electronics

MDPI

Article

SLTP: A Symbolic Travel-Planning Agent Framework with Decoupled Translation and Heuristic Tree Search

Debin Tang [1,†], Qian Jiang [1,†], Jingpu Yang [2], Jingyu Zhao [1], Xiaofei Du [3], Miao Fang [1,*] and Xiaofei Zhang [4,*]

[1] School of Computer and Communication Engineering, Northeastern University, Qinhuangdao 066004, China; 202312623@stu.neu.edu.cn (D.T.); 202316187@stu.neu.edu.cn (Q.J.); zhaojingyu@mails.neu.edu.cn (J.Z.)
[2] Research Institute of Unmanned Systems, Beihang University, Beijing 100191, China; jingpuyang@buaa.edu.cn
[3] School of Mechanical Engineering, Nanjing Institute of Technology, Nanjing 211167, China
[4] School of Management, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China
* Correspondence: fangmiao@neuq.edu.cn (M.F.); zxffs@dlut.edu.cn (X.Z.)
† These authors contributed equally to this work.

Abstract

Large language models (LLMs) demonstrate outstanding capability in understanding natural language and show great potential in open-domain travel planning. However, when confronted with multi-constraint itineraries, personalized recommendations, and scenarios requiring rigorous external information validation, pure LLM-based approaches lack rigorous planning ability and fine-grained personalization. To address these gaps, we propose the Symbolic LoRA Travel Planner (SLTP) framework—an agent architecture that combines a two-stage symbol-rule LoRA fine-tuning pipeline with a user multi-option heuristic tree search (MHTS) planner. SLTP decomposes the entire process of transforming natural language into executable code into two specialized, sequential LoRA experts: the first maps natural-language queries to symbolic constraints with high fidelity; the second compiles symbolic constraints into executable Python planning code. After reflective verification, the generated code serves as constraints and heuristic rules for an MHTS planner that preserves diversified top-K candidate itineraries and uses pruning plus heuristic strategies to maintain search-time performance. To overcome the scarcity of high-quality intermediate symbolic data, we adopt a teacher–student distillation approach: a strong teacher model generates high-fidelity symbolic constraints and executable code, which we use as hard targets to distill knowledge into an 8B-parameter Qwen3-8B student model via two-stage LoRA. On the ChinaTravel benchmark, SLTP using an 8B student achieves performance comparable to or surpassing that of other methods built on DeepSeek-V3 or GPT-4o as a backbone.

Keywords: travel planning; neuro-symbolic planning; LoRA fine-tuning; heuristic tree search; teacher–student distillation

Check for updates

Academic Editor: Ioannis Hatzilygeroudis

Received: 23 December 2025
Revised: 14 January 2026
Accepted: 16 January 2026
Published: 18 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Under the momentum of the digital era, the global tourism industry is undergoing a technology-driven transformation from traditional, centralized services to highly personalized and decentralized ecosystems [1–3]. With the explosive growth of mobile internet and social media, user-generated content (UGC) proliferates in unprecedented volume. Travelers planning itineraries no longer choose from a few templated packages offered by travel agencies, but must instead navigate massive, fragmented POI information, real-time reviews, and complex geographic and transport data [4,5]. This information abundance

enriches options but also causes severe overload and decision fatigue. Traditional travel-recommendation systems [6] focus primarily on POI matching and rating prediction. They struggle to aggregate heterogeneous information streams and are ill-suited for open-ended, multi-day, cross-region planning under combinatorial constraints such as time windows, geographic distance, and budget. As a result, providing a logically consistent, globally feasible itinerary remains challenging. Building an agent that intelligently integrates massive information, satisfies complex constraints, and delivers deep personalization is both a cross-disciplinary research hotspot and a key industrial problem.

Large language models (LLMs), owing to their superior natural language understanding, complex reasoning, and text generation, are seen as a promising solution [7–9]. LLM-Agent architectures (e.g., ReAct [10]) decouple reasoning and action and have shown potential in open-world complex tasks, seemingly enabling end-to-end intelligent planning. However, travel planning is fundamentally a high-order combinatorial optimization involving time, budget, and geography, requiring uncompromising logical rigor and constraint satisfaction. When LLM agents face complex Chinese travel queries, their inherent weaknesses become evident [11].

The main weakness is fragility in logical reasoning: LLMs rely on internal parameterized knowledge and autoregressive generation; when handling multi-step, dependent planning logic, they often produce itineraries violating hard constraints, physically infeasible routes, or internally inconsistent schedules—a phenomenon we call "planning hallucination" [12,13]. Recent benchmarks like TravelPlanner [14] have quantitatively demonstrated that even advanced LLMs struggle to stay on task or track multiple constraints in real-world planning scenarios. This uncertainty prevents pure-LLM planners from achieving the fidelity required in practical applications. To mitigate LLM limitations in rigorous reasoning and constraint verification, the research community has increasingly turned to neuro-symbolic (NeSy) [15] planning paradigms. The philosophical core of NeSy is to combine LLMs' perception of unstructured data with symbolic logic's exact derivation and verification. The technical core is natural-language-to-symbolic-language (NL2SL) translation, converting fuzzy user constraints into formal, verifiable symbolic logic or executable code. The symbolic code can then be used by external planners or verifiers to guarantee logical correctness. This paradigm aligns with the "Code as Policies" [16] philosophy, suggesting that executable code offers a more robust representation for embodied reasoning than unstructured text. However, NL2SL itself faces severe efficiency and accuracy challenges: achieving high-fidelity, error-free NL2SL requires powerful base models to reliably map complex semantics (e.g., "multi-day itinerary within budget where daily activity intensity is moderate") into syntactically and logically correct Python code for strict machine verification. Reliance on extremely large models (GPT-4o [17], DeepSeek-V3 [18]) leads to high training and inference costs, limiting low-cost, large-scale deployment of high-performance agents.

Beyond satisfying hard constraints, intelligent agents must deliver deep personalization. Typical planners only address shallow preferences. True fidelity personalization requires modeling the user's planning style and cognitive preferences: some users prefer highly structured schedules, while others prefer flexible alternatives for spontaneous exploration. To support multi-option decision making, an agent must capture and reflect these cognitive-style preferences and return several high-quality, diverse candidate plans for user comparison [19]. Decision psychology shows that users build trust by comparing multiple options; therefore, an agent should provide multi-option, high-fidelity recommendations rather than a single optimum.

To tackle the NL2SL accuracy requirement, intermediate data shortage, high computational cost, and multi-option personalization, we propose the SLTP framework. SLTP

integrates a two-stage symbol-rule LoRA [20] fine-tuning pipeline with an MHTS planner. It aims to produce a lightweight yet powerful NeSy planning agent through parameter-efficient fine-tuning and knowledge transfer, and to improve personalization through multi-option planning.

The SLTP framework's core innovations lie in precise task decoupling and efficiency optimizations across the planning pipeline:

We split the complex NL2SL pipeline into two sequential LoRA experts: Stage I maps natural language queries to structured symbolic constraints, ensuring accurate semantic extraction and formatting; Stage II compiles symbolic constraints into executable Python planning code. The generated code serves as hard constraints and heuristic guidance for downstream multi-option tree search, enabling a high-fidelity pipeline from language to verifiable code. Teacher–student knowledge distillation [21,22] to overcome high-quality intermediate data scarcity and deployment costs. A large teacher model (DeepSeek-V3.2 [18]) generates high-fidelity structured symbolic constraints and executable Python planning code examples that are human-verified and used as hard-label targets to distill logical reasoning into an 8B-parameter Qwen3-8B [23] student model via two-stage LoRA. This allows a lightweight model to achieve performance comparable to or exceeding the teacher in NL2SL tasks while enabling cost-efficient deployment. Recent studies on model specialization, such as Orca 2 [24], have similarly demonstrated that smaller models can master complex reasoning patterns when trained on curated, explanation-rich signals.

A symbol-driven MHTS planner. The generated Python code is used at search time for dynamic constraint validation and pruning. The planner employs a Top-K strategy to retain diversified candidate itineraries, budget-aware heuristics, and systematic pruning to balance correctness, diversity, and runtime efficiency.

We validate SLTP extensively on the ChinaTravel benchmark. The results show significant advantages in constraint satisfaction and planning rationality metrics, demonstrating that SLTP effectively addresses complex neuro-symbolic planning for travel. SLTP lays a robust foundation for future agents that combine logical rigor, computational efficiency, and deep personalization.

2. Related Work

2.1. LLM Agents and Neuro-Symbolic Planning Paradigms

Large language models have become central to building general-purpose agents due to their capabilities in language understanding, complex reasoning, and instruction following. Early work, such as Chain-of-Thought, improved logical task performance by encouraging explicit intermediate reasoning.

In planning, frameworks like the ReActdecouple model "Thought" from external "Action," enabling agents to interact with open-world environments and partially compensating for LLM knowledge staleness and numerical precision issues. Furthermore, frameworks like ToolLLM [25] have extended this capability to handle massive real-world APIs, yet they still struggle with the global consistency of long-horizon plans. However, when purely autoregressive LLM agents face travel planning—a high-order combinatorial optimization—their logical fragility surfaces: these models easily fall into "planning hallucination," producing logically inconsistent or geographically infeasible itineraries with low constraint satisfaction.

To address LLM logical deficits, approaches such as LLM-Modulo-Verifier [26,27] incorporate external symbolic verifiers to provide feedback and iterative self-correction. While effective for simple planning, error-correction effectiveness decays rapidly after 3–5 iterations for complex multi-day itineraries and can introduce emergent errors where correcting some mistakes causes new ones, yielding unstable macro-level performance.

Recent evaluations [28] indicate that without external ground-truth, LLMs often fail to self-correct intrinsic reasoning errors, leading to a "correction loop" failure.

