GOING BEYOND SCALAR: SEMANTIC MONTE CARLO TREE SEARCH FOR PLANNING WITH LARGE LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world planning rarely admits a single fixed objective. User queries may combine hard constraints, priorities, soft preferences, and objectives such as cost, convenience, or comfort, making it difficult to encode user intent with a scalar reward. Conventional planning methods, including Monte Carlo Tree Search (MCTS), nevertheless evaluate states using scalar values and select actions through numerical exploration–exploitation criteria such as Upper Confidence bound 1 applied to Trees (UCT), potentially discarding semantic information needed for flexible natural-language planning. To address this limitation, we propose Semantic Monte Carlo Tree Search (S-MCTS), which performs planning directly in a semantic value space. Instead of a scalar value, each state maintains a future situation—the best structured plan discovered through that state—and an LLM-generated summary evaluating it against the user's constraints, priorities, preferences, and objectives. In action selection, S-MCTS replaces UCT with LLM-based semantic exploration–exploitation, where candidate actions are annotated with summaries of their future situations and selected through query-conditioned reasoning. During backpropagation, parent and child summaries are compared, and the parent adopts the child's future situation when it is better. Thus, S-MCTS reformulates value representation, selection, and backup as semantic reasoning, enabling search under heterogeneous and difficult-to-scalarize requirements. Empirical results show that S-MCTS outperforms representative baselines on standard and flexible travel-planning benchmarks.

1 INTRODUCTION

Planning assistants increasingly operate on open-ended natural-language objectives rather than explicit mathematical costs (Huang et al., 2017). A user may simultaneously specify budgets, scheduling preferences, restaurant quality, cuisine, convenience, and comfort (Wang et al., 2025). Plan quality therefore depends on interactions among hard constraints, priorities, preferences, and optimization objectives.

This flexibility challenges conventional planning (Oh et al., 2025), which often reduces candidate solutions to scalar objectives (Cao et al., 2025; Wei et al., 2025). Such scalarization can obscure which requirements are satisfied or violated and is especially restrictive when constraints conflict or have different priorities. Flexible travel-planning studies further show that LLMs may fail to preserve global constraints or prioritize hard requirements over soft preferences (Oh et al., 2025). This motivates value representations that retain the semantic structure of plan quality.

LLMs provide a natural mechanism for such representations because they can interpret concepts such as comfort, convenience, preferences, and priorities without manually defined symbolic objectives (Pyatkin et al., 2022; Jimenez et al., 2024). However, direct generation explores only a limited set of alternatives and can make locally reasonable but globally poor decisions. Tree search addresses this limitation by explicitly exploring multiple futures (Chen et al., 2024), while MCTS is particularly suitable because it combines look-ahead, exploration, and value propagation (Wei et al., 2025; Zhao et al., 2023).

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

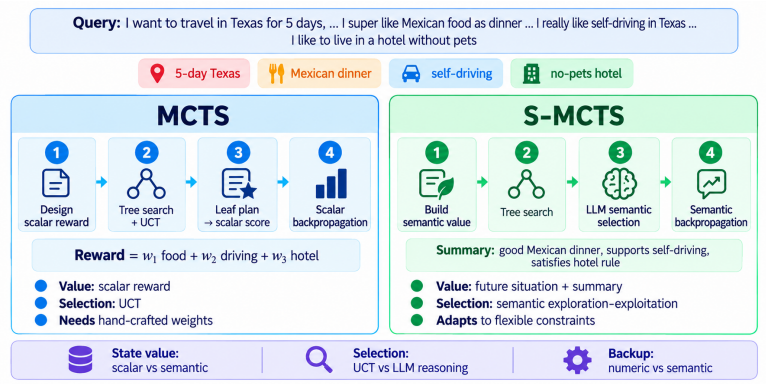


Figure 1: Comparison between MCTS and S-MCTS, given a query with multi-level constraints.

Nevertheless, conventional MCTS is built around a *scalar-valued* search process. A future trajectory is compressed into a numerical value; UCT selects actions by combining this value with a visit-based exploration bonus; and rollout outcomes are propagated toward the root through scalar back-propagation (Świechowski et al., 2023). This creates three information bottlenecks for natural-language planning. First, scalar values discard the semantic structure of a future plan, including which requirements are satisfied and how conflicting requirements are traded off. Second, scalar UCT ignores the fact that LLMs can reason directly about the semantic meaning of candidate actions and their anticipated consequences. Recent work shows that LLM exploration–exploitation behavior is itself strongly influenced by semantic information and semantic priors (Austin et al., 2026), suggesting that discarding this information during selection underutilizes the reasoning capabilities of the model. Third, scalar back-propagation propagates only a compressed outcome through the tree, preventing intermediate states from retaining the rich future plans that justified their values.

