dates, for later search and final synthesis. Completed nodes stay in the table, so the planner can select them by identifier as parents of new subproblems. Parent links encode the decomposition as the tree grows.

At step t, the planner π_H reads the original question and the current table G_{t-1}. It chooses an expand step or a $\mathrm{finalize}$ step,

$$a_t^H \in \{\mathrm{finalize}(y)\} \cup \left\{\mathrm{expand}\Big(\{(p_{t,j}, q_{t,j})\}_{j=1}^{k_t}\Big) : 1 \le k_t \le B\right\}. \tag{4}$$

In an expand step, the planner proposes k_t new subproblems, each attached to a parent already in G_{t-1}. We allow at most $B = 3$ subproblems per step. Subproblems proposed in one round draw only on existing records, so they can be solved independently and in parallel. When a subproblem builds on an earlier result, the planner writes the relevant facts into the subproblem and names its parent node. The returned answers and summaries become records for later planning, from which the planner can extend a completed subproblem or return to another branch. The table thus links independent execution within a round to dependent follow-up questions across rounds.

Executors share one policy π_L for search and answering, with role-specific prompts and parameters separate from the planner's. Each executor receives the original question and its assigned subproblem, formulates a retrieval query, and uses the returned passages to produce a short answer and an evidence summary. These outputs update G_{t-1} to G_t, after which the planner chooses another expansion or synthesizes the final answer. Training and inference use the same table and action interface, so policy learning shapes where search continues, what information is acquired, and when the trajectory ends.

3.3 TRAINING WITH ASYMMETRIC CREDIT GATING

Motivated by the use of PPO in recent agentic training, including the GLM series (Z.ai, 2026; Hou et al., 2026), we explored a learned value baseline for the planner. However, in the evaluated PPO setting, the critic explains little of the variation in its training return targets (Section 4.3). In TDS, we derive group-relative advantages from the observed outcomes of searches for the same question (Eq. (1)), and use the first passing interim check to allocate these advantages across planner steps.

During training, the planner generates an interim answer after each completed expansion. Once the executors have filled the new rows of $G_{i,t}$, the planner answers the original question using a direct-answer prompt and greedy decoding, producing $\widehat{y}_{i,t}$. We apply the same reference-based EM check used for the final return to obtain $\Phi_{i,t}$. This check measures whether the current policy can answer correctly given the updated table. The interim response is not added to the table, and search resumes from $G_{i,t}$. Finalization samples a fresh answer, so the final outcome is determined separately from the interim checks. For valid interim responses, we write

$$\begin{aligned} \widehat{y}_{i,t} &= \mathrm{Greedy}(\pi_H; x, G_{i,t}), && 1 \le t < T_i, \\ \Phi_{i,t} &= \mathrm{EM}(\widehat{y}_{i,t}, y^\star), && 1 \le t < T_i. \end{aligned} \tag{5}$$

We use the first passing interim check, at step κ_i, as the boundary for allocating the outcome advantage (Figure 2(b)). On final success, expansions up to and including κ_i receive A_i and later expansions receive zero advantage. On final failure, the same prefix receives zero advantage, which shields it from the outcome penalty, and later expansions receive A_i. The interim check thus selects which steps receive the outcome advantage and leaves the terminal reward unchanged. The $\mathrm{finalize}$ step

Table 1: EM (%) across seven QA benchmarks. Avg. is the unweighted mean within each task group. **Bold** and underlining mark the best and second-best results per column at each scale.