Consequently, NeSy methods—combining LLM perception with symbolic logic precision—have gained traction. Typical NeSy pipelines translate NL to a domain-specific language (NL2DSL) that an external search/verifier uses to guide or validate search. While this improves rigor, it depends critically on NL2DSL translation fidelity and is constrained by strict search-time budgets. Other methods, like TTG [29], translate natural-language requirements into mixed-integer linear programs (MILP) solvable by optimization solvers, theoretically guaranteeing optimality; but MILP size grows exponentially with complex constraints and POI counts, making practical use infeasible due to computational cost.

Thus, while NeSy addresses logical rigor conceptually, achieving high-fidelity NL2SL translation efficiently without excessive reliance on ultra-large models remains a core challenge.

2.2. Parameter-Efficient Learning, Knowledge Transfer, and Multi-Option Heuristic Tree Search

Model efficiency, knowledge transfer, and user experience are key practical concerns when deploying high-performance LLM agents.

Parameter-efficient fine-tuning methods, notably Low-Rank Adaptation (LoRA), add low-rank update matrices to frozen pre-trained weights, greatly reducing tuning storage and computation load. LoRA enables adapting large models to downstream tasks under resource constraints. Most prior LoRA applications target single end-to-end tasks; leveraging LoRA for fine-grained, sequential task specialization remains underexplored.

Knowledge distillation is a well-established method for transferring capabilities from a large teacher to a smaller student. In reasoning-heavy tasks, distillation can capture intermediate reasoning traces. However, in neuro-symbolic planning—where intermediate outputs are formal code and symbolic constraints—how to distill high-quality intermediate datasets and instill teacher-level logical reasoning into a compact student via LoRA is not fully studied. Frameworks like RAP [30] emphasize using world models for planning, inspiring our approach to distill these reasoning traces into executable symbolic constraints.

Regarding user experience, traditional planners typically aim for a single optimal solution, while real users prefer multiple diverse, high-quality options for comparison. Existing search-based planners focus on efficiency and single-solution optimization and lack systematic mechanisms to generate and evaluate multiple diverse high-quality recommendations under strict constraints. While Tree of Thoughts [31] introduced search algorithms to LLM reasoning, it primarily targets general logic puzzles and lacks the domain-specific heuristics required for budget-constrained trajectory planning.

3. Method

3.1. Overall Architecture of SLTP

SLTP is designed as a high-performance, lightweight, and logically rigorous travel-planning agent. It combines parameter-efficient fine-tuning and symbol-driven MHTS. The framework follows a two-stage pipeline: high-fidelity symbolic conversion and symbol-driven planning search (Figure 1). The process starts with a user's natural-language query Q. The dual-expert LoRA knowledge-transfer module produces structured symbolic constraints S and executable Python 3.8 planning code C. The code C serves as a dynamic constraint validator that drives the MHTS planner, which, together with an external knowledge base K, outputs up to K constraint-satisfying itineraries.

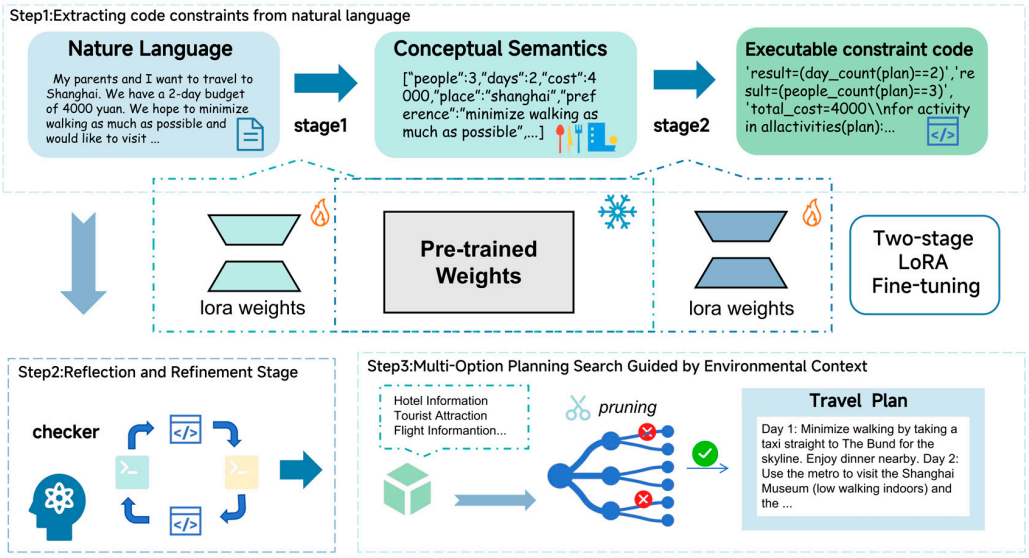


Figure 1. The flowchart illustrates the overall framework of our work: it converts natural language into a symbolic constraint language, then into executable constraint code, and finally generates the corresponding planning solutions.

Symbolic conversion summary: Q (natural-language query) → S (symbolic constraints) → C (executable code) → R (final multi-option itineraries). The framework's novelty lies in task decoupling and distillation for NL2SL and in using generated executable code as search-time validators. In planning (Section 3.4), SLTP treats the LLM as a one-shot translator whose generated executable code acts as hard constraints and heuristic rules guiding efficient MHTS. The planner integrates budget-awareness, temporal heuristics, systematic pruning, and a Top-K mechanism to produce K feasible and diverse itineraries.

3.2. High-Fidelity NL2SL Conversion via Knowledge Distillation

3.2.1. Building and Verifying a Symbolic Middleware Dataset

To construct high-quality training data, we employ a powerful inference teacher model to generate symbolic middleware. The distillation dataset consists of triplets (Q_i, S_i, C_i), where Q_i is a user query, S_i is a structured symbolic constraint representation, and C_i is the executable Python planning code derived from S_i. The teacher model is prompted carefully to produce S_i and C_i strictly follow the DSL syntax.

To ensure the high quality of the symbolic middleware generated by the teacher model (DeepSeek-V3.2), we designed a "Constraint-Check" prompting strategy. We explicitly defined the DSL grammar rules in the prompt and enforced a Chain-of-Thought process where the model must output the intermediate symbolic representation S_i before generating the executable code C_i.

In the following generation, we implemented a semi-automated verification pipeline: samples were first filtered by a syntax checker and then executed in the sandbox environment. Only samples that compiled successfully and produced a valid graph path (even if not optimal) were retained. Ultimately, we constructed a high-quality distillation dataset comprising 1000 (Q_i, S_i, C_i) triplets.

3.2.2. Dual-Expert LoRA Knowledge Transfer

We select Qwen3-8B [23] as the student model and apply LoRA for parameter-efficient fine-tuning. LoRA adds a low-rank update $\Delta W = BA$ to the pre-trained weight W_0:

$$W' = W_0 + \alpha \Delta W = W_0 + \alpha BA \quad (1)$$

where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are low-rank decomposition matrices, with ranks r much smaller than the original matrix dimensions. We adopt a hard-label distillation strategy, taking S_i and C_i as precise target sequences. α is the scaling factor, and by controlling α, the magnitude of the update can be adjusted.

3.3. Neuro-Symbolic Task Decoupling and Dual-Expert LoRA Design

To ensure the entire process of transforming natural language into executable code is reliable, we decouple it into two sequential, specialized subtasks handled by separate LoRA experts. This decoupling improves stage-wise accuracy and reduces logical errors typical in one-shot generation. This design choice aligns with recent findings in ensemble learning [32], which demonstrate that employing a "Dual-Expert" system—where specialized models collaborate to solve distinct sub-problems—yields superior classification and reasoning accuracy compared to monolithic approaches.

3.3.1. Stage I: Natural Language → Symbolic Constraints

$LoRA_{NL2SL}$ focuses on extracting constraints and preferences from Q_i and formatting them as a structured symbol set S_i. LoRA updates in attention layers influence query, key, value, output projections, and thereby the model's focus, identification of key tokens, transmitted information, and multi-head integration.

To clarify the transformation process, Table 1 illustrates the schema definitions and examples of the intermediate symbolic language.

Table 1. Definition and Examples of Symbolic Middleware Schema.

Component	Definition	Example
Stage I Output S_i	Structured Symbol Set	["people":3,"days":2,"cost":4000,"place":"shanghai", "preference":"minimize walking asmuch as possible"]
Stage II Output C_i	Executable Python Code	'result = (day_count(plan) = = 2)', 'result = (people_count(plan) = = 3)', 'total_cost = 4000\\nfor activity in all activities(plan):. . .

3.3.2. Stage II: Symbolic Constraints → Executable Code

$LoRA_{SL2Code}$ consumes S_i and generates executable Python code C_i. Stage II uses the same LoRA configuration but is trained independently from Stage I so that each expert can specialize without gradient interference. Both experts start from the same base model but maintain separate LoRA parameter sets, allowing for independent updates or replacement.

3.3.3. Dual-Expert Collaboration Mechanism

The LoRA parameters for the two expert models are completely independent. This design ensures that the parameter spaces for the two experts are entirely distinct and non-overlapping. Consequently, the loss from the first stage does not influence the parameter updates of the second stage, which in turn allows each expert to achieve optimal performance on its specific task. Through this two-expert LoRA design, SLTP achieves high-fidelity NL2SL conversion while keeping a lightweight deployment footprint.

3.4. Planning Paradigm: Symbolic Translation and Multi-Option Heuristic Tree Search

3.4.1. Motivation and Problem Reformulation

Existing travel planning methodologies generally bifurcate into two streams. The first category comprises LLM-based iterative reasoning methods, which rely on the model's autoregressive generation capabilities for step-by-step decision-making. Despite their flexibility, these methods essentially function as Probabilistic Local Search; they lack the

capacity for precise tracking of global constraints (such as dynamic budget depletion) and are prone to "planning hallucinations" or becoming trapped in local optima. The second category involves general neuro-symbolic methods, which attempt to utilize general-purpose solvers to process transformed mathematical models. However, when confronted with the complex heterogeneous constraints and vast POI state spaces inherent in travel planning, general solvers often face the dual challenges of search space explosion and insufficient DSL translation precision.

