HYPERMCTS: HYPERGRAPH-AUGMENTED MCTS FOR LONG-HORIZON LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tasks require large language model (LLM) agents to coordinate decisions under constraints that span an entire solution. Monte Carlo Tree Search (MCTS) offers a promising approach to test-time scaling by exploring alternative action trajectories, but model computation and environment interaction make search costly. Efficient search therefore requires effective reuse of trajectory feedback. Standard MCTS maintains prefix-specific statistics, without explicitly accumulating outcomes for decision groups that recur across different paths. To fill this gap, we propose **HyperMCTS**, a training-free method that augments an ordered MCTS tree with a cross-trajectory hypergraph. Hyperedges represent groups of canonical decisions and accumulate their observed returns within the current task. Our hypergraph-guided **HyperUCT** selection rule aggregates evidence from overlapping hyperedges into an action prior, allowing outcomes collected under one prefix to inform selection under another while preserving execution histories in the tree. On DeepPlanning, HyperMCTS improves average planning accuracy by 2.3–7.3 percentage points over the strongest baseline for each of three backbone models. It enables Qwen3.6-27B to outperform Claude Opus 4.6 (max) on Shopping Planning, while achieving higher accuracy with fewer LLM calls and output tokens than the evaluated MCTS-based baselines. SealQA experiments further demonstrate improvements in question answering.

1 INTRODUCTION

Large language model (LLM) agents are increasingly studied for long-horizon tasks such as web navigation, software engineering, and travel or shopping planning (Zhou et al., 2024b; Antoniades et al., 2025; Zhang et al., 2026). These tasks require tool interactions and decisions whose consequences extend across the complete solution: individually plausible choices may jointly violate a budget or conflict in time. Figure 1(a) illustrates this difficulty: a DeepPlanning itinerary attains a Composite Score of 93.8% but fails because an attraction is scheduled twice.

Test-time scaling (TTS) allocates additional inference computation to improve task solving (Snell et al., 2024; Zhang et al., 2025). Chain-of-thought encourages intermediate reasoning (Wei et al., 2022), and self-consistency aggregates answers from sampled reasoning paths (Wang et al., 2023). For interactive agents, ReAct interleaves reasoning and interaction (Yao et al., 2023b), while Reflexion revises attempts using feedback (Shinn et al., 2023). These approaches support deliberation and revision without explicitly maintaining a search tree of alternative action histories.

Structured methods organize reasoning into thought trees or graphs (Yao et al., 2023a; Besta et al., 2024). HyperTree Planning (Gui et al., 2025) uses hyperedges for hierarchical subtask decomposition, but guides search through LLM judgments rather than value estimates updated from rollout returns. Monte Carlo Tree Search (MCTS) evaluates candidate actions through simulated continuations and backs up their returns to guide subsequent search, balancing exploration with exploitation (Kocsis & Szepesvári, 2006; Browne et al., 2012). Applications include world-model reasoning in RAP (Hao et al., 2023), self-refinement in MCTSr (Zhang et al., 2024), trajectory verification in rStar (Qi et al., 2025), and value-guided decoding in PPO-MCTS (Liu et al., 2024). For interactive agents, LATS incorporates environment feedback and reflection (Zhou et al., 2024a); ExACT demonstrates search-budget scaling for long-horizon web agents (Yu et al., 2025); SWE-Search explores inference-time scaling for software engineering (Antoniades et al., 2025); and FLARE com-

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.

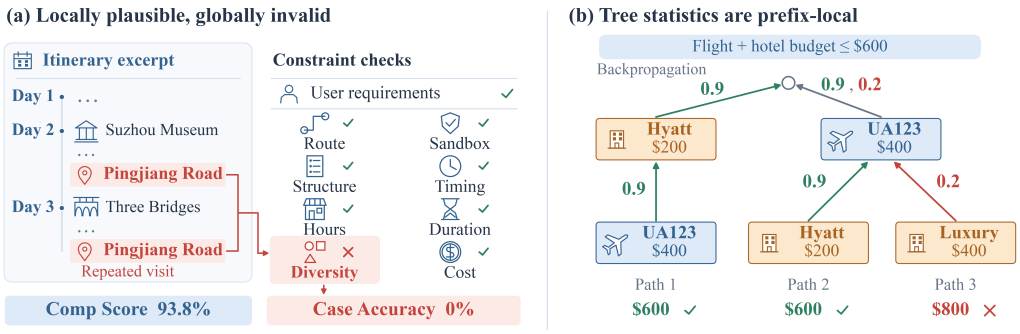

Figure 1: (a) A recorded DeepPlanning Travel failure despite 93.8% Comp Score. (b) An independent schematic under a $600 budget: Paths 1 and 2 construct the same flight–hotel combination in different orders; Path 3 changes the hotel. Returns 0.9 and 0.2 are illustrative feedback, not measurements or values derived from the budget check.

bines lookahead with repeated replanning (Wang et al., 2026). These results motivate further study of MCTS-based test-time scaling for long-horizon agents.

However, each evaluated trajectory requires model computation and environment interaction, making search costly (Yu et al., 2025). Effective scaling therefore depends on reusing its feedback. Standard MCTS stores values for history–action pairs, without an explicit entry for decision groups recurring across prefixes. In Figure 1(b), two paths construct the same flight–hotel combination in different orders, yet update separate tree paths. Another path changes the hotel and produces a different budget outcome. This motivates sharing observed outcomes at the group level, particularly when constraints couple multiple decisions.

Prior work reuses evidence through shared action values (Gelly & Silver, 2011), reasoning graphs (Gong et al., 2026), trajectory memory (Wang et al., 2026), and textual experience (Li & Tao, 2026; Xie et al., 2026). However, these methods do not maintain numerical outcome statistics indexed by recurring, unordered decision groups and aggregate evidence across overlapping groups to guide MCTS selection within the current task.

To fill this gap, we propose **HyperMCTS**, a training-free method augmenting ordered MCTS with a cross-trajectory hypergraph. Nodes identify canonical decisions across prefixes; hyperedges accumulate counts and mean returns for decision groups. **HyperUCT** aggregates incident hyperedge statistics into an action prior that complements tree values and exploration. The tree retains execution history while the hypergraph shares group-level evidence. This sharing is approximate because canonical identities may omit context affecting decision outcomes. We target settings where alternative histories can be replayed or simulated.

We evaluate HyperMCTS on DeepPlanning's Travel and Shopping tasks (Zhang et al., 2026). Under the evaluated configurations, it improves average planning accuracy over each backbone model's strongest baseline by 7.3, 2.7, and 2.3 percentage points for Qwen3.6-27B, Qwen3.5-27B, and Sonnet 4.6, respectively. HyperMCTS enables Qwen3.6-27B to outperform Claude Opus 4.6 (max) on Shopping Planning, reaching 60.8% Case Accuracy compared with the publicly reported 56.2%. In the search-efficiency study, HyperMCTS also achieves higher accuracy with fewer LLM calls and output tokens than the evaluated MCTS-based baselines. Additional experiments on SealQA (Pham et al., 2026) show improvements in question answering, extending the evaluation beyond planning.

Our contributions are threefold:

- A training-free, hypergraph-augmented MCTS framework sharing decision-group outcomes across trajectories while preserving ordered execution histories.
- HyperUCT selection combining cross-trajectory priors with tree values and exploration.
- Evaluation of accuracy and search cost on DeepPlanning, with hypergraph ablations, weight sensitivity, and generalization to question answering on SealQA.

2 PROBLEM STATEMENT

2.1 PRELIMINARIES

We consider an LLM agent interacting with an environment through tools, or application programming interfaces (APIs). Following prior work (Hao et al., 2023; Zhou et al., 2024a; Yu et al., 2025), we formulate this interaction as a finite-horizon partially observable sequential decision process. Let $\mathcal{Z}$, $\mathcal{A}$, and $\mathcal{O}$ denote the latent state, action, and observation spaces. Action $a_t \in \mathcal{A}$ changes state $z_t \in \mathcal{Z}$ according to an unknown kernel $P(z_{t+1} \mid z_t, a_t)$ and yields observation $o_{t+1} \in \mathcal{O}$. The agent acts from its observable history state

$$h_t = (x, o_0, a_0, o_1, \ldots, a_{t-1}, o_t), \tag{1}$$

where x is the user query and o_0 the initial observation. A frozen LLM with parameters θ defines a policy $\pi_\theta(a_t \mid h_t)$ over valid actions. An episode ends with a final answer, a task-defined failure, or the horizon H, producing a trajectory[1]

$$\tau = (x, o_0, a_0, o_1, \ldots, a_{T-1}, o_T), \qquad T \le H. \tag{2}$$

In planning, information-gathering actions $\mathcal{A}_{\text{info}}$ acquire evidence, whereas commit actions $\mathcal{A}_{\text{com}}$ select an item, document, or plan component. The tree branches at these decision points, retaining information-gathering steps in the history. We denote the valid search decisions at h by $\mathcal{A}_{\text{dec}}(h)$; their task-specific granularity is described in Appendix A.3. They label tree edges and map to canonical hypergraph nodes. A commit fixes part of a candidate solution, without necessarily executing a real-world transaction.