	Single-hop				Multi-hop				
Method	NQ	TriviaQA	PopQA	**Avg.**	HotpotQA	2Wiki	MuSiQue	Bamboogle	**Avg.**
Qwen2.5-3B-Instruct									
TDS	**42.5**	**59.4**	**40.7**	**47.5**	**40.5**	**39.4**	**17.4**	**41.6**	**34.7**
IRCoT	13.7	23.2	14.1	17.0	10.1	6.3	1.7	5.6	5.9
Search-R1	32.7	54.1	35.1	40.6	33.2	31.8	10.4	26.4	25.5
DeSA	35.5	57.1	34.2	42.3	35.2	34.4	13.4	31.2	28.6
MemSearcher	<u>40.6</u>	58.5	37.0	45.4	35.6	36.6	14.2	37.6	31.0
Dr. MAS	38.8	57.4	32.4	42.9	33.0	34.3	9.8	28.8	26.5
Tree-GRPO	40.4	<u>58.6</u>	<u>38.0</u>	<u>45.7</u>	<u>37.9</u>	<u>37.1</u>	<u>15.9</u>	<u>38.4</u>	<u>32.3</u>
Qwen2.5-7B-Instruct									
TDS	**47.9**	**62.1**	**44.6**	**51.5**	**43.8**	**44.1**	**22.0**	**45.6**	**38.9**
IRCoT	18.7	36.2	11.5	22.1	21.8	14.6	1.2	4.8	10.6
Search-R1	40.2	61.0	38.3	46.5	31.6	39.9	12.7	37.6	30.5
DeSA	42.8	60.7	40.0	47.8	<u>42.2</u>	37.0	19.4	40.0	34.7
MemSearcher	<u>47.4</u>	61.2	<u>42.6</u>	<u>50.4</u>	41.3	<u>42.8</u>	17.0	<u>44.8</u>	36.5
Dr. MAS	43.5	60.1	41.2	48.3	37.0	40.0	16.2	40.8	33.5
Tree-GRPO	46.5	<u>61.7</u>	41.3	49.8	41.4	42.7	<u>20.2</u>	42.4	<u>36.7</u>
Larger-model references with IRCoT									
Qwen3-32B	29.1	61.6	33.8	41.5	33.1	33.7	12.3	24.0	25.8
Llama3.1-70B	31.2	64.0	39.7	45.0	37.1	33.6	13.8	30.4	28.7

always receives A_i. If no interim check passes, every expansion receives A_i and $\kappa_i = T_i$. Formally, the boundary is defined by

$$\kappa_i = \min(\{t \mid 1 \le t < T_i,\ \Phi_{i,t} = 1\} \cup \{T_i\})\,. \tag{6}$$

An asymmetric gate implements these rules by assigning each planner step the advantage

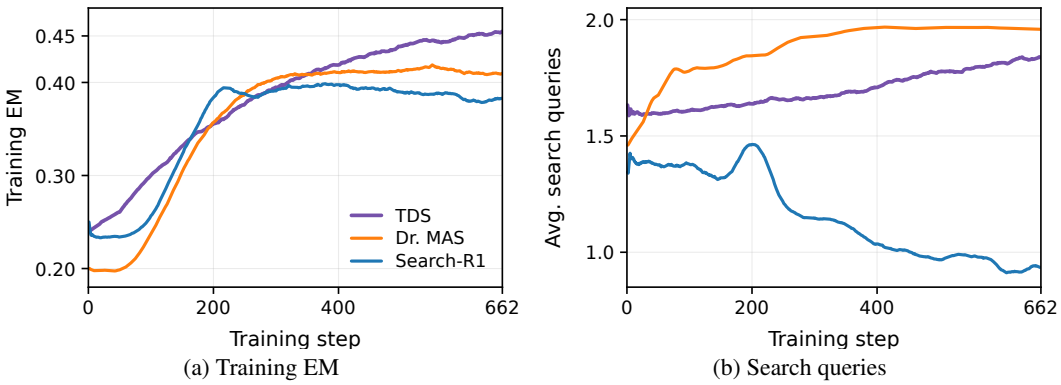
$$\widetilde{A}_{i,t} = g_{i,t} A_i, \qquad g_{i,t} = \begin{cases} 1, & t = T_i \text{ or } \kappa_i = T_i, \\ R_i, & t \le \kappa_i < T_i, \\ 1 - R_i, & \kappa_i < t < T_i. \end{cases} \tag{7}$$

We first train the shared executor policy π_L with GRPO on the single-hop NQ subset of the full training set, treating each question as a subproblem. Each rollout retrieves evidence and produces an answer and summary. The answer check provides outcome supervision for both search and answering actions.

We then freeze π_L and train the planner π_H using complete search trajectories for questions from the full training set. Executors continue to solve the planner's subproblems and record their answers and evidence summaries in the table. For each planner response at step t, we assign $\widetilde{A}_{i,t}$ from Eq. (7) to all of its generated tokens, replacing A_i in Eq. (2). The gate is computed from the collected trajectory and held fixed during the policy update. Gating leaves the token-averaging denominator and the entropy and KL terms unchanged. Training settings are provided in Appendix B.