Furthermore, we posit that an ideal travel agent should not be limited to finding a single "mathematically optimal solution". From the perspective of cognitive decision psychology, users tend to compare and weigh multiple feasible plans that balance planning quality with stylistic differences (Balancing Quality and Diversity), rather than seeking an absolute mathematical optimum [33,34].

To address these limitations, we formalize travel planning as a Top-K Constraint Satisfaction Problem (Top-K CSP) with multiple hard constraints [35]. We propose a paradigm of "One-shot High-Fidelity Translation + Domain-Specific Heuristic Search". This paradigm abandons uncertain reasoning by LLMs during the search phase, instead utilizing the precise Python code generated in the first phase as deterministic state transition equations and pruning validators.

3.4.2. Core Algorithm: Multi-Option Heuristic Tree Search

To efficiently identify K heterogeneous feasible solutions within a vast combinatorial space, we designed the MHTS algorithm. Unlike traditional Depth-First Search (DFS), MHTS innovatively integrates budget-aware look-ahead pruning and a dynamic ranking mechanism based on semantic preferences [36]. The architectural logic and component interaction of MHTS are synthesized in Figure 2.

Figure 2 delineates the operational workflow of the algorithm. The Main Process (Top Center) orchestrates the constraint-driven trajectory, where the search expands from the query node to generate valid paths. To counteract the curse of dimensionality, the Systematic Pruning Taxonomy (Module A, Left) implements a rigorous filtration mechanism, enumerating six distinct pruning strategies, including time constraints, user requirements, and transport rules.

Concurrently, the Budget-Aware Heuristic Strategy (BAHS), visualized in Module B (Bottom), functions as a dynamic resource regulator. The system computes a Budget Urgency Ratio based on the current state, dynamically toggling between "Relaxed" and "Tight" policies to guide specific actions (e.g., prioritizing affordable dining when the budget constraint tightens). Finally, the Top-K Diversity Candidate Reservoir (Module C, Right) aggregates valid reasoning paths, ensuring the output of Top-K recommended schemes that encompass diverse stylistic attributes, such as cultural immersion versus modern exploration.

Algorithm 1 provides a formal description of the complete MHTS process. The algorithm takes the natural language query q and the target number of solutions K as inputs. It utilizes the executable symbolic code C, generated by the dual-expert LoRA, as the core constraint validator. We define C_{mand} as the estimated mandatory cost baseline (e.g., return tickets and minimum accommodation) derived from C. The search process recursively expands the itinerary state, denoted as (day, t, pos), where day represents the current date, t is the current time, and pos is the current geographical location. The algorithm maintains a set of feasible itineraries P_{found}, terminating when $\left|P_{found}\right| \geq K$ or the search space is exhausted.

Algorithm 1: Multi-Option Heuristic Tree Search (MHTS)

Input: Natural language query q, number of solutions K
Output: A set of feasible itineraries P_{found} (at most K)
Global: $P_{found} \leftarrow \emptyset$
procedure Multi-Option Heuristic Tree Search (q, K)
  $C \leftarrow$ NL2DSL (q)
  $C_{mand} \leftarrow$ EstimateMandatoryCosts (C)
  foreach ($trans_{go}$, $trans_{back}$, $hotel$) $\in$ CandidateCombinations (C) **do**
    $plan \leftarrow$ InitializePlan ($trans_{go}$, $trans_{back}$, $hotel$)
    DFS ($plan$, 0, t_0, pos_0, C, C_{mand}, K)
    if $|P_{found}| \geq K$ **then**
      return P_{found}
    end if
  end for
  return P_{found}
end procedure
procedure DFS ($plan$, day, t, pos, C, $Cmand$, K)
  if $|P_{found}| \geq K$ **then**
    return True
  end if
  if IsComplete ($plan$, day, C) **then**
    if ValidateConstraints ($plan$, C) **then**
      $Pfound \leftarrow Pfound \cup \{plan\}$
      return True
    end if
    return False
  end if
  $\tau_{list} \leftarrow$ TimeDrivenTypes (day, t, C)
  foreach $\tau \in \tau_{list}$ **do**
    $POIs \leftarrow$ FilteredCandidates (τ, pos, $plan$, C)
    ▷*Budget-aware heuristics*
    $\rho \leftarrow$ ComputeBudgetRatio ($plan$, day, C, C_{mand})
    $POIs \leftarrow$ BudgetFilter ($POIs$, ρ, C)
    $POIs \leftarrow$ TopN ($POIs$, N)
    foreach $poi \in POIs$ **do**
      $Transports \leftarrow$ SelectTransportByBudget (pos, poi, ρ)
      foreach $trans \in Transports$ **do**
        $activity \leftarrow$ SimulateActivity (poi, $trans$, t)
        if ViolateConstraints ($activity$, $plan$, C) **then**
          continue
        end if
        $plan \leftarrow plan \cup \{activity\}$
        (day, t, pos) $\leftarrow$ UpdateState ($activity$, day, t)
        if DFS ($plan$, day, t, pos, C, C_{mand}, K) **or** $|P_{found}| \geq K$ **then**
          return True
        end if
      end for
    end for
  end for
  return False
end procedure

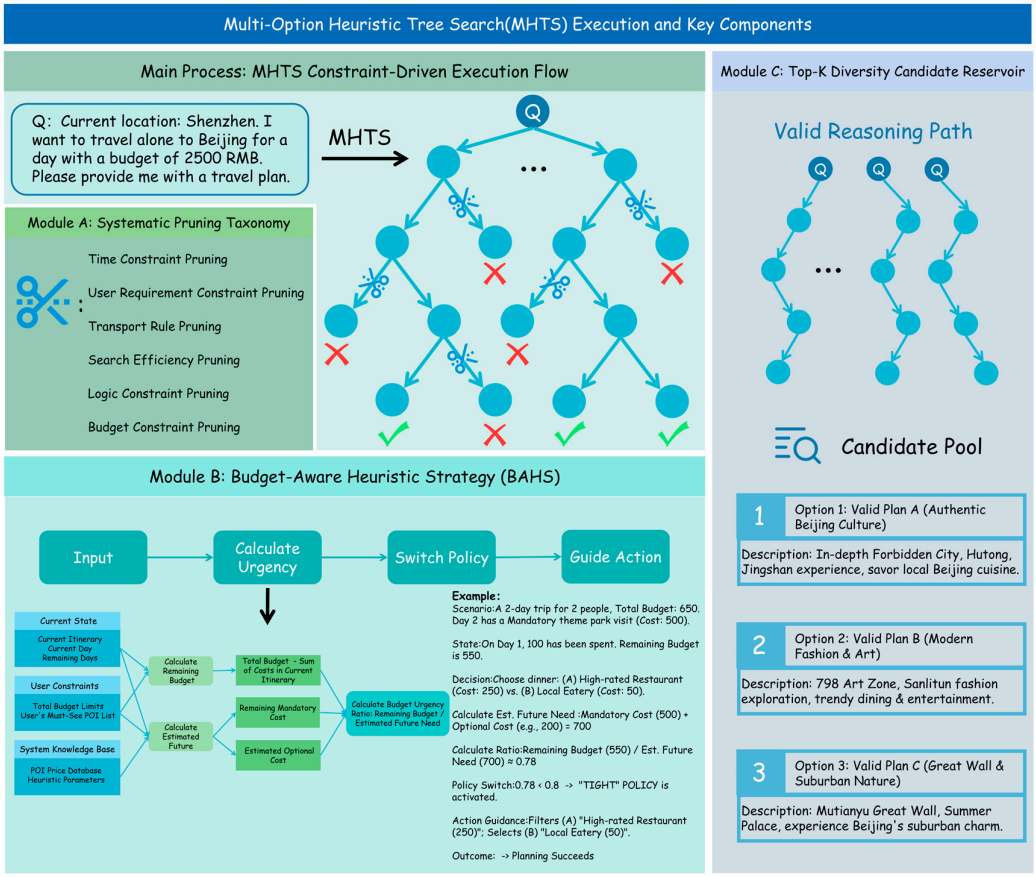


Figure 2. Execution Flow and Key Components of MHTS. The diagram visualizes the constraint-driven search execution (Main Process), supported by Systematic Pruning (Module A), Budget-Aware Heuristics (Module B), and a Top-K Candidate Reservoir (Module C).

In the proposed algorithm, specific functions facilitate the search process. EstimateMandatoryCosts (elaborated in Section 3.4.3) uses C to pre-calculate fixed expenditures, establishing a budget baseline for C_{mand}. FilteredCandidates performs an initial screening to eliminate POIs that violate physical time windows or operating hours defined in C. Crucially, the TopN (POIs, N) function balances heuristic quality and solution diversity: it ranks candidates not solely by their heuristic score but by applying a Homogeneity Penalty if a POI's category overlaps with existing itinerary items in the plan. This ensures the planner prioritizes high-quality attractions while actively suppressing category repetition. Finally, BudgetFilter dynamically prunes candidates based on the real-time budget urgency ratio ρ.

3.4.3. Key Mechanism: Budget-Aware Lifeline and Dynamic Policy

Although LLMs struggle with numerically sensitive planning, the introduced Budget-Aware Heuristic Strategy (BAHS) constitutes the "lifeline" of the search process. We define the budget urgency coefficient ρ_t as the ratio of the current remaining budget to the estimated future minimum cost:

$$\rho_t = \frac{B_{total} - Costcurrent}{EstCostmandatory + EstCostdaily \times Daysremain} \quad (2)$$

The $EstCost_{daily}$ is calculated as the average daily expenditure of the user's historical preferences or, if unavailable, the average cost of mid-range POIs in the database. $EstCost_{mandatory}$ is the pre-calculated sum of fixed costs (e.g., return flight tickets) required to complete the trip.

