

TripScore: Aligning LLMs for Real-World Travel Planning via Expert-Calibrated Reward

Yincen Qu¹, Huan Xiao², Feng Li¹, Gregory Li³, Hui Zhou¹, Xiangying Dai¹, Xuan Huang¹,

¹Trip.com Group, ²Chongqing University, ³Stanford University,

Correspondence: yc.qu@trip.com

Abstract

In our deployed travel-planning service, most users give minimal inputs or free-form requests rather than the structured constraint checklists assumed by existing benchmarks. We therefore present TripScore, a behavior-grounded benchmark and evaluation framework built from real user logs and calibrated against 1,468 pairwise judgments by 203 travel experts. TripScore couples a hierarchical feasibility gate (format and commonsense) with a unified, point-wise reward that aggregates soft quality and preference fulfillment. Using TripScore as both evaluator and reward signal, we benchmark direct prompting, test-time compute, neuro-symbolic solvers, code agents, and fine-tuning. We find that reinforcement learning fine-tuning (e.g., GRPO) provides consistent gains over other approaches under the same base model and practical latency.

1 Introduction

Recently, large language models (LLMs) (Anil et al., 2023; OpenAI, 2023) have demonstrated promising capabilities in complex reasoning tasks. However, benchmarks such as meeting scheduling and graph coloring (Jimenez et al., 2024; Zheng et al., 2024; Stechly et al., 2025) reveal that planning remains a challenging domain to solve comprehensively. Among these challenges, travel planning has gained attention for its real-world impact and intrinsic complexity (Shao et al., 2024a; Chen et al., 2024).

Existing travel-planning benchmarks such as TravelPlanner (Xie et al., 2024) and ChinaTravel (Shao et al., 2024a) predominantly treat the task as a hard constraint satisfaction problem, emphasizing feasibility under explicit constraints. However, this framing can drift from authentic usage patterns. We create a web application that enables users to generate travel plans. With explicit user consent, we collected their click behavior and inputs. In our logs analyzing over

300,000 user requests collected over a one-month period, 39.3% of users specify only minimal constraints (destinations and duration), and 38.3% submit free-form customized requests. Only 22.4% tick the detailed constraint checklist assumed by prior work. This gap suggests that progress measured on constraint-heavy benchmarks can misrepresent progress in deployment, where planners must infer intent and trade-offs while still producing executable itineraries.

We therefore introduce **TripScore**, a behavior-grounded benchmark plus an expert-validated evaluator for real-world travel planning. TripScore is designed not only to measure feasibility and quality, but also to provide an optimizable point-wise reward that can support end-to-end policy optimization. Concretely, TripScore combines a hierarchical feasibility gate (format and commonsense constraints) to prevent invalid plans from being rewarded, and a unified reward that aggregates soft quality and preference satisfaction into a single scalar score. We learn reward weights from pairwise travel-expert preferences and validate robustness via ablation study, making the evaluator stable enough to serve as a training signal rather than merely a post-hoc judge.

Our contributions are four-fold. (1) A behavior-grounded dual-track benchmark (TripScore-M for the 39.3% minimal-constraint users, TripScore-F for the 38.3% free-form users) curated from real production logs. (2) An expert-validated unified reward whose weights are learned from 1,468 pairwise judgments by 203 travel specialists, reaching 60.29% agreement with experts (approximately 82.1% of the noise-adjusted human ceiling) and surpassing LLM-as-judge baselines. (3) A comprehensive method study showing that RL fine-tuning (GRPO) offers the best quality-latency trade-off, narrowing the closed-source model’s expert win rate against a 14B open-source model from 0.701 to 0.477 ($p=0.628$; a two-sided binomial test does

Benchmark	# Query	# City	Query Type	Query Source	Quality Evaluation	Output Form	Expert Validation
TravelPlanner (Xie et al., 2024)	1,225	312	CB Template	Synthetic	✗	Pass rate	✗
Trip Planning (Zheng et al., 2024)	1,600	48	CB Template	Synthetic	✗	Pass rate	✗
ChinaTravel (Shao et al., 2024a)	1,154	10	CB Free-Text	AI-Generated	Rule	Pass rate + Per-dim ranking	✗
TripTailor (Wang et al., 2025)	3,848	40	CB Free-Text	AI-Generated	LLM	Pass rate + Surpass rate	✗
RealTravel (Shao et al., 2025)	1,155	77	CB Free-Text	AI-Generated	LLM	Pass rate + Preference rate	✗
Travel-Sim (Yang et al., 2025b)	112	7	CB Free-Text	AI-Generated	Rule + LLM	Pass rate + Per-dim Score	✗
TripScore-M (ours)	4,593	821	CB Template	Real logs (behav.)	Rule + LLM	Unified score	✓
TripScore-F (ours)	277	134	Free-form req.	Real logs (req.)	Rule + LLM	Unified score	✓

Table 1: Comparison among TripScore and other travel-planning benchmarks. # Query denotes the number of queries in the dataset; # City denotes the number of unique cities covered by each benchmark. Quality Evaluation indicates how plan quality is assessed; Output Form denotes the reported metric(s). ✓ indicates expert validation is provided; ✗ indicates that quality is not evaluated or expert validation is not provided. Abbreviations: CB = constraints-based, behav. = user click behavior, req. = user free-form request.

not reject $H_0: p=0.5$). (4) A real-world deployment, where the GRPO-fine-tuned 14B model has fully replaced the prior closed-source LLM endpoint and now serves all user requests in our production travel-planning web application.

2 Related Work

Existing travel-planning approaches mix solver-based optimization (Ju et al., 2024; Hao et al., 2025; Shao et al., 2024a; Chen et al., 2024) with test-time compute (Gui et al., 2025; Kambhampati et al., 2024; Yang et al., 2025b). Solvers excel on rigid rule sets but mishandle vague intent, while test-time methods are more flexible yet too slow for deployment. Recent RL-based planners (Planner-R1 (Zhu et al., 2025), DeepTravel (Ning et al., 2025), TravelSim (Yang et al., 2025b)) report gains but rely on synthesized queries and LLM-only rewards without expert validation. On the benchmark side, TravelPlanner (Xie et al., 2024) and ChinaTravel (Shao et al., 2024a) focus on hard-constraint feasibility. TripCraft (Chaudhuri et al., 2025), TP-RAG (Ni et al., 2025), TripTailor (Wang et al., 2025), ITINERA (Tang et al., 2024) and RealTravel (Shao et al., 2025) add quality or preference dimensions but report fragmented per-rule scores, and no prior travel-planning benchmark we are aware of has been validated against domain experts at this scale. We differ on three axes simultaneously, namely *real-user logs* as queries, *expert-calibrated unified reward*, and *end-to-end RL with that reward as signal* (Table 1).