These limitations motivate performing the entire search process in the *semantic space*. Instead of assigning each node only a scalar critic, a semantic-valued search can retain a structured future situation together with a natural-language summary of its quality under the original query. Such a representation allows different futures to be compared according to heterogeneous constraints, priorities, preferences, and optimization goals. During selection, exposing the semantic summaries of candidate futures allows the LLM to conduct *semantic exploration–exploitation*: promising actions can be exploited because of the requirements they are expected to satisfy, while less-explored actions can be selected when their futures reveal opportunities to resolve unresolved constraints. During back-propagation, semantic plan-to-plan comparison can repeatedly determine whether a newly discovered future is preferable to the one currently stored at each node, allowing better futures to progressively replace inferior ones from the leaf toward the root. This node-wise semantic propagation preserves the reason behind a value update and provides repeated opportunities to revise earlier search decisions.

Motivated by these observations, we propose **Semantic Monte Carlo Tree Search (S-MCTS)**, which extends MCTS from scalar-valued search to semantic-valued planning. Each node maintains its best discovered structured future situation together with an LLM-generated semantic summary conditioned on the complete user query. S-MCTS then uses these summaries for semantic exploration–exploitation during selection and semantic comparison during back-propagation. By preserving semantic information throughout the search process, S-MCTS is designed for LLM planner to improve the satisfaction of multiple constraints and handle flexible natural-language objectives without requiring heterogeneous user requirements to be manually reduced to a fixed scalar reward.

Our contributions are fourfold: (1) we introduce a semantic value representation based on structured future situations and query-conditioned summaries, extending MCTS to the semantic space; (2) we replace numerical UCT with LLM-based semantic exploration–exploitation; (3) we develop semantic backpropagation that propagates superior future plans through the tree; and (4) we conduct empirical evaluations on standard and flexible travel-planning benchmarks, demonstrating that S-MCTS consistently outperforms representative baselines by more than 10% (20%) on the accuracy of satisfying regular (multi-level) constraints.

2 Related Works

Reasoning-based planning. The strong reasoning capabilities of large language models (LLMs) have motivated their use as general-purpose planners (Wei et al., 2025). Early approaches rely primarily on natural-language reasoning to construct plans. Chain-of-Thought (CoT) elicits explicit intermediate reasoning steps (Wei et al., 2022), while ReAct interleaves reasoning with actions for sequential decision making (Yao et al., 2023b). Self-reflection methods further improve generated solutions through iterative feedback and revision (Shinn et al., 2023; Madaan et al., 2023). Beyond linear reasoning, Tree-of-Thought (ToT) explicitly explores multiple reasoning branches and evaluates intermediate states, providing a search-based mechanism for deliberate problem solving (Yao et al., 2023a). More recent methods scale this idea through evolutionary search or hierarchical decomposition, such as EvoAgent (Yuan et al., 2025), HyperTree Planning (Gui et al., 2025) and deliberate reasoning for planning (Xiong et al., 2025; Gungordu et al., 2026). Despite their flexibility, these methods largely depend on implicit reasoning in natural language, which can become unreliable as planning horizons and the number of interacting constraints increase (Kambhampati et al., 2024).

Solver- and code-augmented planning. To improve reliability, a complementary direction combines LLM reasoning with executable programs, symbolic models, or external solvers. LLMs have been used to construct explicit world models for model-based planning (Guan et al., 2023), while LLM-modulo approaches advocate delegating formal reasoning and verification to external symbolic components (Kambhampati et al., 2024). Formalized planning methods translate natural-language problems into representations that can be solved or verified using formal tools (Hao et al., 2025a;b; Yang et al., 2024). Thought of Search instead asks LLMs to generate executable successor and goal-test functions, turning problem solving into explicit programmatic search (Katz et al., 2024). Related work also investigates interactive code-augmented planning (Liu et al., 2025) and constraint-programming formulations generated from natural language (Michailidis et al., 2024). These approaches improve systematic constraint handling, but typically require translating flexible user objectives into explicit formal representations or solver logic.

MCTS-based planning. Monte Carlo Tree Search (MCTS) provides a principled framework for look-ahead planning by balancing exploration and exploitation during search (Kocsis & Szepesvári, 2006). Recent studies have integrated LLMs into MCTS to improve planning and reasoning. LLM-MCTS employs an LLM as both a commonsense world model and a heuristic policy to guide tree search (Zhao et al., 2023), while Reasoning via Planning (RAP) treats the LLM as a world model and reasoning agent and uses MCTS to explore alternative reasoning trajectories (Hao et al., 2023). GDP-Zero applies open-loop MCTS to goal-oriented dialogue, prompting an LLM to serve as the policy prior, value function, user simulator, and transition model (Yu et al., 2023). Language Agent Tree Search (LATS) further combines MCTS with LLM-based value estimation, self-reflection, and environmental feedback for reasoning and interactive decision making (Zhou et al., 2024). Agent Q integrates guided MCTS with self-critique and preference-based policy learning to improve autonomous agents in complex interactive tasks (Putta et al., 2024). Despite leveraging LLMs within tree search, these approaches largely retain numerical reward or value signals for evaluating and propagating search outcomes.