The design of this formula serves not merely as a simple ratio calculation but acts as a Look-ahead Oracle.

(1) Preventing Myopia: The *EstCost* term in the denominator forces the algorithm to "reserve" survival costs for future days (e.g., accommodation, return tickets) when making decisions at the current step. This effectively resolves the classic defect of greedy algorithms, which tend to "squander early and fail late".
(2) Policy Switching: The coefficient ρ_t drives real-time adjustment of the search policy. In Relaxed Mode ($\rho_t \geq 1.0$), when the budget is sufficient, the heuristic function $H(s)$ is primarily driven by user preferences (e.g., "likes history") and POI ratings, encouraging the exploration of high-quality attractions. Conversely, in Tight Mode ($\rho_t < 0.8$), when the budget is critical, the searcher automatically switches to "Survival Mode," strictly filtering out high-cost POIs and significantly increasing the weight of low-cost transport modes like "walking/metro". This mechanism ensures that the algorithm can intelligently converge to a feasible path even under extremely constrained resources, mirroring the Anytime Repairing A* (ARA*) paradigm where search effort is dynamically allocated based on resource constraints [37].

3.4.4. Multi-Dimensional Pruning and State Compression

To address the problem of combinatorial explosion, in addition to budget awareness, we introduce a Systematic Pruning Taxonomy, ensuring that MHTS achieves millisecond-level response times in large-scale state spaces.

This taxonomy employs a tripartite strategy. First, Hard Logic Pruning directly utilizes the LoRA-generated Python code as a validator, where any branch violating *must_not_visit* or *closed_days* is immediately truncated without entering subsequent calculations. Second, Spatio-Temporal Reachability Pruning calculates the estimated arrival time $t_{arrival} = t_{current} + t_{travel}(u, v)$ when expanding node v. If $t_{arrival} > t_{close}(v)$ or $t_{arrival} + t_{return_home} > t_{limit}$, the branch is deemed infeasible, this pruning based on physical time drastically compresses the invalid search depth. Finally, Homogeneity Deduplication corresponds to the diversity objective. To avoid generating repetitive experiences (e.g., recommending three parks in one day), we maintain a "category exclusion set" during path construction. If the current path already includes high-frequency categories, the heuristic score for subsequent nodes of the same category will be severely penalized. This serves not only as a pruning strategy but also as the underlying guarantee for the diversity of Top-K results, similar to diversity-based ensemble pruning methods [38].

Theoretically, the worst-case time complexity of the travel planning search is $O\left(b^d\right)$, where b is the average branching factor (available POIs or transport options) and d is the planning horizon (days). However, the proposed systematic pruning taxonomy significantly reduces the effective branching factor such that $b_{eff} \ll b$. Specifically, Spatio-Temporal Pruning eliminates the vast majority of invalid node expansions based on physical time constraints, while Budget-Aware Pruning dynamically contracts the search space when resources become scarce. The empirical results demonstrate that these mechanisms enable MHTS to converge within milliseconds, effectively mitigating the combinatorial explosion inherent in multi-day planning.

3.4.5. Section Summary: Dynamic Search Closed-Loop

In summary, the SLTP planning module establishes a dynamic "Translation-Lookahead-Search" closed loop. The LoRA model first solidifies natural language into deterministic symbolic constraints; subsequently, the BAHS mechanism acts as a "navigator," using the ρ value to perceive resource consumption and adjust the course in real-time; simultaneously, multi-dimensional pruning strategies eliminate invalid spaces, and the Top-K mechanism

ensures the collection of solutions that balance planning quality and diversity. This design achieves an organic unification of Correctness (constraint satisfaction) and Diversity (recommendation variety).

4. Experiments and Results

We evaluate SLTP on complex neuro-symbolic travel-planning tasks. We describe datasets, metrics, baselines, training and search configurations, then compare SLTP against state-of-the-art baselines on ChinaTravel [39], with ablation studies for the two-stage LoRA distillation and MHTS.

4.1. Dataset

The dataset is based on an open travel-planning benchmark with real Chinese multi-day multi-POI queries designed to stress combinatorial constraint handling.

4.1.1. Dataset Composition and Distillation

The training set comprises 1000 open-scenario queries with diverse, untemplated formats reflecting real-world user intents. To resolve the shortage of high-quality intermediate data, we apply teacher–student distillation: DeepSeek V3.2 [18] (teacher) generates structured symbolic constraint representation, and executable Python planning code, which are manually verified and used as hard labels to drive two-stage LoRA fine-tuning on Qwen 3-8B (student) [23].

The test set uses Val154, containing 154 manually collected real queries with higher complexity and varied constraint combinations to test generalization and robustness.

4.1.2. External Ticketing and POI Information

ChinaTravel provides a sandbox knowledge base used by SLTP:

POI data: 12,000 POIs (3413 attractions, 4655 restaurants, 4124 hotels) with coordinates, opening hours, price, type, etc.

Transport data: Intercity: 720 flights and 5770 train listings with IDs, departure/arrival cities, times, durations, fares; Intracity: APIs provide subway, taxi, and walking routes with real-time price, distance, and duration.

4.2. Baseline Models

To conduct a comprehensive performance comparison, we selected three representative planning paradigms in the field as baseline models: pure LLM agent methods, verifier-based iterative correction methods, and traditional neuro-symbolic planning methods.

4.2.1. Pure LLM Agent Methods

These methods rely entirely on the autoregressive generation capabilities of large language models to perform planning, using chain-of-thought [40] reasoning or tool calls.

ReAct Agent [10]: The ReAct framework decouples reasoning (Thought) and acting (Action), interacting with the ChinaTravel sandbox's API tools via an iterative "think-act-observe" loop. The agent builds multi-day itineraries incrementally using internal knowledge and tool feedback. Due to the inherent brittleness of LLM logical reasoning, ReAct is prone to producing logically contradictory or geographically infeasible "planning hallucinations" when handling multiple hard constraints, resulting in low constraint-satisfaction rates. In subsequent experimental setups, zero-shot provides no task examples, whereas one-shot supplies one (or a few) examples to guide the model, aiming to improve its task-specific performance and instruction compliance.

Act-only Agent: As an ablation variant of ReAct, this model skips intermediate chain-of-thought generation and directly emits actions. Such models lack an explicit reasoning

trace, but their performance remains an important reference point for evaluating the base planning capability of LLMs.

4.2.2. Verifier-Based Iterative Correction Methods

These methods attempt to compensate for LLM planning's logical shortcomings using feedback from an external symbolic verifier; they are an enhancement of the LLM agent architecture.

LLM-Modulo-Verifier [26,27]: This approach is similar to the Reflexion framework but focuses on symbolic verification. The agent first generates a complete or partial plan, which is then checked by a symbolic verifier for constraint satisfaction. To ensure reproducibility, we implemented this verifier as a deterministic Python-based module that rigorously validates hard constraints such as budget limits and time windows. The verifier returns precise error feedback to the LLM, which performs multiple rounds of self-correction. This method works for simple planning, but for complex multi-day itineraries, the LLM-Modulo's error-correction capability rapidly degrades after 3–5 iterations and tends to introduce new "emergent errors" while fixing old ones, causing unstable performance on macro-level constraint-satisfaction metrics.

4.2.3. Neuro-Symbolic Planning Methods

These methods are the mainstream paradigm for solving complex constrained planning tasks, aiming to combine LLM language understanding with the rigor of symbolic logical planning.

This solution divides the planning pipeline into two steps: NL2DSL translation (the LLM converts the query into DSL logical constraints S) and interactive search. Specifically, for the NL2DSL step, we employ a standard few-shot prompting strategy that leverages the base model's in-context learning capabilities to generate constraints without task-specific fine-tuning. Subsequently, a DFS algorithm builds itineraries using POI orders recommended by the LLM, while a symbolic verifier checks constraints in real time and triggers backtracking. The baseline's performance heavily depends on the accuracy of the NL2DSL translation, and the symbolic search time is strictly limited to 5 min, which constrains its ability to explore optimal solutions in complex scenarios.

TTG (To The Globe) [29]: TTG converts natural language requirements into a mixed-integer linear programming (MILP) model and uses professional optimization solvers to find optimal solutions. This approach guarantees mathematical optimality. However, given the complexity of ChinaTravel's constraints and the POI scale, the MILP model size grows exponentially. Even with POI sampling and time-step discretization, the number of constraints can still reach hundreds of thousands, resulting in extremely high solver computational costs and making it practically infeasible for real-world use.

4.3. Evaluation Metrics

We adopt strict ChinaTravel [39] metrics covering environmental feasibility, logical rigor, and planning fidelity. Metrics exist at micro and macro levels.

Delivery Rate (DR): the ability to produce a correctly formatted solution within a timeout.

Environmental Pass Rate (EPR):

Micro EPR (EPR_{micro}): average proportion of environmental constraints satisfied across plans:

$$EPR_{micro} = \frac{\sum_{p \in P} \sum_{c \in Env} \mathbb{1}_{passed(c,p)}}{|P| \cdot |Env|} \quad (3)$$

Macro EPR (EPR_{macro}): proportion of plans that satisfy all environmental constraints:

$$EPR_{macro} = \frac{\sum_{p \in P} \prod_{c \in Env} \mathbb{1}_{passed(c,p)}}{|P|} \tag{4}$$

where P is the set of all plans. $\mathbb{1}_{passed(c,p)}$ is an indicator function, which is 1 if the plan p satisfies the environmental constraint c, and 0 otherwise. $|Env|$ is the number of constraints in the environmental constraint set Env.