A task specifies m objectives $\mathcal{G} = \{g_1, \ldots, g_m\}$, including local requirements and global constraints. A terminal verifier V provides the trajectory reward

$$R(\tau) = V(\tau; \mathcal{G}) \in [0, 1]. \tag{3}$$

Here R is search feedback (Appendix B.2) and need not equal a final benchmark metric. When available, objective-level feedback $\rho_g(\tau) \in [0, 1]$ supplements these statistics without replacing R. Global constraints, such as a total budget, depend jointly on several decisions, so satisfying most individual requirements need not yield a correct final plan. A list of commonly used notation is provided in Table 2 in Appendix A.1.

2.2 PROBLEM DEFINITION

Let $\mathfrak{T}(x)$ be the set of trajectories that are feasible under the environment dynamics, tool schemas, and horizon H. The ideal planning objective is

$$\tau^\star \in \arg\max_{\tau \in \mathfrak{T}(x)} R(\tau). \tag{4}$$

At test time, however, the planner can evaluate only a finite set of trajectories. Given a budget B of search iterations, let $\mathcal{D}_B \subset \mathfrak{T}(x)$ denote the trajectories generated and evaluated using the frozen policy π_θ. The operational objective is therefore to return

$$\widehat{\tau}_B \in \arg\max_{\tau \in \mathcal{D}_B} R(\tau), \qquad |\mathcal{D}_B| \le B, \tag{5}$$

without updating θ. The problem is to allocate this budget so that evidence obtained from one rollout can inform other rollouts, including those that do not share the same action prefix.

3 METHOD: HYPERMCTS

3.1 OVERVIEW

HyperMCTS augments an ordered history tree $\mathcal{T}_{\text{tree}}$ with a cross-trajectory hypergraph $\mathcal{H}_k = (\mathcal{V}_k, \mathcal{E}_k)$ after k evaluated rollouts of the current task. Figure 2 illustrates the method with a shopping example involving a monitor, keyboard, and webcam under a $600 budget.

[1] We use trajectory and rollout interchangeably throughout the paper. A tree path refers to the portion of a trajectory explicitly represented in the search tree.

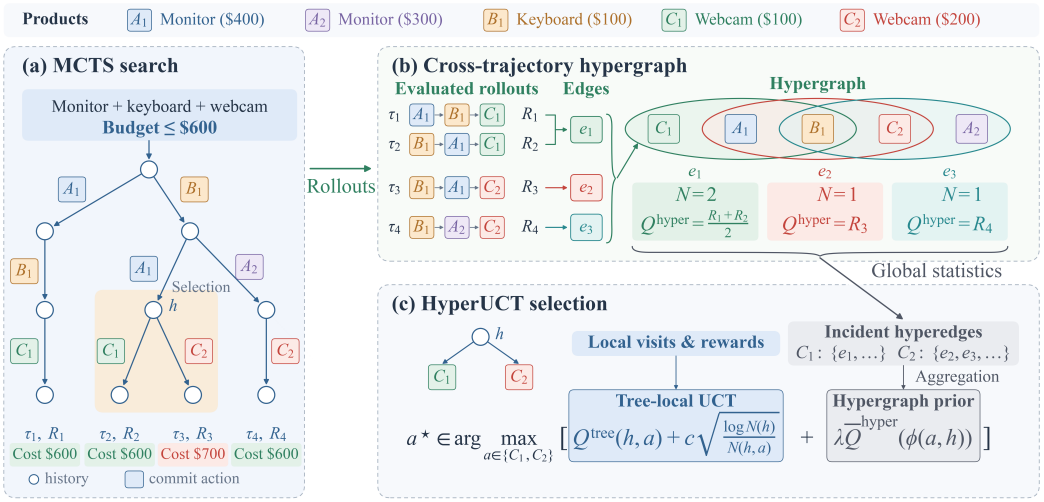

Figure 2: HyperMCTS illustrated with shopping decisions. (a) Four rollouts through a tree of commit actions. (b) Shared and overlapping action groups accumulate counts and mean feedback across trajectories. (c) At the highlighted history h, HyperUCT combines tree-local UCT (blue) with a prior obtained by aggregating incident-hyperedge statistics (gray). R_i denotes symbolic search-time feedback; costs indicate budget feasibility only. The panels show successive operations. Only trajectory-level hyperedges are drawn; search-iteration subscripts are omitted.

In panel (a), the same products can be selected along different tree paths, as in $A_1B_1C_1$ and $B_1A_1C_1$. Panel (b) maps these products to shared decision nodes and pools the two rollouts' feedback in hyperedge e_1, while overlapping product groups form other hyperedges. In panel (c), after choosing B_1 then A_1, the agent compares webcams C_1 and C_2. HyperUCT combines tree-local statistics with priors aggregated from each candidate's incident hyperedges, allowing feedback from other branches to guide this choice. Appendix A.4 details the example.

3.2 Cross-Trajectory Hypergraph

A hypergraph allows one edge to connect multiple decision nodes (Gallo et al., 1993). For a valid decision $a \in \mathcal{A}_{\text{dec}}(h)$, a domain-specific map assigns a canonical identity $v = \phi(a, h) \in \mathcal{U}$, where $\mathcal{U}$ is the space of decision identities. The map retains task-relevant distinctions while removing tree location. Let $\mathcal{C}(\tau) \subseteq \mathcal{U}$ collect these identities for the recorded search decisions in trajectory τ, which may include all search decisions or a task-defined subset. The node set is $\mathcal{V}_k = \bigcup_{i=1}^{k} \mathcal{C}(\tau_i)$.

With label space $\mathcal{Y} = \mathcal{G} \cup \{\text{global}\}$, each hyperedge has the key

$$e = (S_e, y_e), \qquad \emptyset \neq S_e \subseteq \mathcal{V}_k, \quad y_e \in \mathcal{Y}. \tag{6}$$

Each hyperedge is indexed by its decision set S_e and label y_e, while observed scores update its statistics. Trajectory-level hyperedges use $y_e = \text{global}$; objective-level hyperedges use $y_e = g \in \mathcal{G}$ for decisions associated with objective g.

After a rollout, the extractor Ψ returns observations

$$\Psi(\tau, \mathcal{G}) = \big((S_j, y_j, r_j)\big)_{j=1}^{J_\tau}, \qquad \emptyset \neq S_j \subseteq \mathcal{C}(\tau), \quad y_j \in \mathcal{Y}, \quad r_j \in [0, 1]. \tag{7}$$

Here J_τ is the number of observations. A trajectory-level observation pairs $\mathcal{C}(\tau)$ with reward $R(\tau)$ when at least two decisions are recorded. Objective-level observations use $\rho_g(\tau)$ when available, or $R(\tau)$ otherwise. Using R records an association with the whole-trajectory outcome, rather than independent objective satisfaction. Appendix A.2 specifies the extraction rules.

Starting from $\mathcal{E}_0 = \emptyset$, the observations $((S_j, y_j, r_j))_{j=1}^{J_{\tau_k}} = \Psi(\tau_k, \mathcal{G})$ update the edge set as

$$\mathcal{E}_k = \mathcal{E}_{k-1} \cup \{(S_j, y_j) : 1 \leq j \leq J_{\tau_k}\}. \tag{8}$$

For each previously unseen key e, initialize its count $N^{\text{hyper}}(e)$ and mean $Q^{\text{hyper}}(e)$ to zero. Each observation for e with score r updates

$$\begin{aligned} N^{\text{hyper}}(e) &\leftarrow N^{\text{hyper}}(e) + 1, \\ Q^{\text{hyper}}(e) &\leftarrow Q^{\text{hyper}}(e) + \frac{r - Q^{\text{hyper}}(e)}{N^{\text{hyper}}(e)}. \end{aligned} \tag{9}$$

Subscript k denotes statistics after rollout k. Observations with the same key accumulate across prefixes even when their rewards differ. This pooling does not assume that the task is order-invariant; the tree retains the valid execution histories.

3.3 HyperUCT Selection

For a decision node v, let $\mathcal{E}_k(v) = \{e = (S_e, y_e) \in \mathcal{E}_k : v \in S_e\}$ be its incident hyperedges. Their statistics define the action-level prior

$$\overline{Q}_k^{\text{hyper}}(v) = \text{Agg}\big((y_e, Q_k^{\text{hyper}}(e))_{e \in \mathcal{E}_k(v)}\big), \tag{10}$$

where Agg combines objective-level and trajectory-level records into a value in $[0, 1]$, as specified in Appendix A.2; the prior is zero when $\mathcal{E}_k(v)$ is empty.