3 Preliminaries

3.1 LLM Planning

We consider a planning problem specified by a natural-language query q, which may contain hard constraints, priorities, soft preferences, and optimization criteria. A planning problem is defined by a state space $\mathcal{S}$, action space $\mathcal{A}$, and transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. A state s_t represents the partial plan constructed up to step t, while an action a_t denotes the next planning decision, yielding $s_{t+1} = \mathcal{T}(s_t, a_t)$. An LLM planner can be modeled as a policy $\pi_\theta(a_t|s_t, q)$ which selects actions conditioned on the current state and user query.

In the planning problem considered by this work, each state s_t is assumed as the concatenation of actions generated so far, i.e. $s_t := [a_0, a_1, \ldots, a_{t-1}]$, and s_0 is assumed as empty. Therefore, the state transition $\mathcal{T}(s_t, a_t)$ is essentially to append a_t to the tail of s_t.

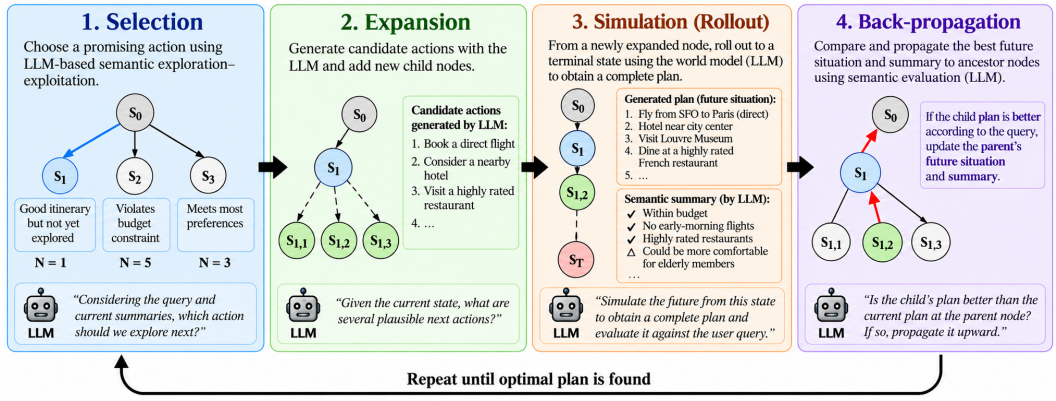


Figure 2: The diagram of four phases of the proposed S-MCTS algorithm. In the selection phase, the agent starts from root node, selects an action (edge), and proceeds to the associated next state (child node), repeating until a leaf node is reached. The expansion phase generates K new actions and creates one child node for every new action. The roll-out phase estimates the future situation for each new child node. The back-propagation updates the future situation and summary of every node on the path selected in the selection phase. Then, the next iteration repeats the same operations until the optimal plan (future situation) at root node has not been updated for sufficiently many iterations.

LLMs are well suited to flexible planning because they can interpret heterogeneous natural-language requirements without manually translating them into fixed symbolic objectives. However, direct generation typically follows a single trajectory and may fail to sufficiently explore alternatives. This motivates combining LLM reasoning with explicit search over multiple candidate futures.

3.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) Kocsis & Szepesvári (2006) incrementally builds a search tree whose nodes represent states and edges represent actions. Each iteration consists of selection, expansion, simulation, and backpropagation.

During selection, MCTS commonly uses UCT (Upper Confidence bounds applied to Trees):

$$a^* = \arg\max_{a \in \mathcal{A}} \left[Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \right] \tag{1}$$

where $Q(s,a)$ is the estimated action value, $N(s)$ and $N(s,a)$ are visit counts, and c balances exploitation and exploration. Define a trajectory of stat-action pairs as $\tau := [s_0, a_0, \ldots, s_{T-1}, a_{T-1}]$.

A rollout is then performed to estimate utility of a future trajectory, typically through a scalar return $G(\tau) = \sum_{k=t}^{T-1} \gamma^{k-t} r(s_k, a_k)$ which is propagated back to ancestor nodes to update their value estimates.

4 Methodology

4.1 Overview

We propose Semantic Monte Carlo Tree Search (S-MCTS) for planning under flexible natural-language requirements. Given a user query q, the goal is to construct a plan that best satisfies heterogeneous requirements, including hard constraints, priorities, soft preferences, and optimization objectives. Following conventional MCTS, we construct a search tree in which each node represents a planning state and each edge represents a planning action. The conventional MCTS-based LLM planning, such as RAP (Hao et al., 2023), adopts the same state-action tree formulation and organizes MCTS into selection, expansion, simulation, and back-propagation.