Logical Pass Rate (LPR):

Micro LPR (LPR_{micro}):

$$LPR_{micro} = \frac{\sum_{p \in P} \sum_{c \in C_p} \mathbb{1}_{passed(c,p)}}{\sum_{p \in P} |C_p|} \tag{5}$$

Macro LPR (LPR_{macro}):

$$LPR_{macro} = \frac{\sum_{p \in P} \prod_{c \in C_p} \mathbb{1}_{passed(c,p)}}{|P|} \tag{6}$$

where P is the set of all plans. p is a single plan in the set P. $\mathbb{1}_{\mathtt{passed}(\mathtt{c},\mathtt{p})}$ is an indicator function, which is 1 if the plan p satisfies the logical constraint c, and 0 otherwise. $\prod c \in C_p \mathbb{1}_{passed(c,p)}$ means it is 1 when the plan p satisfies all logical constraints $c \in C_p$, and 0 otherwise.

Conditional Logical Pass Rate (C-LPR): average logical constraints satisfied, conditioned on satisfying all environmental constraints:

$$C - LPR = \frac{\sum_{p \in P} \mathbb{1}_{passed(Env,p)} \cdot \sum_{c \in C_p} \mathbb{1}_{passed(c,p)}}{\sum_{p \in P} \left|C_p\right|} \tag{7}$$

Final Pass Rate (FPR): proportion of plans satisfying all environmental and all logical constraints:

$$FPR = \frac{\sum_{p \in P} \mathbb{1}_{passed(Env,p)} \cdot \mathbb{1}_{passed(C_p,p)}}{|P|} \tag{8}$$

Overall Score: a weighted aggregate to reflect the comprehensive quality of generated itineraries, primarily utilized to evaluate the quality decay and diversity trade-offs in Top-K Strategy experiments:

$$\begin{aligned} OverallScore = {} & 0.05 \cdot DR + 0.1 \cdot EPR_{\text{micro}} + 0.1 \cdot EPR_{\text{macro}} + 0.15 \cdot LPR_{\text{micro}} \\ & + 0.15 \cdot LPR_{\text{macro}} + 0.15 \cdot C - LPR + 0.3 \cdot FPR \end{aligned} \tag{9}$$

4.4. Training Parameters and Experimental Setup

The SLTP framework underwent two-stage LoRA fine-tuning based on the Qwen 3-8B [23] model. To ensure the reproducibility and resource efficiency of the experiments, we utilized a computing cluster and strictly controlled the training configurations.

4.4.1. Hardware and Software

In this study, all experiments were conducted on a compute cluster equipped with NVIDIA A800-40GB GPUs. We allocated two GPUs per cluster; for LoRA stages, a single A800-40GB GPU was used per stage. The host CPU is an Intel Xeon Platinum 8378A with 128 cores. The software environment comprised Ubuntu 22.04.5 LTS, Python 3.9.23, and PyTorch 2.6.0. Training was performed using bfloat16 (bf16) mixed precision.

4.4.2. LoRA Training Hyperparameters

Both LoRA experts share training setups: LoRA rank r = 8, scaling factor α = 16, dropout = 0.05. We fine-tune only attention projection layers (q_proj, k_proj, v_proj, o_proj). Training epochs = 3, initial learning rate = 2×10^{-5}, optimizer = AdamW. Per-device batch size = 2 with gradient accumulation of 8 steps (effective batch size 16). Maximum sequence length = 4096.

We set the LoRA rank $r = 8$ and scaling factor $\alpha = 16$. Empirical studies in reasoning tasks [24] suggest that this configuration provides an optimal balance between parameter efficiency and the model's plasticity to adapt to the rigorous syntax of our symbolic DSL. We fine-tuned all attention projection layers (q_proj, k_proj, v_proj, o_proj) to maximize the model's capability in capturing complex logic dependencies.

4.5. Heuristic Design and Parameter Configuration

To ensure the reproducibility of the experiment and to verify the robustness of the MHTS algorithm across different modes, this section details the configuration logic for key parameters.

4.5.1. Search Mode Settings

Our framework supports two operational modes, corresponding to different application scenarios. The Benchmark Evaluation Mode ($K = 1$) is configured to conduct a fair comparison with baseline models (ReAct, NeSyPlan, LLM-Modulo), as detailed in Sections 4.6.1 and 4.6.2. In this setting, the algorithm degrades to a fastest-mode search for a single feasible solution. In contrast, the Enhanced Application Mode ($K = 6$) is utilized for the supplementary experiments in Section 4.6.3. This mode is explicitly designed to evaluate the system's performance boundaries and time efficiency when generating diverse schemes.

4.5.2. Budget-Aware Thresholds

We set the policy switching threshold at $\rho_{threshold} = 0.8$. This value is not chosen arbitrarily but is based on empirical observations of travel consumption distributions—dining and transport costs often exhibit a Long-tail Distribution, making unexpected high expenditures likely. Therefore, the system retains a 20% budget Safety Margin to cope with potential consumption fluctuations, a concept aligning with Robust Planning under Uncertainty in AI. Experiments demonstrate that this non-linear conservative strategy significantly reduces Deep Backtracking caused by late-stage budget depletion.

4.5.3. Heuristic Scoring Weights: Hierarchical Prioritization

In the *HeuristicRank* function, the score of a candidate node is determined by the following equation:

$$Score(p) = w_1 \cdot \mathbb{I}(MustVisit) + w_2 \cdot Rating(p) - w_3 \cdot Cost(p) - w_4 \cdot Dist(p) \quad (10)$$

In our experiments, the heuristic weights are configured to strictly enforce Hierarchical Prioritization [41]. Specifically, we set $w_1 = 1000$, while the soft preference weights w_2 and w_3 are adjusted within the range of $[100, 200]$. This order-of-magnitude difference ($w_1 \gg w_2$) serves to simulate Lexicographical Optimization, ensuring that hard constraints (e.g., Must-Visit) strictly dominate soft preferences. Sensitivity analysis in our preliminary experiments indicated that the algorithm's performance is robust regarding the specific values, provided the hierarchical ratio ($w_1 > 10 \times w_{soft}$) is maintained.

4.6. Experimental Results and Analysis

4.6.1. Comparison with Baselines

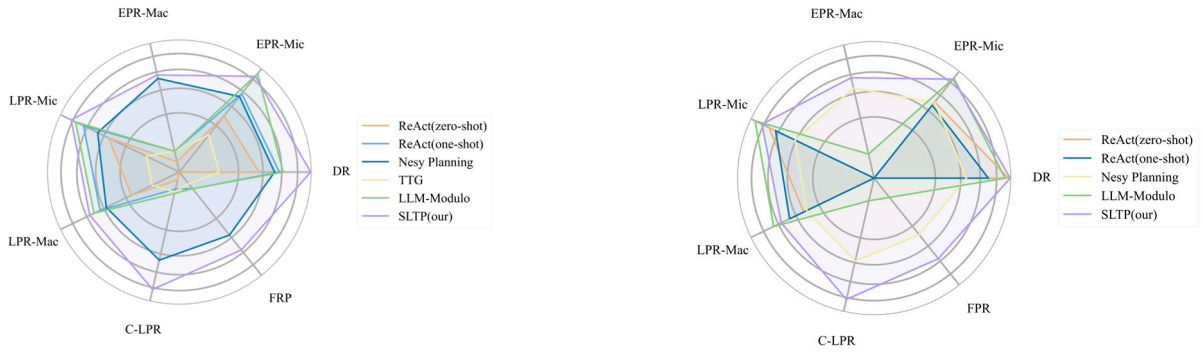
We compare SLTP against baselines on the ChinaTravel test set. Tables 2 and 3 show comparisons: Table 2 uses teacher-backbone methods (DeepSeek-V3 [18] as base), Table 3 uses GPT-4o [17] as reference backbone; SLTP's base model is Qwen3-8B [23]. Radar plots in Figure 3 illustrate per-metric strengths.

Table 2. Performance comparison of teacher-model-based methods. Bold indicates the best result, and underline indicates the second best.

	DR	EPR		LPR		C-LPR	FPR
		Mic	Mac	Mic	Mac		
ReAct (zero-shot)	36.4	29.5	0.65	35.2	16.2	0.38	0.00
ReAct (one-shot)	57.3	57.3	2.60	64.6	44.2	1.71	2.60
Nesy Planning	51.9	53.2	<u>52.5</u>	47.0	37.6	<u>46.5</u>	<u>37.0</u>
TTG	9.09	12.8	2.59	7.65	5.19	2.39	1.29
LLM-Modulo	<u>61.6</u>	<u>90.2</u>	2.59	<u>75.9</u>	<u>51.2</u>	2.75	2.59
SLTP (our)	**98.7**	**96.6**	**95.6**	**85.4**	**56.5**	**82.5**	**56.5**

Table 3. Performance comparison of reference-model-based methods. Bold indicates the best result, and underline indicates the second best.

	DR	EPR		LPR		C-LPR	FPR
		Mic	Mac	Mic	Mac		
ReAct (zero-shot)	<u>96.1</u>	50.5	0.00	72.4	32.5	0.00	0.00
ReAct (one-shot)	69.5	46.3	0.00	63.6	46.8	0.00	0.00
Nesy Planning	45.4	50.1	<u>45.4</u>	40.9	29.8	<u>38.5</u>	<u>27.9</u>
LLM-Modulo	91.5	<u>87.2</u>	3.24	**92.9**	**66.2**	2.87	3.24
SLTP (our)	**98.7**	**96.6**	**95.6**	<u>85.4</u>	<u>56.5</u>	**82.5**	**56.5**



(**a**) (**b**)

Figure 3. (**a**) Comparison of metrics for teacher-model-based methods; (**b**) Comparison of metrics for reference-model-based methods.