4 Experiments

We compare against Search-R1 (PPO) (Jin et al., 2025), DeSA (Wang et al., 2025b), MemSearcher (Yuan et al., 2026), Dr. MAS (Feng et al., 2026), Tree-GRPO (Ji et al., 2026), and IRCoT (Trivedi et al., 2023). All trained baselines are retrained and evaluated under the same protocol as TDS, with



(a) Training EM (b) Search queries

Figure 3: Training dynamics of TDS, Dr. MAS, and Search-R1 with Qwen2.5-7B-Instruct. Panels show (a) training EM and (b) mean search queries per question, using 50-step trailing averages.

Table 2: EM (%) for 3B ablations and an untrained reference. Avg. is the unweighted task-group mean. **Bold** and underlining mark the best and second-best results per column.

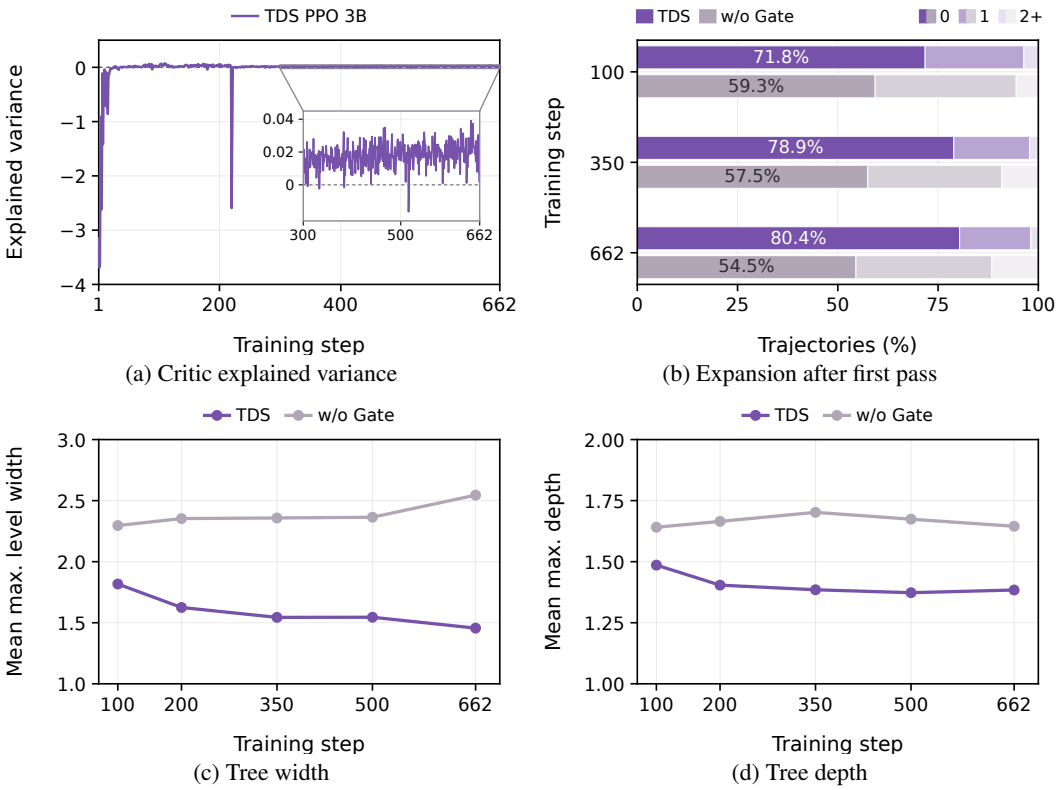
	Single-hop				Multi-hop				
Variant	NQ	TriviaQA	PopQA	**Avg.**	HotpotQA	2Wiki	MuSiQue	Bamboogle	**Avg.**
TDS	<u>42.5</u>	<u>59.4</u>	**40.7**	<u>47.5</u>	**40.5**	**39.4**	**17.4**	**41.6**	**34.7**
Table structure and evidence ablations									
w/o Parent Links	**43.8**	**60.7**	40.1	**48.2**	<u>38.9</u>	38.1	11.8	<u>40.0</u>	32.2
w/o Table	34.5	52.1	36.8	41.1	30.8	27.9	11.3	29.6	24.9
w/o Summary	32.8	51.7	32.7	39.1	30.5	29.2	10.7	25.6	24.0
Asymmetric gating ablations									
w/o Gate	38.5	57.4	39.3	45.1	38.4	37.4	<u>16.9</u>	38.4	32.8
Success-side only	37.3	56.8	36.1	43.4	36.4	31.3	9.2	32.8	27.4
Failure-side only	39.8	58.0	<u>40.3</u>	46.0	38.3	<u>38.9</u>	16.1	<u>40.0</u>	<u>33.3</u>
Training ablations									
TDS w/o RL	28.9	40.2	30.3	33.1	21.5	26.7	6.1	26.4	20.2
TDS PPO	30.5	51.9	34.7	39.0	27.0	27.9	12.0	29.6	24.1
Joint (trainable executor)	37.4	54.0	36.7	42.7	38.4	34.5	13.7	33.6	30.1