The key difference is how S-MCTS represents and propagates value. Conventional MCTS maintains a scalar state-action value $Q(s,a)$, uses UCT to balance exploration and exploitation, and propagates numerical rewards through the tree (Kocsis & Szepesvári, 2006). Such a formulation requires

heterogeneous user requirements to ultimately be compressed into a scalar. S-MCTS instead performs search in a semantic value space. Each state maintains (i) a future situation, representing the best plan discovered so far through that state, and (ii) a summary, which evaluates the future situation against the user query. Selection and back-propagation are then implemented through LLM-based semantic reasoning rather than arithmetic operations over scalar values. The diagram illustrating the four phases of S-MCTS is in Figure 2. The detailed algorithm table is in Appendix B.

Formally, a planning problem consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, and transition function $s' = \mathcal{T}(s, a)$. The initial state s_0 specifies the planning problem and available information. Before solving any query, the first step is to extract action templates which will be used in the prompt of expansion and roll-out to guide the LLM to generate actions in the correct form. The details are introduced in Appendix C.1. A search trajectory $\tau = (s_0, a_0, s_1, \ldots, a_{T-1}, s_T)$ corresponds to a complete plan $P(\tau)$. After sufficient iterations, S-MCTS returns the future situation maintained at the root as the found optimal plan.

Remark. Note that in this work, we do not consider user queries which have multiple optimal solutions, e.g. alternative Pareto-relevant plans. Even though there is only one optimal plan, the scalar value or reward function still cannot adapt to the flexibility in multi-level constraints, .e.g, missing budget constraint.

4.2 Semantic Value Function

The central component of S-MCTS is a semantic value function that represents the future value of a state using structured language rather than a scalar. For each state s, we define its future situation as

$$\mathcal{F}(s) = \operatorname*{Best}_{\tau:\, s\in\tau} P(\tau),$$

where τ is the trajectory passing through state s, and $P(\tau)$ denotes the structured plan of trajectory τ. Here, the maximization is semantic rather than numerical: $\mathcal{F}(s)$ is the best plan found so far among explored trajectories going through s. For example, in travel planning, $\mathcal{F}(s)$ may specify the complete itinerary including transportation, accommodation, restaurants, and attractions. Retaining the actual full plan is important because the full plan can provide more information on query satisfaction than scalar values, and be forwarded to the root to compete for the optimal plan.

To provide a compact representation to reduce token consumption in action selection, we additionally associate each state with a semantic summary

$$\mathcal{V}(s; q) = E_{\mathrm{LLM}}(q, \mathcal{F}(s))\,,$$

where E_{LLM} is an LLM-based evaluator. The summary describes how well the full plan of the future situation satisfies the user query, including, when applicable, which constraints are satisfied or violated, whether higher-priority requirements are respected, how preferences are accommodated, and how the plan performs on objectives such as cost or comfort.

Importantly, $\mathcal{V}(s; q)$ is not required to output a numerical score. Instead, it preserves the semantic reasons that make one future preferable to another. This enables the same value representation to adapt to different queries without manually defining query-specific reward weights.

We further define an LLM-based comparison operator $\mathcal{C}_q(v_1, v_2) \in \{v_1, v_2\}$, which takes two semantic summaries and determines which represents the better plan under query q. This operator serves as the semantic counterpart of the $\max$ operation used in conventional value-based search, which is the foundation of semantic back-propagation.

4.3 Semantic Selection

In each iteration of S-MCTS, as shown in Figure 2, the first phase is the selection which selects a portion of the existing search tree that is most promising for further expansion in the next phase. Starting from the root node (i.e., initial state s_0), at each level of the tree, the algorithm selects an action and proceeds to the connected child node as the next node. The selection phase terminates when a leaf node of the current search tree is reached.

During each action selection, in contrast to UCT equation 1 adopted by conventional MCTS, the proposed S-MCTS replaces this fixed numerical objective with an LLM-based semantic exploration–exploitation operator which can adapt to the flexibility of practical user queries. Specifically, for a

state s with candidate actions

$$\mathcal{A}(s) = \{a^1, \ldots, a^K\},$$

each action is associated with its child state $s_i = \mathcal{T}(s, a^i)$ and, when available, the child's semantic summary ($\mathcal{V}$) and number of visits (N):

$$a^i \quad \longrightarrow \quad \mathcal{V}(s_i; q), N(s, a^i).$$

where $N(s, a^i)$ refers to how many times a^i has been selected at node s in previous iterations. The LLM receives the user query q together with the candidate actions and their current semantic summaries and number of visits $N(s, a^i)$, and selects

$$a^* = \text{Select}_{\text{LLM}}\left(q, \{(a^i, \mathcal{V}(s_i; q), N(s, a^i))\}_{i=1}^K, C\right).$$

The context C in the prompt explicitly instructs the LLM to balance two considerations: exploitation, by preferring actions whose discovered futures close to or already satisfy the query well, and exploration, by considering branches whose potential remains insufficiently explored, i.e. relatively small number of visits. Specifically, LLM is instructed to use its own judgment to balance between exploration and exploitation.