Let $N_k(h)$ be the visit count of history h, $N_k(h, a)$ the backup count of edge (h, a), and $Q_k^{\text{tree}}(h, a)$ its mean backed-up reward. Among expanded valid decisions $\mathcal{A}_{\text{tree}}(h) \subseteq \mathcal{A}_{\text{dec}}(h)$, HyperUCT selects

$$a^\star \in \arg\max_{a \in \mathcal{A}_{\text{tree}}(h)} \left[\underbrace{Q_k^{\text{tree}}(h, a)}_{\text{local tree exploitation}} + \underbrace{c\sqrt{\frac{\log N_k(h)}{N_k(h, a)}}}_{\text{local tree exploration}} + \underbrace{\lambda \overline{Q}_k^{\text{hyper}}(\phi(a, h))}_{\text{cross-trajectory prior}} \right], \tag{11}$$

where $c > 0$ controls exploration and $\lambda \geq 0$ weights the hypergraph prior. Unvisited expanded actions receive infinite score, as in UCT (Kocsis & Szepesvári, 2006). Setting $\lambda = 0$ recovers the UCT scoring rule.

The tree value estimates continuation reward at a specific prefix, while the prior summarizes outcomes of groups containing v across trajectories. The prior does not condition on the complete current decision set and is therefore a heuristic, not a guarantee of compatibility. It guides search without changing the verifier reward $R(\tau)$.

3.4 The HyperMCTS Algorithm

Algorithm 1 follows the standard selection, expansion, simulation, and backup cycle, with an additional hypergraph update after each evaluated rollout. Each nonterminal node is expanded at most once, adding up to b children corresponding to distinct decisions. Iteration k reads statistics from the preceding $k - 1$ rollouts. The candidate set $\mathcal{D}_k$ contains evaluated trajectories, as in Section 2.2, and the final choice uses their stored rewards.

Rollouts terminate under the conditions in Section 2.1, including horizon H. Appendix A.3 describes environment restoration, early stopping, failure handling, and domain-specific guidance. Removing the hypergraph from the full agent also requires disabling any such guidance, beyond setting $\lambda = 0$.

4 Experiments

We design our experiments around five research questions (RQs). **RQ1 (Effectiveness):** does HyperMCTS improve end-to-end case accuracy over greedy reasoning and other search-based agents (Section 4.2)? **RQ2 (Efficiency):** does it do so without inflating search cost in LLM calls and tokens (Section 4.3)? **RQ3 (Ablation):** how does removing the cross-trajectory hypergraph affect performance (Section 4.4)? **RQ4 (Sensitivity):** how does performance vary with the hypergraph weight λ (Section 4.5)? **RQ5 (Generalizability):** does HyperMCTS generalize beyond planning to question answering over noisy evidence (Section 4.6)?

Algorithm 1 HyperMCTS

Require: Initial history h_0, frozen policy π_θ, verifier V, objectives $\mathcal{G}$, budget B, branching factor b, and coefficients c, λ
Ensure: Best evaluated trajectory $\widehat{\tau}_B$
1: Initialize $\mathcal{T}_{\text{tree}}$ at h_0, $\mathcal{H}_0 \leftarrow (\emptyset, \emptyset)$, and $\mathcal{D}_0 \leftarrow \emptyset$
2: **for** $k = 1$ **to** B **do**
3: **Selection:** descend from h_0 using HyperUCT with statistics at $k-1$ until reaching an unexpanded or terminal history h
4: **Expansion:** if h is nonterminal, sample up to b distinct valid actions in $\mathcal{A}_{\text{dec}}(h)$ from π_θ, add their child histories, and set h to one selected child
5: **Simulation:** complete trajectory τ_k from h with π_θ and environment feedback, or retain it if already terminal
6: **Evaluation:** obtain $R_k = V(\tau_k; \mathcal{G})$ and any available objective-level feedback
7: **Tree update:** back-propagate R_k along the tree path from h to h_0, including any edge selected during expansion, to update node and edge counts and Q_k^{tree}
8: **Hypergraph update:** set $\mathcal{V}_k \leftarrow \mathcal{V}_{k-1} \cup \mathcal{C}(\tau_k)$, extract $\Psi(\tau_k, \mathcal{G})$, update $\mathcal{E}_k$ by Eq. 8, and apply Eq. 9 to each extracted observation
9: $\mathcal{D}_k \leftarrow \mathcal{D}_{k-1} \cup \{\tau_k\}$
10: **end for**
11: **return** $\widehat{\tau}_B \in \arg\max_{\tau \in \mathcal{D}_B} R(\tau)$

4.1 Experimental Setup

Benchmarks. DeepPlanning (Zhang et al., 2026) evaluates Travel Planning and Shopping Planning under verifiable constraints. Travel uses Commonsense Score (CS Score), Personalized Score (PS Score), and their mean, Composite Score (Comp Score). Shopping uses Match Score. Both domains report Case Accuracy, which requires all evaluation conditions to be satisfied; Average Accuracy is the mean Case Accuracy across the two domains. We additionally evaluate fact-seeking question answering over noisy evidence on SealQA (Pham et al., 2026), reporting answer accuracy and a partial-credit score (Mean) that assigns half credit to abstentions.

Baselines. We compare with ReAct (Yao et al., 2023b), unstructured TTS methods CoT (Wei et al., 2022) and Reflexion (Shinn et al., 2023), and structured TTS methods ToT (Yao et al., 2023a), RAP (Hao et al., 2023), LATS (Zhou et al., 2024a), and FLARE (Wang et al., 2026). RAP, LATS, and FLARE are MCTS variants.

Models and implementation. We use Qwen3.5-27B, Qwen3.6-27B, and Sonnet 4.6 in thinking mode, keeping the backbone model frozen and fixed within each comparison. Dataset details, baseline procedures, and search hyperparameters are provided in Appendix B.1. Reward computation is described in Appendix B.2.

4.2 Main Results (RQ1)

The DeepPlanning columns in Table 1 report overall planning results, with full breakdowns in Appendix C. Case Accuracy remains much lower than partial-credit scores. For example, Qwen3.6-27B ReAct attains 74.2% Comp Score but only 10.4% Case Accuracy on Travel. A successful plan must satisfy all constraints jointly, so one missed condition can invalidate otherwise correct decisions. All six TTS baselines improve Average Accuracy over ReAct on each model. Additional reasoning and exploration allow agents to revisit choices and detect missed constraints. Structured TTS generally yields larger gains in these comparisons. RAP achieves an Average Accuracy of 40.0% on Qwen3.6-27B versus 36.7% for Reflexion, while LATS reaches 39.2% on Sonnet 4.6 versus 34.4% for Reflexion. By explicitly retaining and evaluating alternatives, structured search can compare downstream consequences before selecting a plan.

HyperMCTS attains an Average Accuracy of 47.3%, 36.9%, and 41.5% on Qwen3.6-27B, Qwen3.5-27B, and Sonnet 4.6, respectively. The strongest baselines on these models are RAP (40.0%), Reflexion (34.2%), and LATS (39.2%), giving HyperMCTS gains of 7.3, 2.7, and 2.3 percentage points. On Travel, it leads all four metrics on both Qwen models, reaching 33.8% Case Accuracy on

Table 1: Results (%) on DeepPlanning and SealQA. Avg Acc. averages Travel and Shopping Case Accuracy. **Bold** marks the best result per metric and model, including ties.

Model	Agent	DeepPlanning							SealQA
		Travel				Shopping		Avg Acc.	Acc.
		CS Score	PS Score	Comp Score	Case Acc.	Match Score	Case Acc.		
Qwen3.6-27B	ReAct	77.2	71.3	74.2	10.4	82.0	46.7	28.6	13.5
	CoT	76.2	67.1	71.7	11.7	83.6	51.7	31.7	14.4
	Reflexion	82.0	75.4	78.7	21.7	84.8	51.7	36.7	14.4
	ToT	84.4	80.8	82.6	20.4	84.1	51.7	36.1	17.1
	RAP	80.5	78.9	79.7	22.5	86.1	57.5	40.0	14.4
	LATS	83.9	77.5	80.7	24.2	81.6	53.3	38.8	15.3
	FLARE	79.9	77.0	78.4	20.0	83.2	53.3	36.7	18.9
	HyperMCTS	**86.9**	**83.8**	**85.3**	**33.8**	**87.0**	**60.8**	**47.3**	**21.6**
Qwen3.5-27B	ReAct	68.2	68.3	68.3	7.1	82.8	49.2	28.2	12.6
	CoT	70.6	64.9	67.7	8.4	83.0	49.2	28.8	13.5
	Reflexion	76.8	81.2	79.0	17.6	82.9	50.8	34.2	13.5
	ToT	76.4	76.5	76.5	15.4	83.2	50.8	33.1	**17.1**
	RAP	75.0	78.7	76.8	15.0	82.2	50.8	32.9	13.5
	LATS	76.9	78.6	77.7	17.5	83.1	50.0	33.8	14.4
	FLARE	76.6	78.7	77.7	16.7	81.2	46.7	31.7	14.4
	HyperMCTS	**79.0**	**81.7**	**80.3**	**19.6**	**85.1**	**54.2**	**36.9**	**17.1**
Sonnet 4.6	ReAct	75.9	64.2	70.0	13.4	76.6	41.7	27.6	14.4
	CoT	75.8	64.1	70.0	14.2	76.8	41.7	28.0	14.4
	Reflexion	81.9	72.2	77.0	23.8	80.2	45.0	34.4	16.2
	ToT	82.1	70.0	76.0	21.3	78.0	43.3	32.3	22.5
	RAP	**86.4**	74.2	80.3	29.6	81.9	48.3	39.0	17.1
	LATS	84.9	78.4	81.6	29.2	83.1	49.2	39.2	19.8
	FLARE	85.2	**81.3**	**83.2**	**32.1**	80.7	44.2	38.2	15.3
	HyperMCTS	84.5	**81.3**	82.9	30.4	**84.3**	**52.5**	**41.5**	**26.1**

Qwen3.6-27B versus 24.2% for LATS. It also achieves the highest overall Travel PS Score across all three models, including a tie at 81.3% on Sonnet 4.6. On Shopping, it leads both Match Score and Case Accuracy across all three models. With Qwen3.6-27B, it reaches 60.8% Case Accuracy on Shopping, exceeding the 56.2% reported for Claude Opus 4.6 (max) on the public leaderboard.[2] These joint gains reflect improvements in both satisfying individual requirements and producing fully correct plans.