the same training data (NQ and HotpotQA), a batch size of 256 questions, and 662 training steps. Additional experimental details are provided in Appendices A, B, and C. We evaluate these methods with Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024) on seven single-hop and multi-hop QA benchmarks (Table 1). We report exact match (EM).

4.1 Main Results

Table 1 shows that TDS achieves the highest EM on all seven benchmarks among methods using the same backbone, at both 3B and 7B. Its seven-benchmark macro-average EM reaches 40.2 at 3B and 44.3 at 7B, exceeding the strongest baseline at each scale by 2.2 percentage points (5.7% relative) and 1.9 percentage points (4.4% relative), respectively. Relative to these baselines, the multi-hop gains reach 2.4 percentage points at both scales, exceeding the single-hop gains of 1.9 at 3B and 1.1 at 7B. The gains are largest on questions that require evidence from several subproblems, where persistent intermediate records serve successive search decisions and final synthesis. TDS at 3B also exceeds the IRCoT references with Qwen3-32B and Llama3.1-70B in both task groups.

Training behavior. Figure 3 tracks training EM and retrieval usage together. TDS improves answer accuracy steadily while issuing more search queries, learning to draw on external evidence more



(a) Critic explained variance
(b) Expansion after first pass
(c) Tree width
(d) Tree depth

Figure 4: Optimization and search behavior at 3B. (a) TDS PPO 3B critic explained variance. (b) Valid expansion rounds after the first passing interim check. (c,d) Per-tree maxima averaged over questions (root depth 0).

actively. Search-R1 (PPO) follows a different course, with retrieval activity falling in later training as EM plateaus and then declines. The comparison with Dr. MAS, another multi-agent system, shows the same coordination. In later training, TDS reaches higher EM with fewer search queries per question, counting retrieval by all agents.

4.2 Ablation Studies

Table 2 examines table structure and retained evidence, asymmetric gating, and policy training with the 3B backbone. The prompts and retrieval budget remain fixed except for changes required by each variant's representation or training rule.

Table structure and evidence. We first ablate table structure and retained evidence with the planner's credit-assignment rule held fixed. The w/o Table variant keeps subproblems, answers, and summaries as chronological text and reaches 31.9 macro-average EM. Storing explicit node records (w/o Parent Links) raises EM to 39.1, and restoring parent links with parent-indexed expansion gives the full model's 40.2. Parent links improve all four multi-hop benchmarks, raising their average EM from 32.2 to 34.7. The planner thus benefits from explicit records it can revisit and extend. The content of those records also matters. Removing evidence summaries lowers EM to 30.5 even though node answers and table structure are kept, below the 31.9 of chronological text that keeps the summaries.

Asymmetric gating. The gating ablations show that the effect of success-side gating depends on whether failure-side protection is enabled. The w/o Gate variant broadcasts the outcome advantage to every planner step; each one-sided variant applies the gate only to its named outcome and broadcasts the advantage on the other. Failure-side gating alone raises macro-average EM from 38.0 to 38.8, and adding the success side further raises it to 40.2. Success-side gating alone, however, lowers EM to

34.3. Eq. (7) explains the reversal. Success-side gating alone gives expansions after the first pass no positive credit while every expansion in a failed trajectory is still penalized. The gate therefore changes the ratio of positive to negative samples on expansion steps. The multi-hop benchmarks need more than one round and suffer most. Their average falls from 32.8 to 27.4 while the single-hop average falls from 45.1 to 43.4. Failure-side protection removes the penalty from expansions that reach a correct interim answer, so an expansion is penalized only after the first pass or in a trajectory that never passes.