SLTP framework's outstanding constraint satisfaction: As shown in Tables 2 and 3, our proposed SLTP framework demonstrates outstanding performance across all evaluation metrics. In particular, on core metrics that measure planning fidelity and constraint satisfaction—such as Conditional Logic Pass Rate (C-LPR) and Final Pass Rate (FPR)—SLTP reaches 82.5 and 56.5, respectively, significantly outperforming all baseline methods. This strongly demonstrates SLTP's powerful ability to generate logically rigorous travel plans that highly conform to user intentions and environmental constraints.

Limitations of pure LLM agent methods: ReAct (zero-shot) and ReAct (one-shot), as representative pure LLM agent methods, perform poorly. In Table 2, ReAct (zero-shot) exhibits extremely low constraint satisfaction rates, with a C-LPR of 0.38, while ReAct (one-shot) only slightly improves to 1.71. Even though ReAct (zero-shot) shows some improvement in Table 3, its macro metrics (such as Mac EPR, C-LPR, and FPR) remain very low, even zero. This confirms the assertion made in the introduction that pure LLM agents are prone to "planning hallucinations" and logical fragility when handling complex, multi-constraint travel planning, making it difficult for them to meet strict constraint requirements.

Challenges of validator-driven iterative correction methods: The LLM-Modulo approach attempts iterative correction via external validator feedback and shows strong performance on micro metrics (Mic EPR, Mic LPR)—for example, in Table 2 Mic EPR is 90.2, and Mic LPR is 75.9; in Table 3 Mic EPR is 87.2, and Mic LPR is 92.9. However, its macro metrics (e.g., Mac EPR, C-LPR, and FPR) are significantly low (in Table 2, FPR is 2.59 and C-LPR is 2.75; in Table 3, FPR is 3.24 and C-LPR is 2.87). This confirms that LLM-Modulo's error-correction ability can degrade in complex multi-day itineraries, and it is prone to introducing sporadic ("burst") errors, resulting in unstable overall performance and difficulty achieving high-fidelity planning.

Effectiveness of the neuro-symbolic planning paradigm and TTG's computational bottleneck: Nesy Planning, as a neuro-symbolic planning method, achieves a C-LPR of 46.50 in Table 2 and 38.50 in Table 3, outperforming pure LLM agents but still showing a substantial gap compared to SLTP. This indicates that the neuro-symbolic paradigm is the right direction for solving complex planning problems, and that SLTP further improves the paradigm's efficiency and accuracy via its novel two-stage LoRA fine-tuning and MHTS.

TTG shows a negligible C-LPR of 2.39 in Table 2. This strongly supports our analysis that TTG transforms complex tasks into mixed-integer linear programming (MILP) models, which, although theoretically able to guarantee optimal solutions, face exponentially growing computational costs in complex scenarios with many constraints and POIs such as the ChinaTravel benchmark, making them nearly infeasible under practical time limits.

In summary, the SLTP framework, owing to its innovative architectural design, effectively overcomes the logical shortcomings of traditional LLM agents and the efficiency bottlenecks of existing neuro-symbolic methods, achieving industry-leading performance on complex travel planning tasks. In particular, SLTP shows significant advantages in ensuring planning logical rigor, satisfying multiple constraints, and providing high-fidelity solutions.

4.6.2. Ablation Studies

To quantify the contribution of SLTP's core components—the two-stage LoRA and the heuristic tree search—to the final performance, we conducted ablation experiments, as shown in Table 4.

Table 4. Ablation study of the SLTP framework. Bold indicates the best result, and underline indicates the second best.

	DR	EPR		LPR		C-LPR	FPR
		Mic	Mac	Mic	Mac		
SLTP (our)	**98.7**	**96.6**	**95.6**	**85.4**	**56.5**	**82.5**	**56.5**
w/o Heuristic	95.5	<u>95.4</u>	<u>94.8</u>	82.0	52.6	<u>81.4</u>	<u>52.6</u>
w/o Lora	<u>96.1</u>	93.0	91.6	<u>83.1</u>	<u>54.5</u>	79.5	<u>52.6</u>
w/o Dual-Expert	95.8	95.3	91.6	81.2	51.2	81.1	51.2

To further validate the necessity of the decoupled two-stage architecture, we introduced the w/o Dual-Expert variant. In this setting, the student model is fine-tuned as a single-stage end-to-end model to directly map natural language queries to executable code, removing the intermediate symbolic layer. As shown in Table 4, the w/o Dual-Expert variant achieves an FPR of 51.2%, which is 5.3% lower than the full SLTP framework. This drop in performance is attributed to the difficulty for a lightweight 8B model to simultaneously learn semantic extraction and strict syntactic code generation, justifying our sequential dual-expert design.

Significant gains from MHTS: We investigated the effectiveness of the heuristic search. Comparing the full SLTP framework (FPR: 56.5) with w/o Heuristic (i.e., removing the heuristic module from MHTS; FPR: 52.6), we found that, under the same LoRA conversion capability, using heuristic tree search rather than a non-heuristic tree search increases the FPR by 3.9 percentage points. In addition, compared to w/o Heuristic, MHTS yields consistent improvements on other key metrics such as Mic EPR, Mac EPR, Mic LPR, Mac LPR, and C-LPR. This indicates that the budget-aware heuristics and multi-level pruning strategy proposed in this work can navigate complex planning spaces more effectively and find higher-quality solutions.

Key role of LoRA and knowledge distillation: We evaluated the roles of LoRA fine-tuning and knowledge distillation in the entire process of transforming natural language into executable code conversion (NL2Code). Comparing the full SLTP framework (FPR: 56.5) with w/o LoRA (i.e., removing the LoRA module and using the base model for NL2Code conversion while retaining the heuristic tree search; FPR: 52.6), we observe that, when the planner uses the same efficient heuristic tree search (MHTS), the LoRA NL2Code conversion module—optimized via two-stage LoRA fine-tuning and knowledge distillation—improves the FPR by 3.9 percentage points. This shows that the LoRA component, by producing accurate natural-language-to-symbolic-constraint and symbolic-constraint-to-executable-code conversions, provides the planner with more reliable and comprehensive hard constraints, thereby improving overall planning quality.

Synergistic effect of core components: The full SLTP framework achieves the highest FPR of 56.5 across all configurations. This demonstrates a strong synergy between the high-precision NL2Code (natural language to executable code) conversion and the efficient MHTS in the planning stage. High-quality symbolic constraints and executable code serve as reliable inputs and verification mechanisms for the planner, allowing MHTS to explore the feasible solution space more accurately and efficiently, which substantially enhances planning fidelity and efficiency.

4.6.3. Efficiency Analysis: Latency and Token Consumption

In addition to success rates, operational efficiency is critical for real-world deployment. Table 5 compares the average inference latency (in seconds) and total token consumption (input + output) per query on the ChinaTravel validation set.

Table 5. Efficiency comparison between SLTP and Baselines. Bold indicates the best result, and underline indicates the second best.

	Avg. Latency (s)	**Avg. Total Tokens**
ReAct (zero-shot)	1937.8	454,133.2
ReAct (one-shot)	1141.58	890,942.0
Nesy Planning	<u>667.4</u>	<u>39,576.8</u>
LLM-Modulo	1187.6	554,395.4
SLTP (our)	**173.3**	**2561.7**

As observed, SLTP reduces inference latency by roughly 6.6 times compared to ReAct (One-shot) and 6.9 times compared to LLM-Modulo. This efficiency stems from SLTP's 'Generate-once' paradigm, whereas baselines require multiple rounds of LLM inference for reasoning or error correction. Furthermore, token consumption is significantly lower, translating to a direct reduction in deployment costs."

4.6.4. Top-K Strategy Efficacy Analysis: Trade-Off Between Quality and Efficiency

To validate the potential of the SLTP framework in real-world interactive scenarios, we conducted extended experiments on the Human-Val dataset with $K = 6$. Figure 4 provides a quantitative performance profiling of the Top-K strategy.

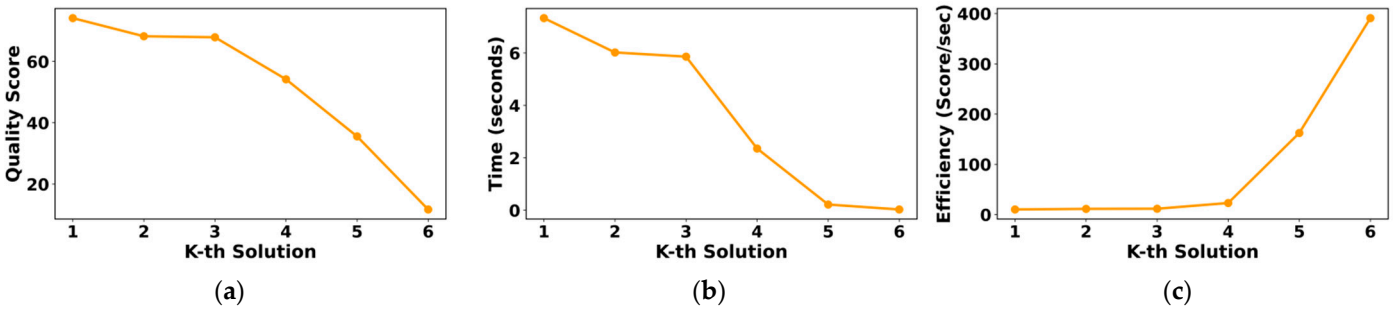


Figure 4. Performance Analysis in Top-K ($K = 6$) Mode. (**a**) The decay in the solution quality score. (**b**) The reduction in marginal search time. (**c**) The increase in system throughput efficiency as K increases.