3 The Benchmark and Evaluator

Problem formulation. Given a user request q (destination, duration, and optional preferences) and a set of candidate resources $\mathcal{C}$ (flights, hotels, attractions from a frozen database), the task is to produce

an itinerary p that maximizes the TripScore reward $\mathcal{R}(p; q, \mathcal{C})$. The reward decomposes into four dimensions evaluated by distinct mechanisms:

Dimension	#Metrics	Mechanism	Role
Format	4	Rule	Feasibility gate
Commonsense	4	Rule	
Soft quality	7	Rule + LLM	Quality scoring
Preference	4/1	Rule / LLM	

3.1 Dataset Construction

Two tracks from real users. We deploy a web application (Appendix A) that logs roughly 10K daily requests under explicit user consent. **TripScore-M** samples interactions where users only specify destinations and duration; from logs we retain only these two fields. We then augment each query by sampling structured preferences (budget, pacing, physical effort, attraction interests) from a predefined inventory (Table 7, Appendix D) and attaching them to the query; these preferences are *not* extracted from user free-text but drawn from the fixed set to enable controlled preference evaluation. **TripScore-F** retains free-form requests longer than 10 words, spanning specific tasks (e.g., attending a meeting), day-by-day attraction lists (e.g., Day 1: city, Day 2: zoo), and theme trails (e.g., a Harry Potter trail). We additionally filter out requests referencing attractions absent from the frozen resource pool, ensuring all queries have corresponding candidate data. TripScore-F is intentionally smaller (219 test queries) due to per-query expert verification, privacy-filtering cost, and this resource-pool filtering.

Information retrieval and quality control. We bypass the tool-call stage and directly feed candidates retrieved from production-grade flight/hotel/attraction databases (Aug. 2025 frozen snapshot) to isolate planning capability from retrieval errors. Quality control combines empty-

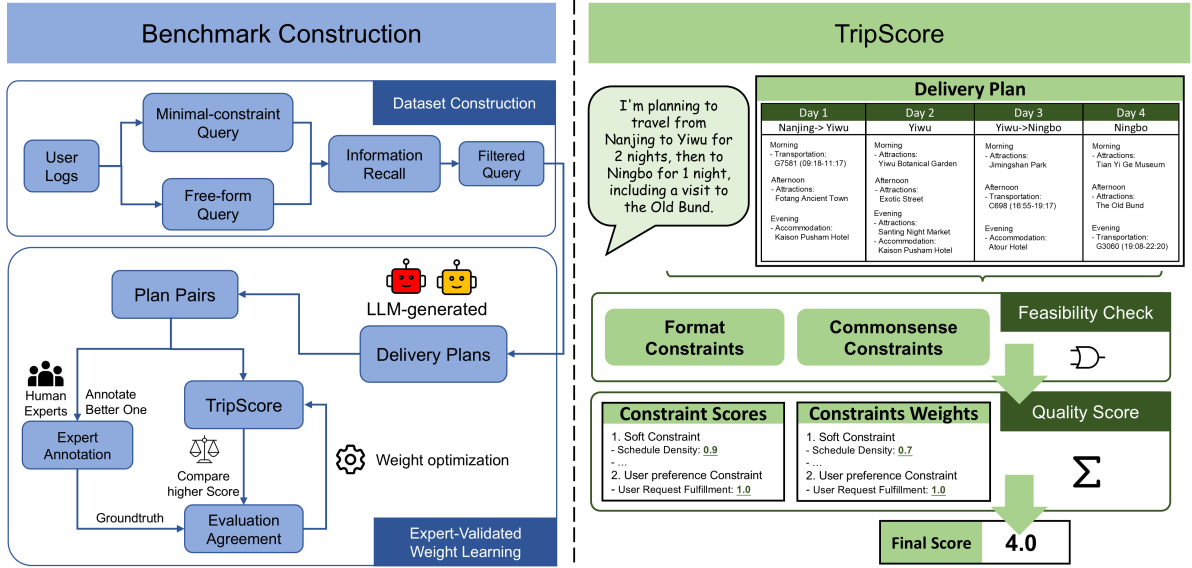


Figure 1: Construction process of TripScore dataset and illustration of TripScore.

retrieval filtering, query-evidence consistency check, and human review of ambiguous or non-sensical requests. The final benchmark totals 4,870 queries (3,493 train / 158 val / 1,219 test) spanning 897 cities, 9,376 hotels, and 10,997 attractions. Full pipeline and detailed statistics are in Appendix B and C.

3.2 TripScore Evaluator

3.2.1 Constraint Introduction

TripScore evaluates plans along four dimensions. **Format** (response structure, information grounding, name/time accuracy, description relevance) and **Commonsense** (information completeness, location consistency, operating hours, chronological order) together form the feasibility gate. **Soft constraints** (schedule density, hotel consistency, daytime utilization, geographic clustering, attraction quality) capture general plan quality. We score most with deterministic rules and use LLM ratings only for semantic criteria (iconic landmarks, attraction diversity). **Personal preferences** are structured for TripScore-M (budget, pacing, physical effort, attraction interests) and free-form for TripScore-F (matched to the user’s request via LLM judgment). Full per-metric definitions and Rule/LLM mechanisms are in Table 7 (Appendix D), and cross-benchmark constraints comparison is in Appendix E.

3.2.2 Two-Stage Evaluation

We separate feasibility from quality hierarchically. **Stage 1 (Feasibility Gate)** applies a severe penalty

for format violations ($S_{\text{form}} = -3$, $S_{\text{com}} = 0$) and a non-rewarding score for commonsense violations ($S_{\text{form}} = 1$, $S_{\text{com}} = -1$). Only plans with completely valid formats and commonsense ($S_{\text{form}} = 1$, $S_{\text{com}} = 1$) can successfully pass this gate, ensuring that invalid outputs never outscore valid ones. **Stage 2 (Quality Scoring)** first aggregates sub-metric scores within each quality dimension, then combines the four dimensions into a unified reward:

$$\bar{S}_{\text{soft}} = \frac{\mathbf{w}_1^\top \mathbf{S}_{\text{soft}}}{\|\mathbf{w}_1\|_1}, \quad \bar{S}_{\text{pref}} = \frac{\mathbf{w}_2^\top \mathbf{S}_{\text{pref}}}{\|\mathbf{w}_2\|_1}, \quad (1)$$

$$\mathcal{R}(\mathbf{S}) = S_{\text{form}} + S_{\text{com}} + w_3 \bar{S}_{\text{soft}} + w_4 \bar{S}_{\text{pref}} \quad (2)$$

where $\mathbf{S}_{\text{soft}}$ and $\mathbf{S}_{\text{pref}}$ collect sub-scores in $[0, 1]$ (seven soft and up to four preference metrics). $\mathbf{w}_1$ and $\mathbf{w}_2$ are their within-dimension weights, and w_3, w_4 scale the aggregated soft and preference terms. $\boldsymbol{\theta} = (\mathbf{w}_1, \mathbf{w}_2, w_3, w_4)$ comprises 13 learnable parameters. Sub-metric definitions, feasibility constants, worked examples, and sensitivity analysis are in Appendix N, Appendix O, and Appendix F.3.