4.3 Search Efficiency (RQ2)

Figure 3 compares accuracy and per-case cost on Shopping Planning with Qwen3.6-27B in thinking mode. We evaluate RAP, LATS, FLARE, and HyperMCTS because these MCTS-based methods support the same branching-factor settings, $b \in \{2, 3, 6\}$. HyperMCTS achieves the highest Case Accuracy and uses the fewest LLM calls and output tokens at each setting. At $b = 3$, it reaches 60.8% accuracy with about 122 calls and 85K output tokens, while RAP reaches 57.5% with 144 calls and 141K tokens. LATS and FLARE use $3.8\times$ and $3.3\times$ as many output tokens as HyperMCTS, respectively. FLARE requires the most LLM calls because it runs MCTS again from the updated state after each committed action to select the next action. Increasing b from 3 to 6 yields only 0.1–1.0 percentage points of accuracy gain across methods despite substantially higher costs. HyperMCTS at $b = 3$ is more accurate and less costly than every baseline at $b = 6$.

[2] https://qwenlm.github.io/Qwen-Agent/en/benchmarks/deepplanning/

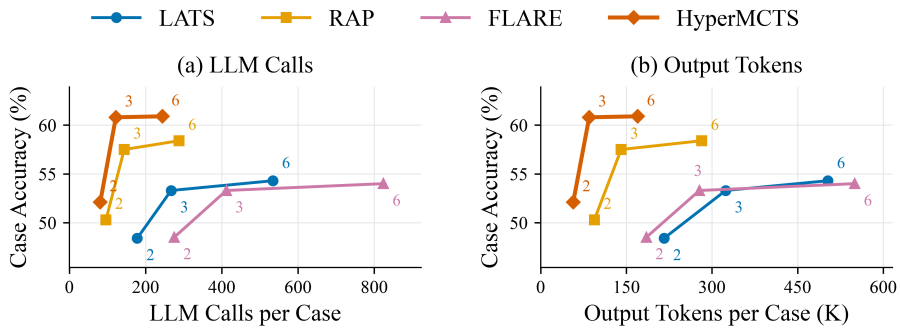

Figure 3: Accuracy–cost trade-offs of MCTS-based methods on Shopping Planning (120 cases, Qwen3.6-27B, thinking mode). Costs are rounded per-case averages. Labels denote branching factors 2, 3, and 6; lines connect configurations of the same method.

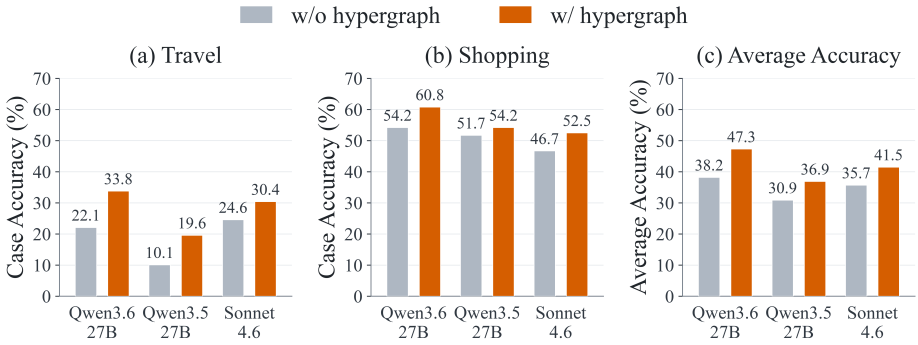

Figure 4: Hypergraph ablation on DeepPlanning, comparing MCTS (w/o hypergraph) and HyperMCTS (w/ hypergraph).

4.4 Ablation Study (RQ3)

We compare HyperMCTS with the MCTS configuration without the hypergraph, using the same backbone models, rollout budget, and branching factor. Figure 4 shows higher Case Accuracy for HyperMCTS in both domains on all three models. Average Accuracy is 47.3% versus 38.2% on Qwen3.6-27B, 36.9% versus 30.9% on Qwen3.5-27B, and 41.5% versus 35.7% on Sonnet 4.6. The two Qwen models benefit more on Travel, with gains of 11.7 and 9.5 percentage points, compared with 6.6 and 2.5 on Shopping. Sonnet gains 5.8 points in each domain. This comparison evaluates the hypergraph component as a whole, rather than isolating the additive HyperUCT term from other uses of its statistics. More detailed results by language and difficulty are provided in Appendix C.3.

4.5 Sensitivity Analysis (RQ4)

We vary the hypergraph weight $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ on Travel while keeping B, b, and c fixed. Figure 5 shows that Qwen3.5-27B attains its highest Comp Score (80.3%) and Case Accuracy (19.6%) at $\lambda = 0.3$. Qwen3.6-27B peaks at $\lambda = 0.7$, reaching 85.3% Comp Score and 33.8% Case Accuracy. Sonnet also achieves its highest Case Accuracy (30.4%) at $\lambda = 0.7$, although its Comp Score peaks at $\lambda = 0.3$. Stronger backbone models may produce more reliable search experience, allowing HyperMCTS to assign greater weight to cross-trajectory evidence in hypergraph statistics. Noisier experience from a weaker model may make a more moderate weight preferable. For the main Travel results, we use $\lambda = 0.3$ for Qwen3.5-27B and $\lambda = 0.7$ for Qwen3.6-27B and Sonnet 4.6. Full results are provided in Appendix C.4.

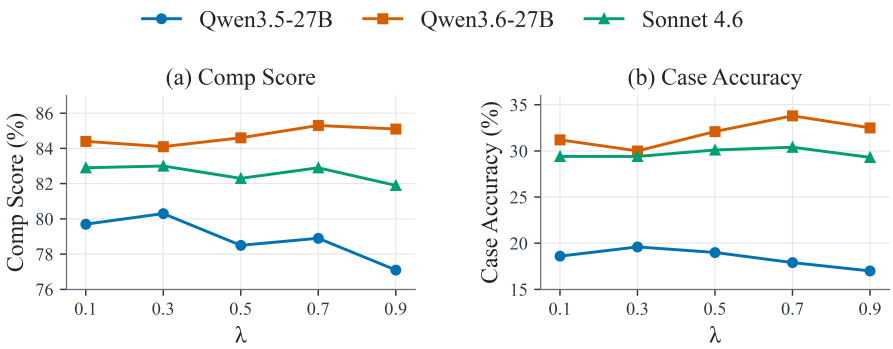

Figure 5: Sensitivity to the hypergraph weight λ on Travel Planning. Each curve shows one backbone model with $B = 5$, $b = 3$, and $c = 1.414$.

4.6 Generalizability (RQ5)

We evaluate on SealQA to test whether HyperMCTS generalizes beyond planning to question answering over noisy evidence. The SealQA column in Table 1 reports accuracy, with full results in Appendix C.5 and the MCTS formulation in Appendix A.3.

HyperMCTS achieves 21.6% and 26.1% accuracy on Qwen3.6-27B and Sonnet 4.6, exceeding the strongest baseline on each model by 2.7 and 3.6 percentage points, respectively. On Qwen3.5-27B, it ties ToT at 17.1%. These results extend the evaluation of HyperMCTS to question answering over noisy evidence.

5 Limitations

HyperMCTS assumes that the environment supports reversible actions, state restoration, or replay so that alternative action trajectories can be explored at test time. This is practical for constructing a shopping cart or revising a travel plan, where tentative choices can be changed before execution. Tasks with irreversible actions or persistent external effects require a simulator or another mechanism for evaluating alternatives without committing their effects.

HyperMCTS is intended for search trees with substantial overlap in decision groups across trajectories. Its hypergraph represents these groups as unordered sets, allowing their outcome statistics to be shared across different action prefixes. Limited overlap reduces the opportunities for such sharing. The set representation is also less suitable when outcomes depend strongly on action order or context omitted from the decision identities. Although the search tree preserves execution order, the hypergraph may pool incompatible feedback in these settings.