Policy training. TDS w/o RL keeps the search interface but drops both executor pretraining and planner training, which lowers macro-average EM to 25.7. Joint uses the same planner initialization, pretrained executor, and gate, but continues updating the executor during planner optimization for 662 steps and reaches 35.5. Executor responses receive group-relative advantages from the final outcome, with gating applied only to the planner. Freezing the executor removes one source of non-stationarity, so the planner learns against a stable policy for subproblem answers and summaries. The TDS PPO variant, which uses a learned critic, is examined in Section 4.3.

4.3 Optimization and Search Behavior

Figure 4 examines optimization and search behavior with the 3B backbone. Panel (a) plots the PPO critic's explained variance logged during training. Panels (b–d) evaluate saved checkpoints with greedy decoding on a fixed set of 1,000 test questions, 500 each from NQ and HotpotQA test sets, aligned with the selected training benchmarks (Appendix B).

PPO planner critic. Following the PPO-based agent training discussed in Section 3.3, the TDS PPO variant uses a learned value baseline to estimate advantages for planner updates. It reaches 30.5 macro-average EM, compared with 38.0 for w/o Gate and 40.2 for TDS (Table 2). The 3B critic's explained variance rises from negative values but stays near zero in later training (Figure 4(a)), so it captures little of the variation in its return targets.

Asymmetric gating and post-pass expansion. Gating also changes how search continues after the first passing interim check. Among trajectories with a passing interim check and a correct final answer, 80.4% of TDS trajectories make no further valid expansion after the first pass at step 662, compared with 54.5% for w/o Gate (Figure 4(b)). Broadcasting gives expansions after the pass the positive advantage of a successful trajectory. The w/o Gate policy expands after the pass more often as training proceeds. The gate withholds this credit, and TDS expands after the pass less often as training proceeds.

Search structure. Search trees also become narrower and shallower. Under the same expansion limits, TDS's mean maximum level width falls during training to 1.46 at step 662, against 2.55 for w/o Gate (Figure 4(c)). The trees of w/o Gate stay wide throughout and widen late in training. Depth follows the same pattern (Figure 4(d)). The w/o Gate policy keeps deeper trees throughout training, with a mean maximum depth between 1.64 and 1.70, whereas TDS falls to 1.40 by step 200 and stays near 1.38 thereafter. TDS thus reaches higher EM with search trees that are both narrower and shallower.

5 Conclusion

TDS pairs a table-driven search structure with asymmetric credit gating to learn parallel multi-agent search from answer supervision for the original question alone. The global table keeps subproblem answers, evidence summaries, and explicit dependencies, supporting parallel execution within rounds and continued planning across rounds through one interface for training and inference. The gate checks interim answers from the same table and uses the first passing check to decide which planner steps receive the outcome advantage. Across seven QA benchmarks at 3B and 7B, TDS improves macro-average exact match over the strongest baseline, with larger gains on multi-hop tasks. Ablations show the value of keeping evidence alongside reusable subproblem records and of pairing success-side gating with failure-side protection. These findings argue for designing the search representation and the learning signal jointly when the final answer is the only supervision.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on the methodology and experiments, to assist with implementing the method and its evaluation, to help interpret results, and to assist with translation. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao. Mao-arag: Multi-agent orchestration for adaptive retrieval-augmented generation. *arXiv preprint arXiv:2508.01005*, 2025.

Yiqun Chen, Lingyong Yan, Zixuan Yang, Erhan Zhang, Jiashu Zhao, Shuaiqiang Wang, Dawei Yin, and Jiaxin Mao. Beyond monolithic architectures: A multi-agent search and knowledge optimization framework for agentic search. *arXiv preprint arXiv:2601.04703*, 2026.

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. On group relative policy optimization collapse in agent search: The lazy likelihood-displacement. *arXiv preprint arXiv:2512.04220*, 2025.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. mas: Stable reinforcement learning for multi-agent llm systems. *arXiv preprint arXiv:2602.08847*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 87362–87388, 2026.