3.2.3 Expert-Validated Weight Learning

Expert annotation. To validate our reward score, we recruited 203 travel experts to rank pairs of travel plans. We restrict annotation to pairs where both plans pass the feasibility gate (format + commonsense constraints), so that expert attention targets subjective quality dimensions rather than rule-verifiable criteria. For each query, we presented

two feasible itineraries to three experts and asked them to choose which one is superior. Experts were assigned destinations with which they were familiar. In total, we obtained 1,468 route pairs with three annotations per pair, yielding majority-vote labels; pairs without majority consensus are excluded. The labeled pairs are randomly split at the pair level into train/validation/test for weight learning. Results show moderate inter-annotator agreement (Cohen’s $\kappa = 0.5421$ for pairwise; Fleiss’s $\kappa = 0.5039$ across three raters), with mean pairwise agreement of 71.69% and overall three-rater agreement of 59.00%. Further details are provided in Appendix F.1.

Weight optimization. We optimize the weights in Equation 2 using expert labels (majority vote per pair). Pairs without a majority consensus (including ties) are excluded from the training set. Given route pairs and their labels, we learn a scoring function $\mathcal{R}(p)$ whose induced pairwise ordering maximizes agreement with expert preferences.

We formulate weight learning as:

$$\theta^* = \arg \max_{\theta} \sum_i \mathbb{I}[\text{sgn}(\Delta \mathcal{R}^{(i)}) = y^{(i)}] \quad (3)$$

where $y^{(i)}$ is the human label. $\Delta \mathcal{R}^{(i)} = \mathcal{R}(p_1^{(i)}) - \mathcal{R}(p_2^{(i)})$. $\mathcal{R}(p_1^{(i)})$ and $\mathcal{R}(p_2^{(i)})$ are the final reward scores of the plan p_1 and p_2 correspondingly.

We perform grid search over 13 parameters, and we choose grid search rather than probabilistic ranking models (e.g., Bradley-Terry) for interpretability of feature importance in this low-dimensional space (Appendix F.2). Empirically, TripScore (grid-search weights) achieves 60.29% accuracy with Kendall’s $\tau=0.2026$, versus a Bradley-Terry model trained on the same 13 features at 58.20% accuracy and $\tau=0.1670$ ($p=0.003$), confirming that the grid-search approach provides both better expert agreement and auditable, controllable weights. To mitigate overfitting on the 1,468-pair set, we apply nested stratified 5-fold cross-validation, $1,000\times$ bootstrap 95% CIs, and Kendall’s τ analysis. The selected weights achieve cross-validated validation accuracy 0.6175 ± 0.0259 (bootstrap 0.6031, 95% CI [0.5343, 0.6667]), with positive ordinal alignment ($\tau=0.1784$ on val). Under a K-class symmetric-noise model ($K = 3$), the 60.29% raw agreement corresponds to a noise-adjusted latent agreement of $\approx 68.9\%$, covering $\approx 82.1\%$ of the human re-

liability ceiling ($r \approx 83.9\%$). The details are in Appendix F.4.

Evaluation agreement. Across LLM backbones, TripScore surpasses the best LLM-as-judge baselines (Zheng et al., 2023; Singh et al., 2024) on validation agreement. In Table 2, we see 60.29% with Gemini-3-flash and 61.22% with Gemini-3.1-pro. Point-wise judges are unstable to ties (Gemini-3-flash drops 11.1 pp once ties count as wrong). Pair-wise judging is competitive but requires $O(n \log n)$ comparisons. Our point-wise score avoids both, with full tie-handling definitions and per-backbone results. The details are provided in Appendix F.5.

Method	LLM	Accuracy	Kendall’s τ
Point-wise	GPT-5.4	47.42 (57.06)	0.0915
	Claude-opus-4.6	43.19 (62.04)	0.1745
	Gemini-3.1-pro	51.17 (61.24)	0.1890
	Gemini-3-flash	51.64 (60.77)	0.1701
	DeepSeek-V3.2	51.43 (54.55)	0.0618
Pair-wise	GPT-5.4	54.46	0.0820
	Claude-opus-4.6	57.75	0.1575
	Gemini-3.1-pro	60.09	0.1977
	Gemini-3-flash	58.22	0.1590
	DeepSeek-V3.2	54.46	0.0811
Ours	Gemini-3.1-pro	61.22	0.2155
	Gemini-3-flash	60.29	0.2026
	GPT-5.4	56.50	0.1307
	DeepSeek-V3.2	57.84	0.1573

Table 2: Comparison with human annotations for our method and LLM-as-judge baselines. All accuracy values are computed on pairs with a clear majority-vote winner (tie/neither cases excluded from the denominator). For point-wise methods, values in parentheses further exclude cases where the judge itself returns a tie.

4 Reinforcement Learning for Planning

Travel plans live in a vast combinatorial space where supervised imitation generalizes poorly, as minor edits can flip an itinerary from feasible to invalid. We therefore turn to reinforcement learning. Beyond evaluation, the expert-validated TripScore reward can also serve as a dense, point-wise optimization signal that provides per-rollout supervision to discriminate between near-feasible variants.

We adopt Grouped Relative Policy Optimization (GRPO) (Shao et al., 2024b) (Figure 3), which consumes fine-grained ordinal rewards directly and uses within-group normalization that is robust to the long-tailed reward distribution. For each query q we sample G rollouts $\{o_i\}_{i=1}^G$ from $\pi_{\theta_{\text{old}}}$, score each with our evaluator, and standardize within

Method	LLM	TripScore-M				TripScore-F			
		FPR↑	CPR ↑	Reward↑	Time↓	FPR↑	CPR↑	Reward↑	Time↓
Direct									
Direct	GPT-5.4	82.18	92.02	1.69	25.09	89.35	78.76	1.81	27.70
	Claude Opus 4.6	79.33	90.49	1.49	36.86	72.20	85.85	1.49	44.04
	Kimi-K2.5	78.50	91.97	1.46	39.84	66.66	80.56	1.12	39.51
	DeepSeek-V3.2	73.92	96.00	1.26	27.61	75.78	78.10	1.44	25.03
	Qwen3-8B	27.14	94.98	-1.31	9.37	39.73	56.30	-1.00	10.48
	Qwen3-14B	39.93	88.70	-0.72	12.35	51.59	67.26	-0.24	13.15
	Qwen3-32B	42.32	95.13	-0.53	19.33	60.27	71.86	0.49	19.83
Test-Time Compute									
CoT	GPT-5.4	83.83	89.43	1.72	29.48	82.19	80.55	1.92	27.14
	Qwen3-8B	22.53	93.47	-1.61	13.77	36.07	51.89	-1.24	13.51
	Qwen3-14B	30.85	83.43	-1.33	17.59	45.79	70.40	-0.44	17.27
ReAct	GPT-5.4	81.77	91.68	1.66	42.87	87.67	76.55	2.16	43.93
	Qwen3-8B	31.10	93.14	-1.23	18.98	40.10	72.68	-0.73	21.88
	Qwen3-14B	54.89	81.79	-0.06	26.16	60.73	80.45	0.66	31.75
LLM-Modulo	GPT-5.4	93.82	91.09	2.36	102.7	94.95	86.47	3.13	106.7
	Qwen3-8B	44.33	90.32	-0.49	34.17	50.22	72.72	-0.16	40.23
	Qwen3-14B	49.54	85.42	-0.29	64.38	56.62	81.56	0.27	54.14
HyperTree	GPT-5.4	78.26	91.72	1.44	58.10	89.95	79.69	2.03	45.96
	Qwen3-8B	14.67	85.75	-2.19	31.31	4.56	100.00	-2.62	35.92
	Qwen3-14B	27.25	66.92	-1.66	34.21	42.01	63.04	-0.72	41.71
Neural-Symbolic									
TTG	GPT-5.4	37.10	99.38	-0.56	2.21	12.78	100.00	-2.17	2.03
NESY	GPT-5.4	57.96	98.74	0.60	77.92	1.82	100.00	-2.89	46.44
	Qwen3-8B	47.37	98.04	-0.25	62.58	2.73	100.00	-2.83	42.12
Fine-Tuning									
SFT	Qwen3-8B	74.02	91.23	1.17	9.00	77.16	82.84	1.53	11.85
	Qwen3-14B	82.80	93.41	1.70	12.69	85.38	76.47	1.89	15.34
RFT	Qwen3-8B	74.49	93.66	1.26	9.77	82.19	81.11	1.82	12.21
	Qwen3-14B	84.75	93.44	1.83	15.13	84.02	77.71	1.83	14.43
GRPO	Qwen3-8B	75.59	90.05	1.25	8.97	88.31	76.72	2.04	9.67
	Qwen3-14B	84.91	94.56	1.87	15.34	90.27	78.50	2.20	14.04