6 Conclusion and Future Work

HyperMCTS augments MCTS with a cross-trajectory hypergraph that organizes search outcomes by labelled decision groups. HyperUCT reads these statistics as an action-level prior alongside prefix-specific tree values and exploration. The resulting search retains ordered histories while making previously collected group evidence available across branches of the same task. On the evaluated DeepPlanning models, HyperMCTS achieves the highest Average Accuracy and improves over the no-hypergraph variant. The Shopping cost study shows higher accuracy with fewer calls and output tokens on Qwen3.6-27B, while the SealQA results extend the evaluation to question answering. Future work can combine HyperMCTS with world models to simulate action outcomes and support planning in agentic tasks with irreversible actions.

AI USE STATEMENT

In this work, generative AI tools were used to assist with manuscript writing and language editing, including improving clarity and readability and proofreading for grammatical errors. Generative AI tools were not used to formulate the research questions or hypotheses, design the methodology or experiments, generate or process data, implement the methods, derive mathematical claims, or interpret the results. All AI-assisted text was reviewed and revised by the authors. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We use the publicly available DeepPlanning (Zhang et al., 2026) and SealQA (Pham et al., 2026) datasets. We describe HyperMCTS in detail in Section 3 and Appendix A, provide core-search pseudocode in Algorithm 1, and specify the main hyperparameters in Appendix B.1 and Section 4.5. Our code will be made publicly available upon acceptance.

REFERENCES

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. In *International Conference on Learning Representations*, volume 2025, pp. 64485–64515, 2025.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Yujia Chen, Yingli Zhou, Fangyuan Zhang, and Cuiyun Gao. Llm-based test case generation in dbms through monte carlo tree search. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*, pp. 355–366, 2026.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201, 1993.

Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

Junfeng Gong, Zhiyi Wei, Junying Chen, Cheng Liu, and Huawei Li. From large to small: Transferring cuda optimization expertise via reasoning graph. In *International Conference on Learning Representations*, volume 2026, pp. 121082–121107, 2026.

Runquan Gui, Zhihai Wang, Jie Wang, Chi Ma, Huiling Zhen, Mingxuan Yuan, Jianye HAO, Defu Lian, Enhong Chen, and Feng Wu. Hypertree planning: Enhancing LLM reasoning via hierarchical thinking. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=45he3Ri6JP.

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.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Xinzhe Li and Yaguang Tao. When does memory help multi-trajectory inference for tool-use llm agents? *arXiv preprint arXiv:2605.28224*, 2026.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kh9Zt2Ldmn.

Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with process reward-guided tree search for better complex reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10256–10277, 2025.

Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. Sealqa: Raising the bar for reasoning in search-augmented language models. In *International Conference on Learning Representations*, volume 2026, pp. 98989–99017, 2026.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solver. In *International Conference on Learning Representations*, volume 2025, pp. 20788–20807, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zehong Wang, Fang Wu, Hongru Wang, Xiangru Tang, Bolian Li, Zhenfei Yin, Yijun Ma, Yiyang Li, Weixiang Sun, Xiusi Chen, et al. Why reasoning fails to plan: A planning-centric analysis of long-horizon decision making in llm agents. *arXiv preprint arXiv:2601.22311*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xin Xie, Dongyun Xue, Wuguannan Yao, Mingxiao Feng, Wengang Zhou, Xiang Qi, Houqiang Li, and Peng Zhang. Sga-mcts: Decoupling planning from execution via training-free atomic experience retrieval. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1176–1193, 2026.

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, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning. In *International Conference on Learning Representations*, volume 2025, pp. 65157–65184, 2025.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*, 2024.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7377–7407, 2026.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=njwv9BsGHF.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024b.

Table 2: Core notation for the search tree and cross-trajectory hypergraph.

Symbol	Meaning
x, h_t, τ	Query, observable interaction history, and complete or horizon-terminated trajectory
H, B, b	Interaction horizon, search-iteration budget, and expansion branching factor
π_θ, $\mathcal{A}_{\mathrm{dec}}(h)$	Frozen proposal policy and task-defined search decisions
V, $R(\tau)$	Terminal search verifier and trajectory reward
$\mathcal{G}$, $\mathcal{Y}$	Query objectives and hyperedge labels
$\mathcal{U}$, ϕ, $\mathcal{C}(\tau)$	Canonical decision space, canonicalization map, and recorded decision set
$e = (S_e, y_e)$, Ψ	Labelled decision-group record and observation extractor
$N_k(h)$	Visit count of a history node
$N_k(h,a)$, $Q_k^{\mathrm{tree}}(h,a)$	Tree-edge backup count and mean search feedback
$N_k^{\mathrm{hyper}}(e)$, $Q_k^{\mathrm{hyper}}(e)$	Hyperedge observation count and mean feedback
$\overline{Q}_k^{\mathrm{hyper}}(v)$	Action-level prior from incident hyperedges
c, λ	UCT exploration coefficient and hypergraph-prior weight

A Additional Method Details

A.1 Notation Summary

Table 2 summarizes the notions used in the search tree and the cross-trajectory hypergraph.

A.2 Hypergraph Specification

This appendix separates the hypergraph's statistical interface from domain-specific search adapters. The general algorithm does not require the same low-level action vocabulary, recorded trace, or candidate-generation procedure in every task.

Decision identities. Canonicalization removes tree location while retaining task-relevant distinctions, such as product identifiers, tool arguments, dates, and semantic roles. Multiplicity and temporal roles must be encoded in the identity when relevant, since a set does not retain repeated occurrences. The recorded set $\mathcal{C}(\tau)$ may include only the decisions materialized in the tree; it need not include every tool call in the simulated suffix.

Observations and labels. The trajectory-level record associates the recorded decision set $\mathcal{C}(\tau)$ with $R(\tau)$ when the set contains at least two distinct identities. The objective-level instantiation associates each relevant decision v with an objective g through a singleton decision set, giving $e = (\{v\}, g)$. These records also receive the terminal search score $R(\tau)$, rather than an independently measured objective reward. Although engineering implementations may call these records "compatibility" and "satisfaction" edges, those names do not establish compatibility in the current prefix or separate objective satisfaction. Multiple decision nodes are connected by the trajectory-level record.

Aggregation. Let $\mathcal{E}_{\mathrm{obj}}(v)$ and $\mathcal{E}_{\mathrm{traj}}(v)$ be the objective-level and trajectory-level records incident to v. Define $q_{\mathrm{obj}}(v)$ as the maximum empirical value over the former and $q_{\mathrm{traj}}(v)$ as the unweighted mean over the latter, when the corresponding set is nonempty. The aggregation is

$$\overline{Q}^{\mathrm{hyper}}(v) = \begin{cases} 0.7q_{\mathrm{obj}}(v) + 0.3q_{\mathrm{traj}}(v), & \text{both types present,} \\ q_{\mathrm{obj}}(v), & \text{only objective records present,} \\ q_{\mathrm{traj}}(v), & \text{only trajectory records present,} \\ 0, & \text{neither type present.} \end{cases} \tag{12}$$

Each incident edge contributes separately, even when another edge has the same label and value. This is a bounded heuristic. The maximum retains the strongest observed objective association; the trajectory term averages group means rather than weighting groups by their observation counts. Neither operation estimates the causal effect of selecting v.

Exact keys and approximate merging. The main formulation uses an exact key (S_e, y_e). The implementation notes additionally describe a Jaccard-based union merge with threshold 0.8. Jaccard similarity between two sets S and S' is $|S \cap S'|/|S \cup S'|$. Applying this operation to eligible records replaces their stored sets by $S \cup S'$. Such a record represents a cluster of overlapping observations; the union need not have appeared in a single rollout and its value is not the empirical return of that exact union. Its count and accumulated return must represent the contributing observations. A concrete merge policy must specify record-type and label eligibility, the treatment of objective metadata, and count migration. Online merge order and tie handling can affect the resulting clusters, so permutation invariance of an exact action-set key does not imply insertion-order invariance of approximate merging.

A.3 Domain Interfaces and Search Integration

Search-loop details. Alternative histories require environment replay, restorable snapshots, or a simulator. Terminal histories bypass expansion and simulation. If expansion yields no valid child, the adapter applies its task-specific failure rule. Candidate generation, child selection, ties, rollout depth, and stopping conditions are defined by the domain interface. When search stops before B iterations, $\mathcal{D}_B$ denotes the trajectories evaluated before stopping; if none completes, the adapter returns its task-defined fallback. The final choice uses stored rewards without re-evaluation. Tree backup updates the represented path, while simulated suffix decisions may contribute hypergraph observations without becoming tree children.

Planning decisions. Shopping identifies product-addition decisions by product identifiers and associates them with parsed query requirements, such as a brand or a named item. Travel uses staged choices, including outbound transport, accommodation, daily plans, and return transport. Its recorded decision set can consist of the materialized tree path, with feedback obtained after completing the rest of the plan. This distinction is why Section 3.2 defines the recorded set separately from the full trajectory. Stage and temporal information are relevant to deciding which choices may share an identity.