Li Jiapeng, Liu Runze, Li Yabo, Zhou Tong, Li Mingling, and Chen Xiang. Tree of reviews: A tree-based dynamic iterative retrieval framework for multi-hop question answering. *arXiv preprint arXiv:2404.14464*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Yujiang Li, Zhenyu Hou, Yi Jing, Jie Tang, and Yuxiao Dong. Compactionrl: Reinforcement learning with context compaction for long-horizon agents. *arXiv preprint arXiv:2607.05378*, 2026a.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Y Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations*, volume 2026, pp. 50524–50570, 2026b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Dongyi Liu, Yifan Niu, Qinwen Wang, Han Xiao, and Jia Li. Pica: Pivot-based credit assignment for search agentic reinforcement learning. *arXiv preprint arXiv:2605.09287*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. In *International Conference on Learning Representations*, volume 2025, pp. 990–1023, 2025.

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

Yuling Shi, Maolin Sun, Zijun Liu, Mo Yang, Yixiong Fang, Tianran Sun, and Xiaodong Gu. Reasoning in trees: Improving retrieval-augmented generation for multi-hop question answering. In *Companion Proceedings of the ACM Web Conference 2026*, pp. 626–635, 2026.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7339–7353, 2024.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn llm agents. *arXiv e-prints*, pp. arXiv–2510, 2025a.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Yiding Wang, Zhepei Wei, Xinyu Zhu, and Yu Meng. Beyond outcome reward: Decoupling search and answering improves llm agents. *arXiv preprint arXiv:2510.04695*, 2025b.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Yutao Xie, Nathaniel Thomas, Nick Hansen, Yang Fu, Li Li, and Xiaolong Wang. Tips: Turn-level information-potential reward shaping for search-augmented llms. In *International Conference on Learning Representations*, volume 2026, pp. 156549–156584, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL `https://arxiv.org/abs/2412.15115`.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. Memsearcher: Iterative memory integration for search agent via end-to-end reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14965–14977, 2026.

Z.ai. Glm-5.2. `https://z.ai/blog/glm-5.2`, 2026. Accessed August 2026.

Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jiexiang Wang, Sheng Guo, Bo Zheng, Haobo Wang, et al. Lets: Learning to think-and-search via process-and-outcome reward hybridization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5109–5122, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21816–21841, 2025.

A Training and Evaluation Protocol

A.1 Data and Evaluation

The single-hop benchmarks are Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023); the multi-hop benchmarks are HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). Task-group averages are unweighted means within these groups.

Following Dr. MAS (Feng et al., 2026), the training set contains 169,615 questions: 79,168 from NQ and 90,447 from HotpotQA. Executor pretraining uses the NQ subset, and planner training uses the full mixture with the executor held fixed. Table 3 lists the dataset sizes.

At each model scale, the compared methods use Qwen2.5-3B-Instruct or Qwen2.5-7B-Instruct. Final evaluation uses one greedily decoded trajectory per question. Exact match (EM) is computed after lowercasing, removing punctuation and the articles *a*, *an*, and *the*, and normalizing whitespace. A prediction is correct if it matches any reference answer after normalization. We report EM for each benchmark and the unweighted mean of the seven benchmark scores.

Table 3: Training and test dataset sizes.

Benchmark	Training	Test
NQ	79,168	3,610
TriviaQA	0	11,313
PopQA	0	14,267
HotpotQA	90,447	7,405
2Wiki	0	12,576
MuSiQue	0	2,417
Bamboogle	0	125
Total	169,615	51,713

A.2 Implementations and Training Settings

The Search-R1, DeSA, Dr. MAS, Tree-GRPO, and MemSearcher results are obtained from implementations retrained and evaluated under our experimental protocol. Training and evaluation use a platform with 16 PPUs. Due to the platform's limited support for tensor parallelism, the tensor parallel size is set to 1 for all methods. We use a common training batch size of 256 questions and the stage budgets in Table 4, while keeping each method's core optimization and sampling mechanisms. Batch size counts input questions; the number of sampled trajectories follows the method-specific training procedure.

Search-R1 uses PPO following the original study's default and its reported training stability relative to GRPO (Jin et al., 2025). DeSA keeps its two-stage progression from retrieval recall to answer correctness (Wang et al., 2025b). Detailed TDS optimization settings are provided in Appendix B.