Table 3: Performance comparison of various planning approaches across different LLMs. FPR and CPR are multiplied by 100. The **best** and **second-best** results are highlighted. Time denotes the average inference time in seconds. Values with gray shading indicate inference times exceeding 40 seconds.

group: $\hat{r}_i = (r_i - \text{mean}(r))/\text{std}(r)$. The token-level advantage is $\hat{A}_{i,t} = \hat{r}_i$, and the policy is optimized with the clipped GRPO objective.

We score rollouts with the *rule-only* component of TripScore. Putting LLM-based sub-scores inside the training loop would invite Goodhart-style hacking against a learnable judge, while rule-based scoring stays deterministic and verifiable. As shown in Section 5, gains from rule-only RL still transfer to the full evaluator and pairwise expert judgments, indicating that mastering verifiable feasibility is a sufficient foundation for downstream semantic quality.

5 Experiments

5.1 Experimental Setup

We evaluate GPT-5.4, Claude Opus 4.6, Kimi-k2.5 (Team, 2026), DeepSeek-V3.2 (DeepSeek-AI et al., 2025), and Qwen3- $\{8B, 14B, 32B\}$ (Yang et al., 2025a) across four method families: *Direct*, *Test-time compute* (ZS-CoT (Wei et al., 2022), ReAct (Yao et al., 2023), LLM-Modulo (Kambhampati et al., 2024), HyperTree (Gui et al., 2025)), *Neuro-Symbolic* (TTG (Ju et al., 2024), NESY (Shao et al., 2024a)), and *Fine-tuning* (SFT, RFT (Yuan et al., 2023), GRPO (Shao et al., 2024b)). We report Format/Commonsense Pass Rate (FPR/CPR; CPR is conditioned on passing the

format gate), Reward, and inference time (mixed closed-API and local inference, indicative only). Gemini-3-flash serves as the LLM judge for soft sub-metrics (temperature 0). All fine-tuning experiments (SFT, RFT, GRPO) report results from a single controlled run (seed 42); multi-seed variance is not characterized due to computational cost. FPR/CPR definitions and per-method implementation details are in Appendix G.

5.2 Experiment Analysis

Test-time compute and solvers trade quality for latency. In Table 3, test-time methods lift feasibility but at prohibitive cost. For example, LLM-Modulo+GPT-5.4 achieves the top reward (3.13) and FPR (94.95%) on TripScore-F yet quadruples inference time over Direct (106.7s vs. 27.70s), making it impractical for interactive deployment. Neuro-symbolic solvers (TTG, NESY) reach near-perfect commonsense (CPR $\approx 99\%$) on the tiny format-passing subset but collapse on free-form FPR ($\sim 1.8\%$ for NESY on TripScore-F). The main reason is they treat ambiguous intent as hard constraints, leaving them over-constrained even with extensive prompt engineering and ample retry budgets (Appendix J). We also find that test-time search is bounded by the base model’s capability. HyperTree on Qwen3-8B drops reward from -1.00 (Direct) to -2.62, indicating that test-time compute amplifies the strength of the base model rather than substituting for it.

RL fine-tuning internalizes planning. GRPO offers the best trade-off between reward and latency. GRPO on Qwen3-8B reaches reward 2.04 on TripScore-F under 10s latency, surpassing Qwen3-8B Direct (-1.00) and even GPT-5.4 Direct (1.81). On Qwen3-14B, GRPO attains the second-best on TripScore-M and TripScore-F rewards while remaining over $7\times$ faster than LLM-Modulo+GPT-5.4. This supports internalizing planning behaviors as parametric knowledge being more compute-efficient than test-time search.

5.3 Travel Plan Quality

To disentangle quality from feasibility, we conduct a controlled pairwise comparison restricted to queries where both plans pass feasibility, pairing each Qwen3-14B variant against GPT-5.4. Each pair is scored both by the TripScore reward and by three experts (we keep only unanimous annotations).

Table 4 shows that SFT, RFT, and especially

GRPO progressively narrow the quality gap with GPT-5.4. On TripScore-M under TripScore reward, GPT-5.4’s win rate drops from 0.726 (vs. Direct) to 0.459 (vs. GRPO). These gains generalize well to free-form queries on TripScore-F (0.744 $\rightarrow$ 0.522). Human expert evaluation further corroborates this trend. Specifically, on TripScore-M the expert win rate narrows progressively: 0.701 $\rightarrow$ 0.549 $\rightarrow$ 0.504 $\rightarrow$ 0.477 (Direct $\rightarrow$ SFT $\rightarrow$ RFT $\rightarrow$ GRPO). For GRPO, the two-sided binomial test does not reject $H_0: p=0.5$ ($p=0.628$), though we note that failure to reject does not formally establish equivalence. Crucially, the reward and expert rankings move in lockstep across all four variants, providing evidence that TripScore tracks human-perceived quality.

Dataset	Opposing Method	Win Rate	p
TripScore Reward			
TripScore-M	Direct (Qwen3-14B)	0.726	0.002
	SFT (Qwen3-14B)	0.587	0.632
	RFT (Qwen3-14B)	0.515	1.000
	GRPO (Qwen3-14B)	0.459	0.484
TripScore-F	Direct (Qwen3-14B)	0.744	0.004
	SFT (Qwen3-14B)	0.659	0.032
	RFT (Qwen3-14B)	0.552	0.711
	GRPO (Qwen3-14B)	0.522	0.245
Expert Selection			
TripScore-M	Direct (Qwen3-14B)	0.701	0.004
	SFT (Qwen3-14B)	0.549	0.137
	RFT (Qwen3-14B)	0.504	0.839
	GRPO (Qwen3-14B)	0.477	0.628
TripScore-F	Direct (Qwen3-14B)	0.785	0.002
	SFT (Qwen3-14B)	0.614	0.008
	RFT (Qwen3-14B)	0.561	0.636
	GRPO (Qwen3-14B)	0.513	0.492

Table 4: Pairwise quality comparison: Direct (GPT-5.4) vs. Qwen3-14B variants. Win rate > 0.5 favors GPT-5.4. The p value is from a two-sided binomial test against $H_0: p=0.5$; $p > 0.05$ indicates no statistically significant preference between the two methods.