Solution Quality: Deep Mining of Diverse Solution Spaces: To evaluate the comprehensive quality of the top-K recommendations, we utilize the weighted Overall Score defined in Equation (9). As shown in Figure 4 (Left), the first generated plan (K = 1) typically represents the global optimal solution under current constraints. As the number of K increases, solution quality shows a gentle downward trend; however, even the 3rd solution maintains a score above 80% of the primary choice. This suggests that selecting $K = 3$ is an ideal choice for users. Furthermore, it indicates that the MHTS algorithm does not simply perform random filling. Instead, benefiting from the Homogeneity Deduplication mechanism introduced in Section 3.4.4, the algorithm is forced to utilize heuristic scoring to mine Diverse Regions of the Solution Space within the search tree. Although these schemes may have slightly lower mathematical scores, they often represent different trade-off orientations (e.g., a route that is more physically demanding but includes an additional attraction), effectively satisfying diverse user preferences and achieving an ideal balance between planning utility and recommendation diversity [42,43].

Search Time: Negligible Marginal Cost Advantages, as shown in Figure 4 (Center), uncover a critical engineering insight: the search latency for finding the first solution is the longest, encompassing the "cold start" costs of constructing the search tree and pre-calculating distance matrices. However, the Marginal Time Cost for finding the 2nd to 6th solutions shows a Drastic Reduction, becoming negligible. This occurs because MHTS leverages a Search Tree Reuse mechanism—diverse solutions naturally exist within different branches of the search tree [44]. Once the main trunk is established, acquiring additional solutions requires only extremely low-cost shallow Backtracking.

Efficiency Analysis: Qualitative Change in Interactive Experience: The "Quality-Time Efficiency Ratio" (Cumulative Score/Cumulative Time) defined in Figure 4 (Right) confirms an exponential rise in the K value. This result powerfully demonstrates that the SLTP framework possesses the capability of "Calculate Once, Serve Multiple". Compared to methods like ReAct, which require a complete reasoning loop for every new plan generated, our Top-K strategy can provide users with a rich decision space at an extremely

low Amortized Cost, greatly enhancing the human–computer interaction experience in instant response systems.

4.6.5. Case Study

To intuitively demonstrate the advantages of SLTP, we analyzed a representative budget-constrained scenario where a user requested a 5-day trip with a very tight budget.

ReAct's Failure Mode: The ReAct Agent selected high-end hotels and taxi services during the first two days, causing budget depletion by Day 3. Although it attempted to "correct" the plan by lowering subsequent costs, the early overspending was irreversible, leading the agent into a "Correction Loop" where it failed to generate a valid, complete itinerary.

SLTP's Success Mode: In contrast, SLTP's MHTS planner calculated a Budget Urgency Ratio $\rho < 0.8$ on Day 1, immediately triggering "Survival Mode". The algorithm automatically downgraded accommodation standards and prioritized subway transport and free attractions. Ultimately, SLTP generated a complete itinerary that not only satisfied all hard constraints but also maximized the travel experience within the budget limit. This confirms the effectiveness of the Budget-Aware Heuristic Strategy (BAHS) in handling long-horizon planning tasks.

4.6.6. Summary

The SLTP framework, by means of an innovative dual-expert LoRA knowledge-distillation strategy, successfully and efficiently transfers the neuro-symbolic planning capabilities of large models to lightweight models, achieving high-fidelity NL2Code (natural language to executable code) conversion. Combined with a symbol-driven MHTS, SLTP demonstrates outstanding constraint-satisfaction rates and planning fidelity on the China-Travel benchmark. In particular, when handling complex and open-ended human queries, its robustness and efficiency greatly surpass competing models. These experimental results strongly support the effectiveness and efficiency of our proposed integrated architecture for addressing complex neuro-symbolic planning problems.

5. Discussion

5.1. Positioning SLTP Within the Planning Landscape

To contextualize the contributions of SLTP, it is essential to distinguish its design philosophy from existing paradigms, specifically optimization-based solvers, world-model reasoning, and standard neuro-symbolic pipelines.

Comparison with Optimization-based Methods. Approaches like the TTG [29] model travel planning as a Mixed-Integer Linear Programming (MILP) problem. While this theoretically guarantees global mathematical optimality, it incurs exponential computational complexity relative to the number of constraints. For complex, multi-day scenarios, the solver often requires minutes to converge, rendering it impractical for real-time applications. In contrast, SLTP prioritizes "satisficing" solutions over absolute optimality. By leveraging the MHTS with budget-aware heuristics, our framework reduces inference latency to the second level, achieving a critical balance between response speed and solution quality that is more aligned with user interaction requirements.

Comparison with World-Model-based Reasoning. Frameworks such as RAP [30] employ extensive Monte Carlo Tree Search (MCTS) driven by large-scale LLMs (e.g., LLaMA-33B) to simulate planning outcomes. Although this yields high reasoning fidelity, the computational cost of iteratively querying a large teacher model is prohibitive for widespread deployment. SLTP addresses this by distilling the reasoning traces into static, executable Python code and utilizing a lightweight 8B student model. This "Generate-

Once, Search-Cheaply" paradigm significantly lowers the deployment cost and token consumption while maintaining logical rigor comparable to larger models.

Comparison with Standard Neuro-Symbolic Pipelines. Traditional neuro-symbolic methods typically rely on a one-shot translation from natural language to Domain-Specific Language (DSL). As demonstrated in our ablation study, this single-step conversion is often brittle when facing intricate constraints, leading to syntax errors that stall downstream solvers. SLTP advances this paradigm through its decoupled two-stage LoRA architecture, which isolates semantic understanding from syntactic generation. This separation ensures that the symbolic solver receives highly accurate constraints, thereby resolving the bottleneck of translation fidelity that limits standard neuro-symbolic planners.

5.2. Limitations and Future Directions

It is important to acknowledge the operational boundaries of the proposed framework, particularly regarding generalization, data dependency, and scalability.

Dataset Specificity and Generalization. While SLTP demonstrates state-of-the-art performance on the ChinaTravel benchmark, we acknowledge that the current evaluation is confined to this specific ecosystem. The LoRA experts are fine-tuned on domain-specific logic, which may not directly transfer to other benchmarks (e.g., TravelPlanner [14]) without re-training. However, the core architecture—decoupling semantic parsing from heuristic search—is methodology-agnostic. Specifically, the SLTP framework holds significant potential for adjacent fields such as Logistics Optimization and Personal Daily Scheduling. Applying SLTP to these domains would require two key adaptations: (1) constructing a domain-specific symbolic knowledge base (e.g., a logistics transport graph or a personal calendar API), and (2) curating a new instruction-tuning dataset to map user commands (e.g., "deliver closest package first") to the specific constraints of that domain.

Dependency on Knowledge Base Integrity. As a neuro-symbolic system, SLTP strictly adheres to the data provided by the external environment. Unlike end-to-end LLMs that might rely on parametric knowledge to hallucinate "common sense" corrections for missing data, SLTP follows a "Garbage-In, Garbage-Out" principle. If the underlying POI database contains inaccuracies—such as outdated ticket prices or missing closure dates—the generated itinerary, while logically valid relative to the code, will be practically infeasible. This highlights the need for robust data verification mechanisms in real-world deployments.

Scalability in Ultra-Large Action Spaces. The current MHTS algorithm is optimized for city-level or regional planning (handling approximately 10^4 POIs). In scenarios requiring nationwide or global-scale planning (millions of POIs), the branching factor for transport nodes would expand drastically, potentially degrading search efficiency. Future iterations could address this by incorporating a Hierarchical Task Network (HTN), where coarse-grained city-to-city planning precedes fine-grained POI scheduling, thereby maintaining efficiency across varying scales.

6. Conclusions

To address the challenges faced by large language models (LLMs) in complex travel planning—such as planning hallucinations, logical brittleness, and high deployment costs—this study proposes the SLTP framework, a lightweight, high-performance planning agent that integrates neuro-symbolic methods, parameter-efficient learning, and diversified heuristic search.

The core contributions of SLTP are: By introducing a novel two-stage LoRA fine-tuning strategy, the framework decouples the complex NL2Code (natural language to executable code) task. Combined with teacher–student knowledge distillation (from DeepSeek V3.2 [18] to Qwen 3-8B [23]), SLTP achieves high-accuracy, high-fidelity NL2Code conver-

sion while substantially reducing deployment cost, enabling a lightweight model to possess strong neuro-symbolic planning capabilities. Moreover, the framework uses Python planning code generated by LoRA experts as an external symbolic verifier to drive an MHTS. This mechanism, which combines hard-constraint validation and a Top-K strategy, not only ensures the logical rigor of plans but also provides multiple high-quality, diverse itineraries that satisfy users' varied requirements.

The experimental results show that SLTP outperforms existing baselines on key metrics such as FPR and C-LPR on the ChinaTravel benchmark. Ablation studies further confirm the critical gains from task decoupling, knowledge distillation, and heuristic search for planning fidelity and efficiency.

In summary, the SLTP framework successfully fuses the semantic understanding of LLMs with the logical rigor of symbolic planning, offering an efficient, scalable, and user-friendly solution for designing agents for complex tasks. It lays a solid foundation for applying travel planning and lightweight models in the neuro-symbolic planning domain. Future work will further explore dynamic integration of external information and optimization of the Top-K strategy to provide more real-time and personalized itineraries [20].

Author Contributions: Conceptualization, M.F. and Q.J.; methodology, D.T. and Q.J.; software, D.T. and Q.J.; validation, D.T.; formal analysis, X.D.; investigation, X.D.; resources, X.D.; data curation, X.D.; writing—original draft preparation, Q.J. and D.T.; writing—review and editing, J.Y. and J.Z.; visualization, Q.J. and D.T.; supervision, M.F.; project administration, M.F. and X.Z.; funding acquisition, M.F. and X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Laboratory of Language and Artificial Intelligence, Center for Linguistics and Applied Linguistics, Guangdong University of Foreign Studies, China (grant No. LAI202306), the National Social Science Fund of China (No. 23BGL239).