This design allows exploration decisions to depend on the semantics of the planning problem and action's future consequences. For example, an under-explored branch that may satisfy a critical high-priority constraint can be preferred over a well-explored branch that only improves a low-priority preference. Such distinctions are difficult to express through a scalar UCT coefficient. This can also explain why S-MCTS still outperforms in regular planning benchmarks where every constraint in the query is in the same priority level.

4.4 Expansion and Future-situation Roll-out

Expansion. Once selection reaches a leaf node s_t, e.g. node S_1 in Figure 2, S-MCTS proceeds to the expansion phase. In this phase, this leaf node is expanded by prompting LLM and generating a set of candidate planning actions

$$\{a_t^1, \ldots, a_t^K\} \sim \pi_{\text{expansion}}(\cdot \mid s_t, q),$$

subject to the currently available actions and planning constraints. Each action a_t^i produces a successor $s_{t+1}^i = \mathcal{T}(s_t, a_t^i)$, which is added to the search tree as a new child node. This follows the general MCTS structure in which expansion adds candidate actions and their resulting states. As stated in Section 3.1, since every state s_t is a sequence of actions generated so far, the transition function $\mathcal{T}$ appends action a_t^i to s_t to form s_{t+1}^i and build a new child node s_{t+1}^i of s_t. In implementations, this K is chosen to be 3.

Roll-out. For every new child node s_{t+1}^i, S-MCTS performs a rollout to obtain a complete candidate plan as the estimate of its future situation, where the LLM-based rollout policy $\pi_{\text{rollout}}(\cdot|s, q)$ is used. For example, starting from s_{t+1}^i, the rollout policy repeatedly proposes actions by prompting LLM until a terminal planning state, i.e., maximal number of traveling days, is reached:

$$s_{t+1}^i \to a_{t+1} \to s_{t+2} \to \cdots \to s_T.$$

Since $s_{t+1}^i = [a_0, \ldots, a_{t-1}, a_t^i]$, the resulting plan $P(\tau^i)$ includes actions from a_0 to a_{T-1}, and can form an initial candidate plan as future situation for node s_{t+1}^i. Then, its summary is generated as

$$v_\tau^i = E_{\text{LLM}}(q, P(\tau^i)).$$

Since the new child node s_{t+1}^i has no previously stored future situation, we initialize

$$\mathcal{F}(s_{t+1}^i) \leftarrow P(\tau^i), \qquad \mathcal{V}(s_{t+1}^i; q) \leftarrow v_\tau^i.$$

Repeat the roll-out operation for every new child node s_{t+1}^i whose future situation and summary are initialized as above. Then, node s_t, where the selection phase stopped, is not a leaf node anymore, and every child node of s_t has its summary of future situation, providing sufficient action summaries for any future action selection conducted at node s_t.

Conventional MCTS uses simulation to estimate expected future scalar rewards; RAP (Hao et al., 2023), for instance, rolls out future states until termination and uses resulting numerical rewards to estimate Q-values. In contrast, S-MCTS retains the concrete future situation (full plan) produced by the rollout, thereby avoiding premature compression of future consequences into a number.

4.5 SEMANTIC BACK-PROPAGATION

After a rollout discovers a candidate future, conventional MCTS propagates numerical rewards toward the root to update state-action values. S-MCTS instead introduces semantic back-propagation. Consider two consecutive nodes s and s' on the propagation path, where s' is a child of s. We compare their summaries under the original query:

$$\mathcal{C}_q\left(\mathcal{V}(s;q), \mathcal{V}(s';q)\right).$$

If the child contains a better discovered future situation,

$$\mathcal{V}(s';q) \succ_q \mathcal{V}(s;q),$$

we perform the update

$$\mathcal{F}(s) \leftarrow \mathcal{F}(s'), \qquad \mathcal{V}(s;q) \leftarrow \mathcal{V}(s';q).$$

Otherwise, the parent's semantic value remains unchanged. The procedure is repeated recursively until the root is reached.

Consequently, when a promising plan is discovered deep in the search tree, the entire structured future situation and its semantic assessment (summary) propagate toward earlier planning decisions. In this case, the future situation of the root node s_0 keeps the best complete plan discovered up to the current iteration.