5.4 Evaluator Robustness Analysis

We re-score outputs under Rule-only, LLM-only, and leave-one-out (LOO) ablations of individual sub-scores (full protocols and tables in Appendix I). System-level Kendall’s τ between Full TripScore and the ablated rankings remains high on TripScore-M (Rule-only $\tau=0.85$, LLM-only $\tau=0.60$), but on TripScore-F the Rule-only ranking flips to $\tau=-0.07$ while LLM-only stays strong ($\tau=0.88$). Removing Request Fulfillment alone collapses correlation to $\tau=0.01$, confirming that rigid rule compliance is insufficient for free-form

queries. Our central claim remains stable across all variants. Specifically, GRPO (Qwen3-14B) matches GPT-5.4 on feasible plans (2.42 vs 2.43), outperforms it on Rule-only metrics (1.91 vs 1.88), and trails marginally on LLM-only semantic metrics (2.11 vs 2.15), indicating that RL fine-tuning bridges the gap primarily by mastering execution rules, providing a structural foundation for semantic quality.

5.5 Production Deployment

To validate real-world impact, we conducted an on-line A/B test comparing the GRPO-fine-tuned 14B model against the prior closed-source general LLM API that previously served all production traffic. Over 539K generated itineraries (425,784 from the prior API; 113,783 from GRPO), we measured plan adoptability (users accepting the generated plan) and constraint-aware adoptability (acceptance of plans satisfying stated constraints). GRPO achieves 16.1% / 32.6% on these metrics versus 7.35% / 18.9% for the prior API, a relative lift of +121%. Based on these results, the GRPO model now fully replaces the closed-source endpoint in production.

5.6 Can Code Agents Solve Planning?

We test whether frontier code-execution agents can improve planning by writing scripts. On 50 random TripScore-M queries, Claude Code (v2.1.92, Opus 4.6) with an explicit tool-use prompt degrades FPR (84% vs. 96% direct), reward (1.27 vs. 2.48), and quintuples latency (204s vs. 38s; Table 5). Inspecting all 50 traces, 84% either skipped code or used tools trivially (e.g., print) despite explicit encouragement; only 7/50 wrote useful code (e.g., computing weekday), and 1/50 fell into an infinite loop. On a sole-planning task with all information in-context, scripting an ambiguous multi-objective routing problem is harder than reasoning natively—what limits performance is parametric planning competence, not tool-use proficiency.

Metric	Direct	Claude Code
FPR	96.0%	84.0%
CPR	93.8%	73.8%
Reward	2.48	1.27
Avg. Generation Time	38.3s	204.2s

Table 5: Direct prompting vs. Claude Code on 50 TripScore-M queries (same queries, same model). Full prompt and case analysis in Appendix K.

6 Conclusion

We introduce TripScore, a behavior-grounded benchmark and expert-validated reward calibrated against 1,468 expert pairwise judgments. Across prompting, test-time compute, neuro-symbolic solvers, code agents, and fine-tuning, RL on a 14B open-source model matches GPT-5.4 under expert evaluation, and is now deployed in production replacing the prior closed-source endpoint.

Limitations

We highlight several limitations of this work. First, our benchmark focuses on sole-planning given retrieved information, rather than end-to-end tool use. While this isolates planning competence, it may overestimate performance relative to interactive settings with retrieval errors, tool failures, and partial observability. Second, parts of TripScore rely on LLM-based sub-metrics (iconic landmark coverage, attraction diversity, free-form request fulfillment with Gemini-3-flash as judge). To mitigate Goodhart effects, our RL/RFT training uses the rule-only scorer, but the full evaluator may still inherit judge-specific biases (e.g., sensitivity to prompting, model version updates). Future iterations could ensemble multiple judges or ground semantic checks in more granular verifiable facts. Third, TripScore-F is smaller than TripScore-M (219 test queries), which may limit statistical power for fine-grained ablations and for detecting small effect sizes in pairwise comparisons. Expanding free-form coverage and auditing distribution shifts across locales, languages, and traveler profiles are important next steps. Finally, due to the high computational cost of fine-tuning and full-pipeline evaluation, we report results from a single run with a fixed random seed (42). Future versions should report variance across multiple seeds.

References

- Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, and 33 others. 2023. [Gemini: A family of highly capable multimodal models](#). *CoRR*, abs/2312.11805.
- Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, Jiayin Zhang, Juanzi Li, and

- Lei Hou. 2023. [Benchmarking foundation models with language-model-as-an-examiner](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Soumyabrata Chaudhuri, Pranav Purkar, Ritwik Raghav, Shubhojit Mallick, Manish Gupta, Abhik Jana, and Shreya Ghosh. 2025. [Tripcraft: A benchmark for spatio-temporally fine grained travel planning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 17035–17064. Association for Computational Linguistics.
- Aili Chen, Xuyang Ge, Ziquan Fu, Yanghua Xiao, and Jiangjie Chen. 2024. [Travelagent: An AI assistant for personalized travel planning](#). *CoRR*, abs/2409.08069.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *CoRR*, abs/2501.12948.
- Runquan Gui, Zhihai Wang, Jie Wang, Chi Ma, Hui-Ling Zhen, Mingxuan Yuan, Jianye Hao, Defu Lian, Enhong Chen, and Feng Wu. 2025. [Hypertree planning: Enhancing LLM reasoning via hierarchical thinking](#). *CoRR*, abs/2505.02322.
- Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. 2025. [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: Human Language Technologies (Volume 1: Long Papers)*, pages 3434–3483, Albuquerque, New Mexico. Association for Computational Linguistics.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. [Swe-bench: Can language models resolve real-world github issues?](#) In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Da Ju, Song Jiang, Andrew Cohen, Aaron Foss, Sasha Mitts, Arman Zharmagambetov, Brandon Amos, Xian Li, Justine T Kao, Maryam Fazel-Zarandi, and Yuandong Tian. 2024. [To the globe \(TTG\): Towards language-driven guaranteed travel planning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 240–249, Miami, Florida, USA. Association for Computational Linguistics.
- Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. [Position: LLMs can’t plan, but can help planning in llm-modulo frameworks](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with pagedattention](#). In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP ’23*, page 611–626, New York, NY, USA. Association for Computing Machinery.
- Hang Ni, Fan Liu, Xinyu Ma, Lixin Su, Shuaiqiang Wang, Dawei Yin, Hui Xiong, and Hao Liu. 2025. [TP-RAG: benchmarking retrieval-augmented large language model agents for spatiotemporal-aware travel planning](#). *CoRR*, abs/2504.08694.
- Yansong Ning, Rui Liu, Jun Wang, Kai Chen, Wei Li, Jun Fang, Kan Zheng, Naiqiang Tan, and Hao Liu. 2025. [Deeptravel: An end-to-end agentic reinforcement learning framework for autonomous travel planning agents](#). *CoRR*, abs/2509.21842.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- Jie-Jing Shao, Xiao-Wen Yang, Bo-Wen Zhang, Baizhi Chen, Wen-Da Wei, Guohao Cai, Zhenhua Dong, Lan-Zhe Guo, and Yu-feng Li. 2024a. [Chinatrans: A real-world benchmark for language agents in chinese travel planning](#). *CoRR*, abs/2412.13682.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024b. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *CoRR*, abs/2402.03300.
- Zijian Shao, Jiancan Wu, Weijian Chen, and Xiang Wang. 2025. [Personal travel solver: A preference-driven LLM-solver system for travel planning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27622–27642, Vienna, Austria. Association for Computational Linguistics.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. [Hybridflow: A flexible and efficient RLHF framework](#). In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.
- Harmanpreet Singh, Nikhil Verma, Yixiao Wang, Manasa Bharadwaj, Homa Fashandi, Kevin Ferreira, and Chul Lee. 2024. [Personal large language model agents: A case study on tailored travel planning](#). In *Proceedings of the 2024 Conference on Empirical*

- Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 486–514. Association for Computational Linguistics.
- Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2025. [On the self-verification limitations of large language models on reasoning and planning tasks](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Yihong Tang, Zhaokai Wang, Ao Qu, Yihao Yan, Zhaofeng Wu, Dingyi Zhuang, Jushi Kai, Kebing Hou, Xiaotong Guo, Jinhua Zhao, Zhan Zhao, and Wei Ma. 2024. [ItiNera: Integrating spatial optimization with large language models for open-domain urban itinerary planning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1413–1432, Miami, Florida, US. Association for Computational Linguistics.
- Kimi Team. 2026. [Kimi K2.5: visual agentic intelligence](#). *CoRR*, abs/2602.02276.
- Kaimin Wang, Yuanzhe Shen, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. 2025. [Triptailor: A real-world benchmark for personalized travel planning](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 9705–9723. Association for Computational Linguistics.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. [Travelplanner: A benchmark for real-world planning with language agents](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025a. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- Dongjie Yang, Chengqiang Lu, Qimeng Wang, Xinbei Ma, Yan Gao, Yao Hu, and Hai Zhao. 2025b. [Plan your travel and travel with your plan: Wide-horizon planning and evaluation via LLM](#). *CoRR*, abs/2506.12421.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. [Scaling relationship on learning mathematical reasoning with large language models](#). *CoRR*, abs/2308.01825.
- 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. 2024. [NATURAL PLAN: benchmarking llms on natural language planning](#). *CoRR*, abs/2406.04520.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Siyu Zhu, Yanbin Jiang, Hejian Sang, Shao Tang, Qingquan Song, Biao He, Rohit Jain, Zhipeng Wang, and Alborz Geramifard. 2025. [Planner-r1: Reward shaping enables efficient agentic RL with smaller llms](#). *CoRR*, abs/2509.25779.

A Web Application Usage

Our web application interface is designed to ensure effortless access for users worldwide. Figure 2 showcases the main screen through which users submit their travel requests and generate travel plans. Users can select destinations and duration, choose either enumerated preferences or type free text needs. Then users can click on “Plan a Trip with AI” to generate tailored travel plans.

To mitigate the risk of collecting low-quality data, we enforce several quality-control measures. First, users must register with an email or phone number for identity verification. Second, we audit logs in real time and take action to ban accounts that generate NSFW content, open multiple sessions or operate at abnormal speed. Third, we use an NSFW keyword list blocks to prevent users from harmful outputs. Last, each user is initially allowed at most 100 interactions per day.

Consent, privacy, and de-identification. Users are informed that their interactions may be used to improve the system and for research, and logging is enabled only after explicit consent. For research processing, we remove direct identifiers and store requests in a de-identified form. We do

Starting from **Shanghai** ▾

📍 **Heading to**
Country / City / Landmark

📅 **Date/Duration**
Select

♥ **Preferences** ^

With Whom

☐ Solo ☐ Family ☐ Couple ☐ Friends ☐ Elderly

Travel Style

☐ Cultural ☐ Classic ☐ Nature ☐ Cityscape ☐ Historical

Travel Pace

☐ Ambitious ☐ Relaxed

Accommodation

☐ Comfort ☐ Premium ☐ Luxury

Other Needs

Enter more preferences here

0/1000

[Clear](#) [Confirm](#) [Create It Myself](#) [Plan a Trip with AI](#)

Figure 2: Main interface of the travel-planning web application used for collecting real-user interaction logs.

not include account identifiers, emails/phone numbers, or payment information in the benchmark data. We also exclude requests that contain obvious personally identifying content. The retained data—destinations, durations, general departure cities, and stated preferences—are common, non-unique attributes shared across large user populations and do not permit re-identification of individual users. Access to raw logs is restricted to authorized personnel, and the benchmark snapshot used in this paper is frozen (August 2025) to support versioned evaluation.

B Dataset Construction Pipeline

Real data from real users. A key advantage of TripScore is that our data is collected from real, intrinsically motivated users, rather than paid crowd workers or AI. We developed a web application that allows users to initiate their own travel plans by selecting destinations and duration, and by providing either predefined preference constraints or free-form needs. With explicit user consent, we log their selections and requests, recording approximately ten thousand requests per day. In our logs analyzing over 300,000 user requests collected over a one-month period, 39.3% of users select only

destinations and duration, 38.3% submit free-form customized requirements, and 22.4% choose predefined preference options. Based on this distribution, we establish two distinct evaluation tracks. The first track strictly retains only destinations and duration, while the second track accommodates free-form user requests without predefined constraints.

TripScore-M query selection. To enhance authenticity, we incorporate real user requests from logs under appropriate permissions. We construct minimal-constraint queries by randomly sampling real user interactions where only destinations and trip duration are provided. While these users initially provide minimal input, implicit preferences often exist. We augment these queries with explicit preferences, selected from a predefined set outlined in the personal preference constraints in Table 7, to test the planner’s ability to handle such constraints when they are specified or inferred, ensuring the benchmark covers the full spectrum of planning complexity.

TripScore-F query selection. We construct free-form queries from user interactions where they submit free-form text queries. These queries capture genuine planning intent rather than contrived prompts. These queries span a wide range, from

specific tasks (e.g., attending a meeting), to day-by-day attraction preferences (e.g., Day 1: city, Day 2: zoo), and to theme-based trips (e.g., a Harry Potter trail). To maintain quality, we exclude requests under 10 words, as such inputs are overly simplistic and insufficiently challenging for models to generate tailored itineraries. Note that the TripScore-F set is kept relatively small (219 queries) due to the high cost of rigorous expert verification and privacy filtering.