SealQA as MCTS. In the ReAct formulation of SealQA, a search state consists of the question and its accumulated action–observation history. The actions are `Search[query]`, which retrieves a passage from the question's document pool; `Lookup[keyword]`, which returns a matching sentence from the latest retrieved document; and `Finish[answer]`, which terminates the trajectory. Executing an action appends its observation to the history; alternative branches replay their action prefixes. Expansion proposes candidate continuations, and a bounded rollout completes the selected branch. Search feedback updates the tree path and hypergraph records over canonical action identities, with the question as the objective label. Action identities use the action type and arguments. An argument-only lookup identity can conflate decisions made under different retrieval contexts.

Search feedback and reporting. Appendix B.2 specifies the domain rewards and evaluation prompts. Evaluator and guidance calls contribute to computational cost even when they do not constitute another completed search iteration.

A.4 Illustrative Search Example

The shopping example in Figure 2 explains the update and readout operations. It is a constructed example, not an experimental case study. Panel (a) summarizes the decision sequences rather than specifying which tree nodes are materialized during simulation. The task is to select a monitor, keyboard and webcam within \$600. Monitors A_1 and A_2 cost \$400 and \$300; keyboard B_1 costs \$100; webcams C_1 and C_2 cost \$100 and \$200. The four rollouts $A_1B_1C_1$, $B_1A_1C_1$, $B_1A_1C_2$ and $B_1A_2C_2$ cost \$600, \$600, \$700 and \$600, respectively. These totals indicate budget feasibility only; R_1–R_4 denote symbolic search scores.

Under exact-key aggregation, the first two rollouts contribute two observations to $e_1 = (\{A_1, B_1, C_1\}, \text{global})$, with $N^{\text{hyper}}(e_1) = 2$ and $Q^{\text{hyper}}(e_1) = (R_1 + R_2)/2$. The third and fourth rollouts update $e_2 = (\{A_1, B_1, C_2\}, \text{global})$ and $e_3 = (\{A_2, B_1, C_2\}, \text{global})$, each with count one and mean feedback R_3 and R_4, respectively. In the figure's hyperedge records, N abbreviates N^{hyper}.

At a later history h after choosing B_1 then A_1, both C_1 and C_2 may be valid candidates. The prior for C_1 can read e_1, including feedback collected under prefix A_1B_1; the prior for C_2 can read both e_2 and e_3. Thus, the prior aggregates statistics from hyperedges incident to the candidate rather than filtering them by the current prefix. HyperUCT also uses their separate tree values and visit counts, so the hyperedge means alone do not determine which candidate is selected. This demonstrates a possible transfer of outcome evidence across prefixes. It does not establish how often that transfer occurs in the experiments, whether it changes a decision, or whether it prevents a repeated evaluation. The action identities, allowed construction orders, and illustrative outcomes must be checked against a concrete interface before using this example as an implementation trace.

B Implementation Details

B.1 Experimental and Implementation Details

Datasets and evaluation. DeepPlanning Travel contains 120 tasks with Chinese and English versions, giving 240 evaluated instances. Shopping contains 120 multi-product cart tasks across three difficulty levels, with 50, 50, and 20 cases at Levels 1, 2, and 3. Final outputs are scored by rule-based checkers; Travel additionally requires parsing the generated plan into a structured representation. Appendix B.2 specifies the feedback used during search and its relation to these reporting metrics. We use the 111-question Seal-0 subset of SealQA; the accuracy and partial-credit formulas are given in Appendix C.5. Planning scores and accuracies are reported as percentages, while SealQA Mean is on a 0–1 scale. Table captions specify the boldface convention.

Baseline procedures. ReAct interleaves reasoning with tool actions and observations. CoT uses the same interaction procedure with an additional zero-shot instruction to reason step by step (Kojima et al., 2022), while Reflexion uses verbal self-critique to revise subsequent attempts. These unstructured TTS methods do not maintain an explicit search tree. Structured TTS methods organize alternative reasoning or action trajectories, including ToT and the MCTS variants RAP, LATS, and FLARE. HyperMCTS belongs to the latter family and augments MCTS with a cross-trajectory hypergraph. All methods use the same frozen backbone model within each comparison. The open-weight Qwen3.5-27B and Qwen3.6-27B models and Sonnet 4.6 operate in thinking mode.

Search hyperparameters. Unless otherwise stated, HyperMCTS uses at most $B = 5$ search iterations and branching factor $b = 3$ on DeepPlanning. We use a UCT exploration coefficient $c = 1.414$ as in Park et al. (2025) and Chen et al. (2026). The efficiency study varies $b \in \{2, 3, 6\}$, as described in Section 4.3.

Reported weights. The Shopping configuration uses $\lambda = 0.5$. Travel uses $\lambda = 0.3$ for Qwen3.5-27B and $\lambda = 0.7$ for Qwen3.6-27B and Sonnet 4.6 (Section 4.5). SealQA uses $\lambda = 0.2$ for Qwen3.6-27B and Sonnet 4.6 and $\lambda = 0.5$ for Qwen3.5-27B and Opus 4.8, as detailed in Appendix C.5.

B.2 Reward Computation and Evaluation Prompts

We describe reward computation for DeepPlanning and SealQA. The hypergraph prior guides selection but is not added to the reward. Terminal rewards update both tree and hypergraph statistics.

Travel Planning. We convert the final itinerary to JSON using DeepPlanning's language-specific conversion prompts (Zhang et al., 2026), with the parser configured under the `claude-haiku` alias. The parser formats the plan; programmatic checkers evaluate it against task constraints and the local travel database. For parsed plan p_τ,

$$R_{\text{travel}}(\tau) = \tfrac{1}{2}\big[\text{CS}(p_\tau) + \text{PS}(p_\tau)\big], \qquad \text{CS}(p_\tau) = \tfrac{1}{8}\sum_{d=1}^{8} c_d(p_\tau). \tag{13}$$

Each commonsense dimension c_d is one only if its nonempty set of evaluated checks all pass, and zero otherwise. PS applies the same all-pass rule to the personalized constraints. Undefined checks are excluded. Plans shorter than 50 characters, failed conversion, a missing task database, or checker errors receive zero. If the database argument or sample identifier is absent, a separate fallback returns 0.3 when the parsed object contains `daily_plans`, and 0.1 otherwise.

Shopping Planning. The judge uses the agent's model configuration and receives the request, cached product attributes, and applied coupons. Serialized product details are truncated to 3,000 characters; the full interaction transcript is not provided. For a valid JSON response,

$$R_{\text{shopping}}(\tau) = \text{clip}_{[0,1]}(s_\tau), \tag{14}$$

where s_τ is the returned `score`, used directly rather than recomputed from the returned counts. Empty carts receive zero. Failed JSON extraction triggers a textual-fraction fallback, returning the ratio (zero for a zero denominator) without clipping; this fallback does not enforce the normalized-reward assumption in Section 2.1. Unparseable responses or judge-call errors return 0.5. Final benchmark scores instead use reference-product and coupon matching.

SealQA. A trajectory-value evaluator uses the actor's model configuration to score the question and its sequence of thoughts, actions, and observations. Its LATS-style prompt (Zhou et al., 2024a) requests an integer $s_\tau \in \{1, \ldots, 10\}$, giving

$$R_{\text{SealQA}}(\tau) = s_\tau/10. \tag{15}$$

The evaluator also scores intermediate candidates; the terminal `value_score` is used for backup and hypergraph updates. Empty trajectories or failed calls receive zero. If numeric parsing fails, responses containing "incorrect" or "wrong" receive 0.2; otherwise, "correct" or "reasonable" yields 0.7, and other responses yield zero. A separate reference-answer grader maps `CORRECT`, `NOT_ATTEMPTED`, and `INCORRECT` to 1, 0.5, and 0 for reporting. For SealQA, the grader's score and label remain in the terminal observation supplied to the value evaluator, including after forced `Finish`. Thus, terminal value evaluation receives reference-derived feedback even though the value prompt has no reference-answer field.

All three configurations retain the candidate with the highest observed terminal reward. Travel and Shopping permit early stopping at 0.95; SealQA uses its configured `value_stop`. The complete search-evaluation prompts follow, with braces denoting runtime substitutions. Travel reuses DeepPlanning's conversion prompts and has no additional LLM scoring prompt.

Shopping terminal-judge prompt

You are a strict evaluator checking whether a shopping cart perfectly satisfies a user's requirements.