5 EXPERIMENTS

In this section, we first introduce experiment setups in Section 5.1, then present the main results and complexity analysis on regular planning benchmarks in Section 5.2 and 5.3. Then, we introduce the results on flexible planning in Appendix D, ablation study in Appendix E and token cost analysis in Appendix F. Finally, we explain the advantages of S-MCTS in Appendix G.

5.1 EXPERIMENTAL SETUP

Datasets. **TravelPlanner** is a real-world benchmark requiring complete itineraries that satisfy commonsense and explicit user constraints (Xie et al., 2024). Plans jointly determine transportation, accommodation, restaurants, and attractions under requirements such as budget, cuisine, room type, house rules, and transportation preferences, testing consistency across interdependent decisions.

Natural Plan evaluates LLM planning across realistic domains (Zheng et al., 2024). We focus on *Trip Planning*, where itineraries must satisfy city-stay durations, fixed visit dates, and direct-flight constraints. Following prior work (Gui et al., 2025; Deik et al., 2026), we use it to evaluate search over alternative city orderings and global itinerary consistency.

Flex-TravelPlanner extends TravelPlanner to prioritized and potentially conflicting requirements (Oh et al., 2025). We consider its priority-aware setting, where satisfying all constraints may be infeasible and higher-priority requirements should dominate lower-priority preferences. This directly tests S-MCTS's ability to reason about semantic trade-offs rather than merely counting satisfied constraints. We follow their ideas but generate queries with multi-level constraints by ourselves.

Baselines. We compare S-MCTS with representative prompting, search, and classical planning methods. **Direct** generates a plan without explicit reasoning or search. **CoT** performs sequential reasoning (Wei et al., 2022), while **ToT** explores multiple branches using LLM-based heuristic evaluation (Yao et al., 2023a). **RAP** uses MCTS with an LLM world model, scalar rewards, and UCT selection (Hao et al., 2023), whereas **LATS** augments MCTS with LLM evaluation, reflection, and environmental feedback (Zhou et al., 2024). We also include **CPMPy**, a classical constraint-programming framework that formulates problems as variables and constraints and solves them with a generic CP solver (Michailidis et al., 2024). These baselines allow us to evaluate the benefits of semantic value representation, semantic back-propagation, and semantic exploration–exploitation.

Evaluation Metrics. For **TravelPlanner**, we follow the original protocol (Xie et al., 2024) and report Delivery Rate (DR), Commonsense Constraint Pass Rate (CPR), Hard Constraint Pass Rate

Table 1: Main results of different planning methods across three benchmarks. DR, CPR, HCPR, and SR denote Delivery Rate, Commonsense Pass Rate, Hard Constraint Pass Rate, and Success Rate in %, respectively. For Natural Plan, we report the Success Rate on the Trip Planning task. For Flex-TravelPlanner, we report the overall SR in %. The best results are highlighted in **bold**.

Model	Method	TravelPlanner						Natural	Flex
		DR	CPR		HCPR		SR		
			Micro	Macro	Micro	Macro		SR	SR
GPT-5.6-luna	Direct	100	97.5	91.2	87.5	83.3	80.2	78.1	79.2
	CoT	100	98.1	93.1	89.2	84.5	83.9	80.2	83.9
	ToT	96.5	98.3	91.5	87.2	81.2	76.2	66.3	71.3
	LATS	100	91.4	89.2	86.4	85.5	84.3	84.2	78.2
	RAP	100	87.2	86.3	83.3	79.3	78.1	78.2	73.1
	CPMPy	65.2	54.3	52.4	47.2	45.3	39.2	21.3	28.5
	S-MCTS	100	**98.6**	**96.3**	**93.4**	**92.1**	**91.9**	**90.1**	**87**
GPT-5-mini	Direct	100	**98.3**	90.3	88.1	84.2	81.1	75.2	77.2
	CoT	100	97.2	92.9	88.3	83.2	82.3	78.1	80.1
	ToT	94.5	97.5	90.2	85.2	80.1	73.2	60.2	67.2
	LATS	100	90.1	88.5	85.3	83.4	82.7	80.1	75.2
	RAP	100	86.2	83.3	82.1	79.4	76.2	74.3	71.1
	CPMPy	59.2	56.3	52.3	45.3	42.1	37.3	18.1	22.6
	S-MCTS	100	97.8	**95.9**	**93.8**	**92.3**	**90.2**	**88.9**	**84.1**
Gemini-2.5-Pro	Direct	100	99.2	92.3	89.5	85.6	84.5	76.2	77.5
	CoT	100	99.5	94.2	92.6	87.2	86.1	79.7	82.1
	ToT	100	99.1	92.1	90.5	84.4	82.2	69.3	73.3
	LATS	100	97.2	89.1	88.2	87.1	85.3	82.5	80.5
	RAP	100	96.3	87.5	85.3	83.3	80.4	80.3	76.3
	CPMPy	65.7	63.2	62.3	56.3	53.2	45.1	32.2	29.1
	S-MCTS	100	**99.7**	**98.3**	**95.2**	**94.9**	**93.9**	**92.2**	**89.3**

(HCPR), and Success Rate (SR). CPR and HCPR use both micro scores, measuring individual constraint satisfaction, and macro scores, measuring whether all constraints in a category are satisfied.