Data Availability Statement: The dataset for this experiment is openly available at https://huggingface.co/datasets/LAMDA-NeSy/ChinaTravel (accessed on 1 November 2025), and we used a subset of this dataset.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Popşa, R.E. Exploring the Generation Z Travel Trends and Behavior. *Stud. Bus. Econ.* **2024**, *19*, 189. [CrossRef]
2. Wang, Z.; Zhang, H.; Cao, C.; Ramachandran, S. A Cognitive Appraisal of the City Walk Travel Trend, Considering City Image, Visitor Engagement, Place Attachment, and Behavioural Intention. *Humanit. Soc. Sci. Commun.* **2025**, *12*, 1657. [CrossRef]
3. Mihalic, T. Trends in Sustainable Tourism Paradigm: Resilience and Adaptation. *Sustainability* **2024**, *16*, 7838. [CrossRef]
4. Halder, S.; Lim, K.H.; Chan, J.; Zhang, X. A Survey on Personalized Itinerary Recommendation: From Optimisation to Deep Learning. *Appl. Soft Comput.* **2024**, *152*, 111200. [CrossRef]
5. To, W.M.; Yu, B.T.W. Artificial Intelligence Research in Tourism and Hospitality Journals: Trends, Emerging Themes, and the Rise of Generative AI. *Tour. Hosp.* **2025**, *6*, 63. [CrossRef]
6. Shrestha, D.; Wenan, T.; Shrestha, D.; Rajkarnikar, N.; Jeong, S.-R. Personalized Tourist Recommender System: A Data-Driven and Machine-Learning Approach. *Computation* **2024**, *12*, 59. [CrossRef]
7. Winata, G.I.; Madotto, A.; Lin, Z.; Liu, R.; Yosinski, J.; Fung, P. Language Models are Few-shot Multilingual Learners. In Proceedings of the 1st Workshop on Multilingual Representation Learning, Punta Cana, Dominican Republic, 11 November 2021; pp. 1–15.
8. Song, S.; Yang, C.; Xu, L.; Shang, H.; Li, Z.; Chang, Y. TravelRAG: A Tourist Attraction Retrieval Framework Based on Multi-Layer Knowledge Graph. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 414. [CrossRef]
9. Aribas, E.; Daglarli, E. Transforming Personalized Travel Recommendations: Integrating Generative AI with Personality Models. *Electronics* **2024**, *13*, 4751. [CrossRef]
10. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv* **2023**. [CrossRef]
11. Shinn, N.; Cassano, F.; Berman, E.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv* **2023**. [CrossRef]

12. Bang, Y.; Ji, Z.; Schelten, A.; Hartshorn, A.; Fowler, T.; Zhang, C.; Cancedda, N.; Fung, P. HalluLens: LLM Hallucination Benchmark. *arXiv* **2025**. [CrossRef]
13. Wei, J.; Yao, Y.; Ton, J.-F.; Guo, H.; Estornell, A.; Liu, Y. Measuring and Reducing LLM Hallucination without Gold-Standard Answers. *arXiv* **2024**. [CrossRef]
14. Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; Su, Y. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. *arXiv* **2024**. [CrossRef]
15. Mao, J.; Tenenbaum, J.B.; Wu, J. Neuro-Symbolic Concepts. *arXiv* **2025**. [CrossRef]
16. Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; Zeng, A. Code as Policies: Language Model Programs for Embodied Control. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 9493–9500.
17. OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; et al. GPT-4 Technical Report. *arXiv* **2024**. [CrossRef]
18. DeepSeek-AI; Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; et al. DeepSeek-V3 Technical Report. *arXiv* **2025**. [CrossRef]
19. Gong, L.; Guo, S.; Han, X.; Lin, Y.; Lin, Y.; Liu, Y.; Lu, Y.; Wan, H.; Zhang, X. Mobility-LLM: Learning Visiting Intentions and Travel Preference from Human Mobility Data with Large Language Models. In *Advances in Neural Information Processing Systems 37*; Neural Information Processing Systems Foundation, Inc. (NeurIPS): Vancouver, BC, Canada, 2024; pp. 36185–36217.
20. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**. [CrossRef]
21. Hsieh, C.-Y.; Li, C.-L.; Yeh, C.; Nakhost, H.; Fujii, Y.; Ratner, A.; Krishna, R.; Lee, C.-Y.; Pfister, T. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*; Association for Computational Linguistics: Toronto, ON, Canada, 2023; pp. 8003–8017.
22. Zhao, J.; Zhao, W.; Drozdov, A.; Rozonoyer, B.; Sultan, M.A.; Lee, J.-Y.; Iyyer, M.; McCallum, A. Multistage Collaborative Knowledge Distillation from a Large Language Model for Semi-Supervised Sequence Generation. In Proceedings of the Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Bangkok, Thailand, 11–16 August 2024; pp. 14201–14214.
23. Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. Qwen3 Technical Report. *arXiv* **2025**. [CrossRef]
24. Mitra, A.; Corro, L.D.; Mahajan, S.; Codas, A.; Simoes, C.; Agarwal, S.; Chen, X.; Razdaibiedina, A.; Jones, E.; Aggarwal, K.; et al. Orca 2: Teaching Small Language Models How to Reason. *arXiv* **2023**. [CrossRef]
25. Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *arXiv* **2023**. [CrossRef]
26. Gundawar, A.; Verma, M.; Guan, L.; Valmeekam, K.; Bhambri, S.; Kambhampati, S. Robust Planning with LLM-Modulo Framework: Case Study in Travel Planning. *arXiv* **2024**. [CrossRef]
27. Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L.; Murthy, A. LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. *arXiv* **2024**. [CrossRef]
28. Huang, J.; Chen, X.; Mishra, S.; Zheng, H.S.; Yu, A.W.; Song, X.; Zhou, D. Large Language Models Cannot Self-Correct Reasoning Yet. *arXiv* **2024**. [CrossRef]
29. JU, D.; Jiang, S.; Cohen, A.; Foss, A.; Mitts, S.; Zharmagambetov, A.; Amos, B.; Li, X.; Kao, J.T.; Fazel-Zarandi, M.; et al. To the Globe (TTG): Towards Language-Driven Guaranteed Travel Planning. *arXiv* **2024**. [CrossRef]
30. Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; Hu, Z. Reasoning with Language Model is Planning with World Model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 8154–8173.
31. Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.L.; Cao, Y.; Narasimhan, K. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv* **2023**. [CrossRef]
32. Panigrahi, A.; Pati, A.; Addula, S.R.; Pati, A.K.; Sahoo, G.; Dash, M. An Ensemble Machine Learning-Based Model for Blockchain Transactional Data Classification. In *Biologically Inspired Techniques in Many Criteria Decision-Making*; Dehuri, S., Dash, S., Thulasiram, R.K., Singh, R.H., Favorskaya, M., Eds.; Learning and Analytics in Intelligent Systems; Springer Nature: Cham, Switzerland, 2025; Volume 45, pp. 430–438. ISBN 978-3-031-82705-1.
33. Ziegler, C.-N.; McNee, S.M.; Konstan, J.A.; Lausen, G. Improving Recommendation Lists through Topic Diversification. In Proceedings of the 14th international conference on World Wide Web—WWW '05, Chiba, Japan, 10–14 May 2005; ACM Press: New York, NY, USA, 2005; p. 22.
34. Pu, P.; Chen, L.; Hu, R. A User-Centric Evaluation Framework for Recommender Systems. In Proceedings of the fifth ACM conference on Recommender systems, Chicago, IL, USA, 23–27 October 2011; pp. 157–164.

35. Nguyen, T.A.; Do, M.; Gerevini, A.E.; Serina, I.; Srivastava, B.; Kambhampati, S. Generating Diverse Plans to Handle Unknown and Partially Known User Preferences. *Artif. Intell.* **2012**, *190*, 1–31. [CrossRef]
36. Gavalas, D.; Kasapakis, V.; Konstantopoulos, C.; Mastakas, K.; Pantziou, G. A Survey on Mobile Tourism Recommender Systems. In Proceedings of the 2013 Third International Conference on Communications and Information Technology (ICCIT), Beirut, Lebanon, 19–21 June 2013; pp. 131–135.
37. Hansen, E.A.; Zhou, R. Anytime Heuristic Search. *J. Artif. Intell. Res.* **2007**, *28*, 267–297. [CrossRef]
38. Ragab Hassen, H.; Alabdeen, Y.Z.; Gaber, M.M.; Sharma, M. D2TS: A Dual Diversity Tree Selection Approach to Pruning of Random Forests. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 467–481. [CrossRef]
39. Shao, J.-J.; Zhang, B.-W.; Yang, X.-W.; Chen, B.; Han, S.-Y.; Wei, W.-D.; Cai, G.; Dong, Z.; Guo, L.-Z.; Li, Y.-F. ChinaTravel: An Open-Ended Benchmark for Language Agents in Chinese Travel Planning. *arXiv* **2025**. [CrossRef]
40. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Proceedings of the NIPS'22: 36th International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Curran Associates Inc.: Red Hook, NY, USA, 2022; pp. 24824–24837.
41. Rustagi, P.; Wray, K.H.; Saisubramanian, S. Multi-Objective Multi-Agent Path Finding with Lexicographic Cost Preferences. *arXiv* **2025**. [CrossRef]
42. Kunaver, M.; Požrl, T. Diversity in Recommender Systems—A Survey. *Knowl.-Based Syst.* **2017**, *123*, 154–162. [CrossRef]
43. Adomavicius, G.; Kwon, Y. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 896–911. [CrossRef]
44. Koenig, S.; Likhachev, M.; Furcy, D. Lifelong Planning A∗. *Artif. Intell.* **2004**, *155*, 93–146. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.