Evaluation Rules
- Each requirement must be satisfied by EXACTLY ONE product in the cart
- ALL constraints in a requirement must be met (brand, name keywords, color, size, rating thresholds, sales volume, stock, delivery time, etc.)
- If a requirement says "more than 200 four-star reviews", check the actual number — 199 does NOT satisfy it
- If a requirement says "'Omni-Wick' in the name", the product name MUST contain that exact phrase
- If a requirement specifies a brand, ONLY that brand satisfies it — no substitutes
- A product that meets MOST but not ALL constraints of a requirement counts as NOT satisfied
- Be strict: when in doubt, mark as NOT satisfied

Few-Shot Examples

Example 1:
Requirement: "a pink 'Active Performance Tank Top' from Puma with average score > 4.5"
Product in cart: name="Puma Active Performance Tank Top", brand="Puma", color="Pink", avg_score=4.7
→ SATISFIED (brand ✓, name contains phrase ✓, color ✓, score 4.7>4.5 ✓)

Example 2:
Requirement: "a pink 'Active Performance Tank Top' from Puma with average score > 4.5"
Product in cart: name="Puma Classic Running Vest", brand="Puma", color="Pink", avg_score=4.8
→ NOT SATISFIED (name does NOT contain 'Active Performance Tank Top' ✗)

Example 3:
Requirement: "item from Salomon with stock quantity over 250 and more than 400 five-star ratings"
Product in cart: name="Salomon Trail Shoes", brand="Salomon", stock=300, five_star=380
→ NOT SATISFIED (five_star 380 < 400 ✗, even though stock 300>250 ✓)

Example 4:
Requirement: "monthly sales volume more than 550, transport time less than 2 days"
Product in cart: monthly_sales=600, transport_time=3 days
→ NOT SATISFIED (transport_time 3 > 2 ✗)

User's Shopping Request
{self.query}

Current Cart Contents
{json.dumps(cart_details, indent=2, ensure_ascii=False)[:3000]}

Coupons Applied
{json.dumps(used_coupons, ensure_ascii=False) if used_coupons else 'None'}

Task
For each requirement, check if there is a product in the cart that satisfies ALL its constraints.
Be strict — partial matches do not count.

Output
Return ONLY a JSON object:
{"satisfied_count": <int>, "total_requirements": <int>, "score": <float 0-1>, "reasoning": "<brief explanation per requirement>"}

SealQA trajectory-value prompt

Analyze the trajectory of a solution to a fact-seeking question answering task. The trajectory consists of:
- Thought N: reasoning about the current situation
- Action N: one of Search[entity], Lookup[keyword], or Finish[answer]
- Observation N: the result of executing the action

Given the question and the trajectory, evaluate its correctness. Pay particular attention to whether:
- The retrieved evidence actually supports the final answer
- The final answer is consistent with the observations and not a hallucination
- The trajectory navigated noisy/conflicting search results sensibly
- For false-premise questions, whether the trajectory recognizes the false premise

Provide brief reasoning, then on the LAST line conclude exactly:
"Thus the correctness score is s"
where s is an integer from 1 to 10 (10 = clearly correct, 1 = clearly wrong).

Question: {question}

Trajectory:
{trajectory}

Reasoning:

Table 3: Travel Planning, per-language breakdown (%) with all models in thinking mode. CS Score (Commonsense Score), PS Score (Personalized Score), Comp Score (Composite Score), and Case Acc. (Case Accuracy) follow the definitions in the main text. For each model and language group (including Overall), the best value of every metric is in **bold**, including ties.

Model	Agent	ZH				EN				Overall			
		CS Score	PS Score	Comp Score	Case Acc.	CS Score	PS Score	Comp Score	Case Acc.	CS Score	PS Score	Comp Score	Case Acc.
Qwen3.6-27B	ReAct	74.7	76.7	75.7	7.5	79.6	65.8	72.7	13.3	77.2	71.3	74.2	10.4
	CoT	73.3	72.5	72.9	6.7	79.1	61.7	70.4	16.7	76.2	67.1	71.7	11.7
	Reflexion	79.5	85.8	82.7	15.8	84.5	65.0	74.8	27.5	82.0	75.4	78.7	21.7
	ToT	82.4	86.6	84.5	15.8	86.4	**75.0**	80.7	25.0	84.4	80.8	82.6	20.4
	RAP	79.7	82.8	81.3	22.5	81.2	**75.0**	78.1	22.5	80.5	78.9	79.7	22.5
	LATS	81.7	85.8	83.8	22.5	86.0	69.2	77.6	25.8	83.9	77.5	80.7	24.2
	FLARE	78.9	83.0	81.0	18.3	80.8	71.0	75.9	21.7	79.9	77.0	78.4	20.0
	HyperMCTS	**85.5**	**93.3**	**89.4**	**29.2**	**88.3**	74.2	**81.3**	**38.3**	**86.9**	**83.8**	**85.3**	**33.8**
Qwen3.5-27B	ReAct	67.6	68.3	68.0	6.7	68.8	68.3	68.6	7.5	68.2	68.3	68.3	7.1
	CoT	72.0	68.9	70.5	8.4	69.2	60.8	65.0	8.3	70.6	64.9	67.7	8.4
	Reflexion	76.7	84.9	80.8	17.7	76.8	**77.5**	77.2	17.5	76.8	81.2	79.0	17.6
	ToT	75.6	78.8	77.2	13.3	77.2	74.2	75.7	17.5	76.4	76.5	76.5	15.4
	RAP	74.9	81.5	78.2	14.2	75.1	75.8	75.5	15.8	75.0	78.7	76.8	15.0
	LATS	75.4	81.4	78.4	16.7	78.3	75.8	77.1	18.3	76.9	78.6	77.7	17.5
	FLARE	76.2	81.3	78.8	16.7	77.0	76.1	76.6	16.7	76.6	78.7	77.7	16.7
	HyperMCTS	**79.0**	**86.4**	**82.7**	**20.3**	**79.0**	76.9	**78.0**	**18.8**	**79.0**	**81.7**	**80.3**	**19.6**
Sonnet 4.6	ReAct	65.7	55.1	60.4	1.7	86.0	73.3	79.7	25.0	75.9	64.2	70.0	13.4
	CoT	66.1	55.0	60.6	2.5	85.5	73.2	79.4	25.8	75.8	64.1	70.0	14.2
	Reflexion	74.5	69.9	72.2	**7.5**	89.3	74.4	81.9	40.0	81.9	72.2	77.0	23.8
	ToT	73.1	67.5	70.3	4.2	91.0	72.5	81.8	38.3	82.1	70.0	76.0	21.3
	RAP	**77.1**	71.7	74.4	5.8	**95.6**	76.7	86.2	53.3	**86.4**	74.2	80.3	29.6
	LATS	75.6	**76.7**	**76.2**	**7.5**	94.1	80.0	87.1	50.8	84.9	78.4	81.6	29.2
	FLARE	74.9	**76.7**	75.8	**7.5**	95.5	85.8	90.7	**56.7**	85.2	**81.3**	**83.2**	**32.1**
	HyperMCTS	73.7	74.2	74.0	**7.5**	95.2	**88.3**	**91.8**	53.3	84.5	**81.3**	82.9	30.4

C Additional Experimental Results

This appendix provides the complete per-language Travel Planning and per-difficulty Shopping Planning breakdowns underlying the DeepPlanning results in Table 1, including Qwen3.6-27B, Qwen3.5-27B, and Sonnet 4.6. It also reports detailed ablation and sensitivity results, along with the full SealQA metrics and additional results for Opus 4.8.

C.1 Travel Planning

Table 3 reports the detailed Travel Planning results for all three models. On Sonnet 4.6, HyperMCTS achieves the highest EN PS Score (88.3%) and Comp Score (91.8%); its overall PS Score is 81.3% (tied best), and its overall Comp Score is 82.9%.

Sonnet 4.6 performs substantially better on EN than ZH in Travel, with HyperMCTS attaining 53.3% versus 7.5% Case Accuracy, as shown in Table 3. This contrast with Qwen's stronger ZH results may partly reflect differences in Chinese–English capabilities acquired during pretraining, which can influence how reliably each model interprets and satisfies planning constraints.

Table 4: Shopping Planning, per-difficulty breakdown (%) with all models in thinking mode. Case Acc. denotes Case Accuracy. For each model and difficulty group (including Overall), the best value of every metric is in **bold**, including ties.