Relative to the original experimental configurations, Tree-GRPO uses the common NQ–HotpotQA mixture and training budget in our experiments, whereas its original configurations train on NQ or HotpotQA separately (Ji et al., 2026). MemSearcher uses the common retrieval setting specified below, with three passages per query compared with ten in its original configuration (Yuan et al., 2026).

IRCoT alternates reasoning, retrieval, and answering with fixed model weights. IRCoT with Qwen3-32B and Llama3.1-70B provides larger-model references (Yang et al., 2025; Grattafiori et al., 2024).

A.3 Retrieval, Interaction, and Generation

All compared learned methods use the same December 2018 Wikipedia corpus and E5-base-v2 retriever (Wang et al., 2022), returning three passages per query.

Table 4: Training stages and method-specific settings, shared by the 3B and 7B models. Budgets count training updates within each stage.

Method	Stages and updates	Optimization and sampling	Policy structure
TDS	Executor: 150; planner: 662	Executor GRPO; planner group-relative updates with asymmetric gating; 5 planner trajectories/question	Separate planner and executor; Search/Answer share executor
Search-R1	662	PPO; 1 trajectory/question	One actor policy with a learned critic
DeSA	Retrieval recall: 662; answer correctness: 662	Two-stage optimization	One policy
Dr. MAS	662	GRPO; 5 trajectories/question	Three independent role policies
Tree-GRPO	662	Tree sampling: $M = 2$, $N = 2$, $L = 1$, $K = 3$; intra-tree and inter-tree relative advantages	One policy
MemSearcher	662	GRPO; 5 trajectories/question; 1,024-token memory	One policy

In TDS, the planner reads the original question and the current global table. Each executor receives the original question and its assigned subproblem, generates a retrieval query, and produces a short answer and an evidence summary from the returned passages. The answers and summaries fill the corresponding table rows before the next planning decision.

TDS allows up to four expansion rounds, followed by finalization. Each model call uses at most 4,096 input tokens and generates at most 800 tokens. The compared learned methods use the same round limit and per-call input and output length limits during training and evaluation.

During planner training, the planner samples with temperature 1 and top-$p = 1$, while the frozen executor uses greedy decoding. Interim answers also use greedy decoding. Planner responses use a `<think>` prefill.

A.4 Abbreviated Prompts

Tables 5 and 6 summarize the role instructions and response formats. Repeated formatting constraints, few-shot examples, and the contents of the current table are omitted from these summaries; the full templates are part of the implementation.

B Optimization and Diagnostic Details

B.1 Optimization Settings

Executor pretraining uses GRPO on NQ, with each rollout comprising a retrieval query followed by an answer and evidence summary. The shared executor policy is updated from groups of rollouts for the same question, using the answer-check outcome as supervision. Planner training uses the trajectory advantages in Eq. (1), without normalization by the group standard deviation. Appendix A.2 specifies the question batch size, trajectory count, and stage budgets; Table 7 gives the remaining planner optimization settings. We use full-parameter updates in bfloat16 with FSDP2 and vLLM generation.

The planner training loader drops an incomplete final batch. Its budget therefore corresponds to 169,472 question presentations and 847,360 sampled trajectories before masking. The minibatch size adapts to the number of planner response sequences generated in the batch.

Table 5: Abbreviated role instructions.

Role	Input and instruction
Planner	Read the original question and current table. If the table is sufficient, synthesize a short final answer. Otherwise, propose 1–3 atomic subquestions. Use parent 0 for independent branches and existing nodes for dependent subquestions. Do not repeat recorded subquestions. Write subquestions and the final answer in English.
Search	Read the original question and assigned subquestion. Produce one focused retrieval query for the subquestion, using the original question as context.
Answer	Read the original question, subquestion, and retrieved passages. Return the exact short answer to the subquestion and a non-empty English evidence summary of one or two sentences.
Interim answer / finalization	Read the original question and current table. Synthesize a short answer in English from the recorded answers and summaries. Do not include an expansion action in this response.

Table 6: Response formats following `<think>...</think>`. Parent identifiers refer to existing table nodes.

Action	Output format
Expansion	`<expand><node parent="id">subquestion</node> ...</expand>`
Final / interim answer	`<final><answer>short answer</answer></final>`
Search	`<search>query</search>`
Answer	`<answer>short answer</answer> <summary>evidence summary</summary>`

Table 7: Planner training settings, shared by the 3B and 7B models.