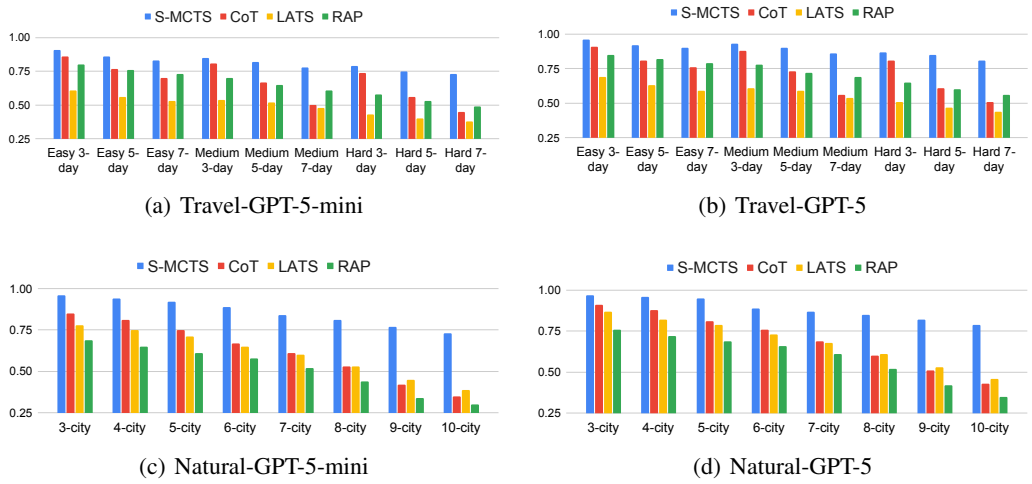
For **Natural Plan–Trip Planning**, we follow the original setting (Zheng et al., 2024) and report **SR**, defined as the fraction of generated plans exactly matching the unique reference plan.

For **Flex-TravelPlanner**, we report overall SR and priority-wise CPR (Oh et al., 2025). A plan satisfying the global budget while violating one conflicting constraint is still counted as successful. Appendix D further reports CPR at each priority level to evaluate whether higher-priority requirements are preserved under conflicts.

5.2 Main Results

The main results are shown in Table 1. S-MCTS consistently achieves the strongest constraint-satisfaction performance across all three benchmarks. On TravelPlanner, S-MCTS substantially improves both hard-constraint satisfaction and overall success, indicating that semantic search better preserves requirements across interacting transportation, accommodation, dining, and budget decisions. On Natural Plan–Trip Planning, S-MCTS obtains higher constraint-satisfaction and plan-generation accuracy, demonstrating that the advantage is not restricted to the richer TravelPlanner environment. The improvement is most pronounced on Flex-TravelPlanner, where requirements explicitly conflict across different priority levels. This is a particularly challenging regime for existing LLM planners: prior work shows that models can incorrectly sacrifice higher-priority global constraints, such as budget, to accommodate lower-priority preferences (Oh et al., 2025). S-MCTS is substantially more robust to such conflicts, demonstrating the benefit of preserving the semantic content of candidate futures throughout search. We report the SR of queries with global and local constraints in Table 1, and other results are presented in Appendix D.

We notice there is a recent work which proposed SCOPE (Deik et al., 2026) and achieved very good results on TravelPlanner and Trip Planning datasets. However, since it is designed for regular



(a) Travel-GPT-5-mini

(b) Travel-GPT-5

(c) Natural-GPT-5-mini

(d) Natural-GPT-5

Figure 3: Success rate vs query complexity on Travel Planner in the first row and Natural Planning datasets in the second row.

multi-constraint sanctification, S-MCTS can outperform SCOPE on flexible constraint satisfaction, which is further discussed in Appendix D.

5.3 Success Rate over Complexity

We further investigate the robustness of different planning methods as query complexity increases. For **TravelPlanner**, we characterize complexity jointly by query difficulty (easy, medium, and hard) and trip duration (3, 5, and 7 days), following the benchmark setting (Xie et al., 2024; Gui et al., 2025). Longer and more difficult queries involve more planning steps and interacting constraints, substantially increasing the difficulty of producing a globally valid itinerary. For **Natural Plan–Trip Planning**, complexity is measured by the number of cities to visit, where more cities induce a larger combinatorial space of possible itineraries and more temporal and flight-connectivity constraints (Zheng et al., 2024). For every complexity, we randomly sample 100 queries in that complexity level from the dataset.