Level	Agent	Qwen3.6-27B		Qwen3.5-27B		Sonnet 4.6	
		Match Score	Case Acc.	Match Score	Case Acc.	Match Score	Case Acc.
L1 (n=50)	ReAct	84.7	46.0	83.3	46.0	75.3	36.0
	CoT	84.6	50.0	**86.1**	**52.0**	75.8	38.0
	Reflexion	86.5	54.0	84.2	**52.0**	84.2	46.0
	ToT	86.1	50.0	84.2	48.0	76.0	38.0
	RAP	88.4	58.0	82.8	46.0	81.4	44.0
	LATS	84.2	54.0	**86.1**	**52.0**	84.7	46.0
	FLARE	87.0	58.0	83.2	46.0	80.1	42.0
	HyperMCTS	**88.8**	**62.0**	85.1	50.0	**85.6**	**50.0**
L2 (n=50)	ReAct	80.0	48.0	80.9	52.0	77.0	48.0
	CoT	**83.9**	**56.0**	78.3	42.0	77.0	46.0
	Reflexion	82.2	46.0	81.7	50.0	75.2	42.0
	ToT	82.6	54.0	81.3	52.0	79.6	48.0
	RAP	83.5	54.0	77.8	46.0	80.9	54.0
	LATS	79.1	52.0	79.6	48.0	81.3	52.0
	FLARE	80.0	46.0	78.0	46.0	79.9	48.0
	HyperMCTS	**83.9**	**56.0**	**84.3**	**54.0**	**85.2**	**58.0**
L3 (n=20)	ReAct	80.4	45.0	86.1	50.0	78.6	40.0
	CoT	80.2	45.0	87.1	60.0	78.6	40.0
	Reflexion	87.1	60.0	82.9	50.0	82.9	**50.0**
	ToT	82.8	50.0	85.7	55.0	79.1	45.0
	RAP	87.1	65.0	**91.4**	**75.0**	**85.7**	45.0
	LATS	81.2	55.0	84.2	50.0	83.5	**50.0**
	FLARE	81.4	60.0	84.3	50.0	84.0	40.0
	HyperMCTS	**90.0**	**70.0**	87.1	65.0	78.6	45.0
Overall (n=120)	ReAct	82.0	46.7	82.8	49.2	76.6	41.7
	CoT	83.6	51.7	83.0	49.2	76.8	41.7
	Reflexion	84.8	51.7	82.9	50.8	80.2	45.0
	ToT	84.1	51.7	83.2	50.8	78.0	43.3
	RAP	86.1	57.5	82.2	50.8	81.9	48.3
	LATS	81.6	53.3	83.1	50.0	83.1	49.2
	FLARE	83.2	53.3	81.2	46.7	80.7	44.2
	HyperMCTS	**87.0**	**60.8**	**85.1**	**54.2**	**84.3**	**52.5**

C.2 Shopping Planning

Table 4 reports the detailed Shopping Planning results for all three models.

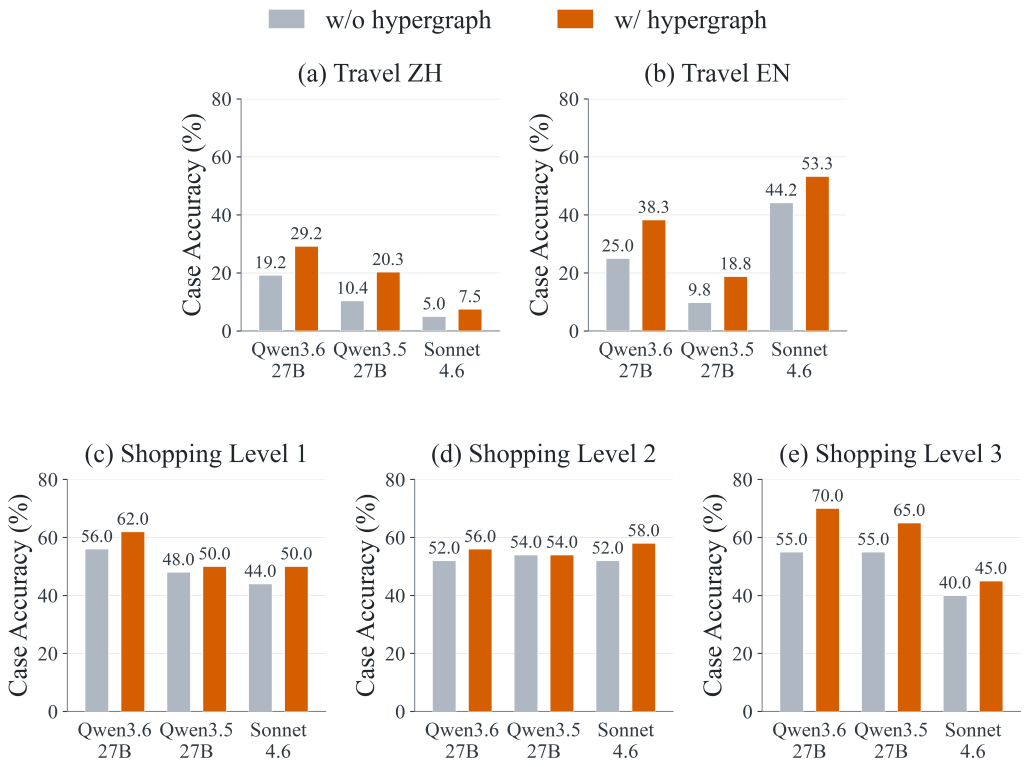

Figure 6: Case Accuracy (%) by Travel language and Shopping difficulty for MCTS (w/o hypergraph) and HyperMCTS (w/ hypergraph).

C.3 Hypergraph Ablation Breakdowns

Figure 6 reports Case Accuracy by language and difficulty for the ablation in Section 4.4. HyperMCTS improves over MCTS on both ZH and EN for all three models. On Shopping, the largest gains for the two Qwen models occur at Level 3, which includes coupon optimization, reaching 15.0 and 10.0 percentage points. Sonnet's gains are similar across levels (6.0, 6.0, and 5.0 points). Qwen3.5-27B ties MCTS at 54.0% on Level 2. These results suggest that the benefit depends on both the model and the task; it does not increase uniformly with difficulty. Level 3 contains only 20 cases, compared with 50 each for Levels 1 and 2.

Table 5: Travel sensitivity to λ (%). **Bold** marks the best value of each metric within each model, including ties.

Model	λ	CS Score	PS Score	Comp Score	Case Acc.
Qwen3.6-27B	0.1	86.1	82.8	84.4	31.2
	0.3	86.0	82.1	84.1	30.0
	0.5	86.7	82.5	84.6	32.1
	0.7	**86.9**	83.8	**85.3**	**33.8**
	0.9	85.6	**84.6**	85.1	32.5
Qwen3.5-27B	0.1	77.7	**81.7**	79.7	18.6
	0.3	79.0	**81.7**	**80.3**	**19.6**
	0.5	**79.4**	77.5	78.5	19.0
	0.7	78.2	79.6	78.9	17.9
	0.9	76.5	77.6	77.1	17.0
Sonnet 4.6	0.1	83.9	**81.9**	82.9	29.4
	0.3	84.5	81.5	**83.0**	29.4
	0.5	**85.0**	79.6	82.3	30.1
	0.7	84.5	81.3	82.9	**30.4**
	0.9	83.0	80.7	81.9	29.3

C.4 Hypergraph Weight Sensitivity

Table 5 reports all four overall Travel metrics for the sweep in Section 4.5. We hold $B = 5$, $b = 3$, and $c = 1.414$ fixed across the five values of λ.

Table 6: Full SealQA results. Mean is on a 0–1 scale; accuracy is in %. **Bold** marks the best Mean and accuracy for each model, including ties.

Agent	Qwen3.6-27B			Qwen3.5-27B		
	Mean	Acc. (%)	C/NA/W	Mean	Acc. (%)	C/NA/W
ReAct	0.338	13.5	15/45/51	0.441	12.6	14/70/27
CoT	0.324	14.4	16/40/55	0.437	13.5	15/67/29
Reflexion	0.311	14.4	16/37/58	0.401	13.5	15/59/37
ToT	0.302	17.1	19/29/63	0.446	**17.1**	19/61/31
RAP	0.297	14.4	16/34/61	0.405	13.5	15/60/36
LATS	0.284	15.3	17/29/65	0.423	14.4	16/62/33
FLARE	0.347	18.9	21/35/55	0.432	14.4	16/64/31
HyperMCTS	**0.392**	**21.6**	24/39/48	**0.468**	**17.1**	19/66/26
Agent	Sonnet 4.6			Opus 4.8		
	Mean	Acc. (%)	C/NA/W	Mean	Acc. (%)	C/NA/W
ReAct	0.401	14.4	16/57/38	0.428	19.8	22/51/38
CoT	0.369	14.4	16/50/45	0.428	20.7	23/49/39
Reflexion	0.374	16.2	18/47/46	0.405	22.5	25/40/46
ToT	0.329	22.5	25/23/63	0.405	23.4	26/38/47
RAP	0.360	17.1	19/42/50	0.369	20.7	23/36/52
LATS	0.333	19.8	22/30/59	0.428	26.1	29/37/45
FLARE	0.329	15.3	17/39/55	0.423	26.1	29/36/46
HyperMCTS	**0.446**	**26.1**	29/41/41	**0.491**	**30.6**	34/41/36

C.5 SEALQA RESULTS

Table 6 reports the full metrics for the SealQA results in Table 1 and Section 4.6, together with additional evaluations using Opus 4.8. C, NA, and W denote correct, not-attempted, and wrong answers, respectively, and sum to $N = 111$. Accuracy is $100C/N$, and the partial-credit score (Mean) is $(C + 0.5\,\mathrm{NA})/N$.

For HyperMCTS on SealQA, we use $\lambda = 0.2$ for Qwen3.6-27B and Sonnet 4.6, and $\lambda = 0.5$ for Qwen3.5-27B and Opus 4.8.

On Opus 4.8, HyperMCTS reaches 30.6% accuracy, exceeding the strongest baselines, LATS and FLARE (26.1%), by 4.5 percentage points.