Setting	Value
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate	10^{-6}, constant, no warmup
Weight decay	0.01
Policy-update epochs per batch	1
Minibatch updates per training step	1, with adaptive minibatch sizing
Microbatch per device	4 response sequences
Clipping range ϵ / dual-clip bound c	0.2 / 3
Gradient norm limit	1
KL coefficient β_{KL}	0.001
Entropy coefficient λ_{ent}	0.001
Loss aggregation	Mean over retained response tokens

B.2 Policy Objective and Masking

For the token probability ratio ρ and advantage A, the clipped surrogate is

$$s_\epsilon(\rho, A) = \min\{\rho A, \mathrm{clip}(\rho, 1-\epsilon, 1+\epsilon)A\}\,,$$
$$\ell_{\mathrm{clip}}(\rho, A) = \begin{cases} s_\epsilon(\rho, A), & A \geq 0, \\ \max\{s_\epsilon(\rho, A), cA\}\,, & A < 0, \end{cases} \tag{8}$$

where ϵ sets the clipping range and $c > 1$ sets the additional bound for negative advantages. Each token in a planner response at step t uses $A = \widetilde{A}_{i,t}$ from Eq. (7). The sampled reference-policy penalty is $d_\theta(z, h) = \mathrm{clip}(u - \log u - 1, -10, 10)$, where $u = \pi_{\mathrm{ref}}(z \mid h)/\pi_\theta(z \mid h)$. The entropy term is computed from the full next-token distribution. The KL penalty is applied in the planner loss; the task return remains the terminal answer check.

Loss masking. Only planner responses for expand and $\mathrm{finalize}$ steps enter the planner loss. Interim answers are neither added to the table nor used as policy-update samples. If an interim response fails format validation, we carry the preceding check value forward (initially 0), leaving the first passing step unchanged. Groups whose outcome advantages are all zero are excluded before gating. For the remaining groups, response tokens assigned zero advantage by the gate still contribute to the loss denominator, entropy, and KL terms.

B.3 Ablation and Diagnostic Settings

The gating variants share the pretrained executor, planner initialization, interim-answer generation, and optimization settings, as well as the expansion and generation limits in Appendix A.3.

Critic diagnostic. The TDS PPO variant initializes both the planner and critic from Qwen2.5-3B-Instruct and updates both from the first training step, with no critic-only warmup. For the 3B PPO run in Figure 4(a), explained variance measures how well the critic predicts its return target $\widehat{G}_t$:

$$\mathrm{EV} = 1 - \frac{\mathrm{Var}(\widehat{G}_t - V(G_{t-1}))}{\mathrm{Var}(\widehat{G}_t) + 10^{-5}}. \tag{9}$$

The critic predicts one value before each planner action, and generalized advantage estimation is computed across planner turns within a trajectory. The value loss and explained variance use the same valid planner-state positions. The GRPO planner does not use a critic.

Search statistics. Panels (b–d) of Figure 4 use a fixed diagnostic set of 1,000 questions drawn from the test sets of NQ and HotpotQA, the two training benchmarks (500 each). Each checkpoint is run on this set under the evaluation configuration with one greedily decoded trajectory per question. Interim checks are computed as in training.

An expansion round contains one planner expansion and the execution of its proposed subproblems. Additional expansions count rounds that add valid, non-rejected nodes after the first passing check and before finalization; finalization, invalid actions with no new nodes, and rejected duplicate nodes are excluded. Panel (b) uses the trajectories with a correct final answer and a passing interim check and groups the counts into 0, 1, and at least 2 rounds at checkpoints 100, 350, and 662. For panels (c,d), we compute the maximum level width and maximum tree depth of each search tree, with root depth 0, and average over all 1,000 questions at checkpoints 100, 200, 350, 500, and 662.

C Computational Cost

Table 8: Training and evaluation times (hours). TDS training is reported separately for executor pretraining and planner training. Dr. MAS uses joint training. Evaluation is measured in separate runs.

	Qwen2.5-3B-Instruct		*Qwen2.5-7B-Instruct*	
	Dr. MAS	TDS	Dr. MAS	TDS
Training	99.2	Executor: 8.9 Planner: 78.1	168.6	Executor: 18.4 Planner: 143.9
Evaluation	39.9	38.2	52.0	39.7