As shown in Figure 3, S-MCTS consistently outperforms all baselines, including CoT, LATS, and RAP, and its advantage becomes more pronounced as complexity increases. CoT is particularly sensitive to query complexity: its success rate decreases substantially on longer, harder TravelPlanner queries and Trip Planning instances involving more cities, since errors accumulated along a single reasoning trajectory cannot be effectively corrected. In contrast, LATS and RAP perform explicit search over multiple candidate trajectories and therefore exhibit smaller performance degradation as complexity grows. Nevertheless, S-MCTS remains consistently stronger because its semantic future situations and summaries provide richer information for action selection and back-propagation, enabling the search to better distinguish promising branches in increasingly large and constraint-heavy planning spaces.

6 Conclusion

We presented **Semantic Monte Carlo Tree Search (S-MCTS)**, which extends conventional MCTS from scalar evaluation to semantic reasoning. S-MCTS maintains a structured future situation and semantic summary at each node, and uses them for semantic action selection and back-propagation.

Experiments on TravelPlanner, Natural Plan–Trip Planning, and Flex-TravelPlanner show that S-MCTS consistently outperforms strong prompting and tree-search baselines, with larger gains on more complex and conflicting queries. Ablations confirm the importance of semantic value representation and semantic back-propagation. Although S-MCTS consumes more tokens, its cost remains controllable, making semantic tree search a promising direction for real-world planning.

7 AI ASSISTANCE

In this work, we used generative AI tools for no tasks with required disclosure. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for draft parts of a research paper, and summarize or analyse existing literature. 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

David Eric Austin, Kaheer Suleman, and Jackie Chi Kit Cheung. In-context exploration-exploitation is biased by semantic priors. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=PbNQoIllIM.

Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. Large language models for planning: A comprehensive and systematic survey. *arXiv preprint arXiv:2505.19683*, 2025.

Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for llm planning? it depends on the discriminator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13659–13678, 2024.

Derrick Goh Xin Deik, Quanyu Long, Zhengyuan Liu, Nancy Chen, and Wenya Wang. Programming over thinking: Efficient and robust multi-constraint planning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 43785–43837, 2026.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. In *Advances in Neural Information Processing Systems*, 2023.

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 *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Oguzhan Gungordu, Siheng Xiong, and Faramarz Fekri. Pathwise: Planning through world model for automated heuristic design via self-evolving llms. *arXiv preprint arXiv:2601.20539*, 2026.

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. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.507.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. Large language models can solve real-world planning rigorously with formal verification tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025a.

Yilun Hao, Yang Zhang, and Chuchu Fan. Planning anything with rigor: General-purpose zero-shot planning with llm-based formalized programming. In *International Conference on Learning Representations*, 2025b.

C Derrick Huang, Jahyun Goo, Kichan Nam, and Chul Woo Yoo. Smart tourism technologies in travel planning: The role of exploration and exploitation. *Information & management*, 54(6): 757–770, 2017.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. Thought of search: Planning with language models through the lens of efficiency. In *Advances in Neural Information Processing Systems*, 2024.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning*, pp. 282–293. Springer, 2006. doi: 10.1007/11871842_29.

Anthony Z. Liu, Xinhe Wang, Jacob Sansom, Yao Fu, Jongwook Choi, Sungryull Sohn, Jaekyeom Kim, and Honglak Lee. Interactive and expressive code-augmented planning with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Kostis Michailidis, Dimos Tsouros, and Tias Guns. Constraint modelling with llms using in-context learning. In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming*, 2024.

Juhyun Oh, Eunsu Kim, and Alice Oh. Flex-travelplanner: A benchmark for flexible planning with language agents. *arXiv preprint arXiv:2506.04649*, 2025.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents, 2024.

Valentina Pyatkin, Jena D Hwang, Vivek Srikumar, Ximing Lu, Liwei Jiang, Yejin Choi, and Chandra Bhagavatula. Reinforced clarification question generation with defeasibility rewards for disambiguating social and moral situations. *arXiv preprint arXiv:2212.10409*, 2022.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: a review of recent modifications and applications: M. świechowski et al. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

Kaimin Wang, Yuanzhe Shen, Changze Lv, Xiaoqing Zheng, and Xuan-Jing Huang. Triptailor: A real-world benchmark for personalized travel planning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9705–9723, 2025.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. Plangenllms: A modern survey of llm planning capabilities. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Siheng Xiong, Ali Payani, Yuan Yang, and Faramarz Fekri. Deliberate reasoning in language models as structure-aware planning with an accurate world model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31900–31931, 2025.

Yu'an Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. Can llms reason in the wild with programs? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9806–9829, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Xiao Yu, Maximillian Chen, and Zhou Yu. Prompt-based monte-carlo tree search for goal-oriented dialogue policy planning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7101–7125. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.439.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024.