Relevant information recall. To ensure adequate and accurate retrieval, we rely on production-grade industry data curated and maintained by professional operators. In deployment, an agent utilizes corresponding search tools to query the travel database (covering flights, hotels, and attractions) using keywords extracted from the user request. The retrieval process continues iteratively until relevant options for all requested categories are found or a maximum step limit is reached. The planning model then consumes both the collected information and the user request to produce the final plan. However, to streamline the evaluation process in this benchmark, we bypass the tool-call stage and directly provide the planning model with the information collected by the agent. This approach allows us to focus specifically on assessing the model’s planning capabilities while ensuring access to all necessary data for generating final travel plans.

Quality control. After collecting the user request and corresponding information, we implement a rigorous quality assurance process to remove the invalid request, leveraging both LLMs and human evaluation. We filter out requests with empty retrieved information, ensuring that each query has adequate information for planning. Then we use Gemini-3-flash to filter out samples whose relevant information is inconsistent with the user query. After the automated filtering step, human annotators review the remaining pool to remove requests that are prohibitively difficult or nonsensical.

C Benchmark Details

In Table 6, we list the detailed data distribution on training, validation and test set.

D Constraints

As illustrated in Table 7, we evaluate plan feasibility and quality across four dimensions, which

Dataset	Type	#Examples	#City	#Hotel	#Attraction
Training	TripScore-M	3,493	781	7,830	9,788
	TripScore-F	58	41	638	1,313
Validation	TripScore-M	100	70	1,057	1,525
	TripScore-F	219	122	1,894	2,057
Test	TripScore-M	1,000	358	4,459	5,452
	TripScore-F	219	122	1,894	2,057
All		4,870	897	9,376	10,997

Table 6: Statistics of dataset distribution. Splits are random at the query level; no stratification by user, city, hotel, or attraction is applied. The City, Hotel, and Attraction columns report post-split descriptive statistics (entity counts), not splitting criteria.

include format and commonsense constraints for feasibility checks, as well as soft and personal preference criteria for overall quality assessment.

Format constraints ensure the output is structurally parsable and grounded in the provided database. Commonsense constraints enforce physical and temporal validity, such as travel times and operating hours. Soft constraints capture general quality indicators like route efficiency and variety, while Personal Preference constraints measure alignment with specific user needs, ensuring the plan is not just valid, but desirable.

E Constraints Comparison

Constraints across benchmarks are summarized in Table 8, showing which dimensions are evaluated and by what mechanism. TripScore distinguishes itself by covering a broader spectrum of constraints compared to prior benchmarks like TravelPlanner or ChinaTravel. While previous works primarily focus on hard constraints (feasibility), TripScore integrates a comprehensive set of soft and preference constraints to evaluate plan quality holistically. Furthermore, our hybrid evaluation mechanism uniquely combines deterministic rule-based checks for objective criteria with LLM-based assessments for semantic and subjective qualities, providing a more robust and expert-aligned evaluation signal.

F Expert Validation

F.1 Expert Annotation

We conducted a comparative evaluation, integrating expert assessments with our automated framework. We utilized DeepSeek-V3.2, Qwen3-32B and GPT-5.4 to independently generate travel plans based on 3,000 user queries. After filtering out plans that failed to satisfy format and commonsense constraints, we obtained a final total of 1,468 pairwise

Constraint	Description	Evaluation
Format Constraint		
Response Format	All responses must follow the requested structure and organization exactly as specified.	Rule
Information Verification	All attractions, transportation, and hotels in the plan must come from the provided information, otherwise, it will be considered a hallucination.	Rule
Information Accuracy	Details like name and departure time must match the provided information.	Rule
Information Relevance	All descriptions must specifically match their intended attractions (e.g., not mixing up or blending details from different places).	Rule
Commonsense Constraint		
Information Completeness	All necessary information must be included, especially accommodation for each destination with multi-day stay and essential transportation.	Rule
Chronological Order	All activities must be listed in chronological order.	Rule
Location Consistency	Each day's activity must be scheduled in the city where the traveler is actually present and change only after any required transportation.	Rule
Operating Hours	Visits to attractions must only be scheduled during their confirmed opening hours.	Rule
Travel Block-Out	No activities can be scheduled between departure and arrival times during transportation.	Rule
Transport. Consistency	Arrange the transportation from the starting city to each destination in sequence to avoid jumps or repetitive routes.	Rule
Soft Constraint		
Schedule Density	Each day's plan should be thoughtfully paced, ensuring neither overly lengthy nor excessively brief periods of activity.	Rule
Hotel Consistency	When staying in the same city, the same hotel should be used throughout to avoid unnecessary check-ins and check-outs.	Rule
Daytime Utilization	Fill daytime hours when evening travel is planned; avoid starting the day with attractions that open late.	Rule
Unique Attractions	Attractions should not appear more than once in the plan.	Rule
Location Clustering	When possible, group attractions that are close to each other to reduce travel time and maximize sight-seeing time.	Rule
Iconic Landmarks	Include renowned local attractions and must-see sites in the plan.	LLM
Attraction Diversity	Avoid overrepresentation of similar attractions to ensure a varied and engaging itinerary.	LLM
Personal Preference Constraint		
Budget Preference	The plan should align with the user's budget expectations (eg. "premium", "budget-conscious", or "best value").	Rule
Pacing Preference	The plan should reflect the user's desired pacing (eg. "relaxed", "moderate", or "compact").	Rule
Attraction Prioritization	The plan should prioritize or include the user-specified types of attractions.	Rule
Physical Effort Preference	The plan should balance walking distances and physical intensity, matching the user's indicated effort level (eg. "light", "moderate", or "strenuous").	Rule
User Request Fulfillment	The plan should follow the user's specific request.	LLM

Table 7: Constraint description. The format constraints assess whether the plan adheres to the specified format guidelines. The commonsense constraints evaluate the logical feasibility of the plan. The soft and preference constraints measure the alignment of the plan with plan quality standards and user preferences. Constraints in black originate from existing benchmarks, whereas constraints in blue are our newly proposed constraints. Exact formulas and parsing rules for each metric are detailed in Appendix N and Appendix O.

comparisons. This final dataset consists of 267 comparisons between Qwen3-32B and DeepSeek-V3.2, 303 between Qwen3-32B and GPT-5.4, and 898 between GPT-5.4 and DeepSeek-V3.2.

Then we tasked 203 travel specialists with ranking these pairs of travel plans and providing rationales for their choices. For each pair, the specialists evaluated three options, namely favoring route A, favoring route B, or indicating that neither route met satisfactory standards. The specialists are verified professionals who earn their living by designing customized travel plans for clients and possess domain knowledge of specific destinations. We verified their credentials through employment verification with partner travel agencies and required a minimum of 2 years of professional planning experience. All annotators were compensated at professional rates via partner-agency contracts, commensurate with their regular consulting compensation. Each specialist was assigned plans featuring destinations with which they were familiar. Each plan pair was independently evaluated by three expert annotators.

After annotation, we computed inter-annotator agreement using Cohen’s κ for pairwise comparisons and Fleiss’ κ for multi-rater reliability. The results demonstrated moderate agreement among the experts, with Cohen’s κ reaching 0.5421 for pairwise comparisons and Fleiss’ κ achieving 0.5039 for multi-rater reliability. Furthermore, we evaluated inter-rater reliability. The results demonstrate an average pairwise agreement of 71.69% between annotators, while the overall consensus across all three raters stands at 59.00%.

F.2 Grid Search

We employ a grid search optimization to find the optimal weight for our evaluation framework. The search space consists of 13 parameters: 7 soft constraint weights (w_1), 4 preference weights (w_2), and 2 multiplier weights (w_3 , w_4). We discretize the parameter space using coarse grids: $w_1 \in \{0.1, 0.2, 0.4, 0.5, 0.7\}$, $w_2 \in \{0.2, 0.6, 1.0\}$, $w_3 \in \{0.8, 1.0, 1.2\}$, and $w_4 \in \{0.1, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$. The optimization is performed jointly for TripScore-M and TripScore-F tracks, and we set track-specific parameters for different tracks, shown in Table 9. The optimized weight values are provide in Table 9. While some optimal weights lie on the search space boundaries (e.g., $w_4 = 0.1$ for TripScore-M), we found that extending the range (e.g., to 0.05) did

Dimension	Constraints	Ours (Eval)	ChinaTravel (Eval)	TravelPlanner (Eval)	TripTailor (Eval)
Format	Response format	✓(Rule)	✓(Rule)	✓(Rule)	✓(Rule)
	Info verification	✓(Rule)	✓(Rule)	✓(Rule)	✓(Rule)
	Info accuracy	✓(Rule)	✓(Rule)	✓(Rule)	✓(Rule)
	Info relevance	✓(Rule)	✗	✗	✗
Commonsense	Info completeness	✓(Rule)	✓(Rule)	✓(Rule)	✓(Rule)
	Chronological order	✓(Rule)	✓(Rule)	✗	✗
	Location consistency	✓(Rule)	✓(Rule)	✓(Rule)	✓(Rule)
	Operating hours	✓(Rule)	✓(Rule)	✗	✗
	Travel block-out	✓(Rule)	✗	✗	✗
	Transport consistency	✓(Rule)	✓(Rule)	✓(Rule)	✓(Rule)
Soft	Schedule density	✓(Rule)	✗	✗	✗
	Hotel consistency	✓(Rule)	✗	✗	✗
	Daytime utilization	✓(Rule)	✗	✗	✗
	Unique attractions	✓(Rule)	✳(Rule)	✗	✳(Rule)
	Location clustering	✓(Rule)	✳(Rule)	✗	✳(Rule)
	Iconic landmarks coverage	✓(LLM)	✗	✗	✗
Preference	Attraction diversity	✓(LLM)	✗	✗	✗
	Budget style	✓(Rule)	✳(Rule)	✳(Rule)	✳(Rule)
	Pacing style	✓(Rule)	✗	✗	✗
	Attraction prioritization	✓(Rule)	✗	✗	✗
	Physical effort	✓(Rule)	✗	✗	✗
	User request	✓(LLM)	✗	✗	✓(LLM)

Table 8: Dimension-wise coverage of evaluation constraints across benchmarks. Cells indicate whether a constraint is evaluated and by which mechanism (Eval: Rule or LLM). ✓ = evaluated; ✳ = similar constraint exists; ✗ = not evaluated.

not yield statistically significant improvements in expert agreement on the validation set, suggesting the current boundaries are sufficient to capture the relative importance of these factors without overfitting.

We conduct nested stratified 5-fold cross-validation with an inner-loop grid search, yielding validation accuracy 0.6175 ± 0.0259 . A single held-out split gives accuracy 0.6029 with $\tau=0.2071$. On the full 1,468-pair set, $1,000 \times$ bootstrap gives accuracy 0.6031 with a 95% CI of [0.5343, 0.6667] and Kendall’s $\tau=0.2073$ with a 95% CI of [0.0712, 0.3377].

F.3 Sensitive Analysis

Moreover, we probe robustness by jointly scaling the penalty multipliers $(w_3, w_4) \in \{(0.5, 0.05), (1.0, 0.10), (1.5, 0.15), (2.0, 0.20)\}$ and by applying simplex normalization (including temperature smoothing) to the soft and preference constraint weights. Across all variants, train agreement remains in the range 59.40%-62.26% and validation agreement in 58.47%-61.32%, i.e., within ≈ 2.8 pp (train) and ≈ 2.9 pp (val) absolute of the baseline. The near-constant validation accuracy under multiplier scaling and simplex reparameterizations indicates the method is robust to these design choices and does not rely on a narrow hyperparameter configuration.

Parameter	TripScore-M	TripScore-F
w1 (soft constraints)	Schedule Density	0.70
	Hotel Consistency	0.50
	Daytime Utilization	0.40
	Unique Attraction	0.20
	Location Clustering	0.70
	Iconic Landmark	0.10
w2 (preferences)	Attraction Diversity	0.20
	Attraction Prioritization	0.20
	Pacing	0.60
	Budget	0.60
	Physical Effort	0.60
	User Request	-
w3 (soft constraint multiplier)	1.00	1.00
w4 (preference multiplier)	0.10	1.40

Table 9: Optimized weights value. Due to the different evaluation method, minimal-constraint (TripScore-M) and free-form (TripScore-F) datasets share the weights for soft constraints but utilize distinct preference weights.

F.4 Noise-adjusted Analysis

Under a standard K-class symmetric-noise model, we can relate the average human-human pairwise agreement A_{pair} and a typical annotator’s agreement with respect to the latent truth r as follows:

$$A_{\text{pair}} = r^2 + \frac{(1-r)^2}{K-1} \quad (4)$$

$$r = \frac{1 + \sqrt{(K-1)(K \cdot A_{\text{pair}} - 1)}}{K} \quad (5)$$

Where K is the number of classes. With $K = 3$ and $A_{\text{pair}} = 0.7169$, we calculate the human reli-

ability ceiling, $r \approx 0.8390$. Then the expected model-human agreement $A_{\text{model},h}$ relates to the model’s latent-truth agreement r_{model} by:

$$A_{\text{model},h} = r_{\text{model}} \cdot r + \frac{(1 - r_{\text{model}})(1 - r)}{K - 1} \quad (6)$$

Which can be rearranged to solve for r_{model} :

$$r_{\text{model}} = \frac{K \cdot A_{\text{model},h} - A_{\text{model},h} + r - 1}{K \cdot r - 1} \quad (7)$$

Substituting $A_{\text{model},h} = 0.6029$, $K = 3$ and $r \approx 0.8390$ yields $r_{\text{model}} \approx 0.689$. Thus, the raw 60.29% agreement corresponds to a noise-adjusted latent agreement of approximately 68.9%. We can express this as a ratio of the model’s performance to the human reliability ceiling $r_{\text{model}}/r \approx 0.821$. This indicates that the model achieves about **82.1%** of the human reliability ceiling.

As a verification, we can also calculate the predicted three-annotator all-agree rate A_{all} by the calculated human reliability ceiling r :

$$A_{\text{all}} = r^3 + \frac{(1 - r)^3}{(K - 1)^2} \approx 0.594 \quad (8)$$

This closely matches the observed overall agreement of 0.590 between all annotators, providing a sanity check for our calculations.

F.5 Evaluation Comparison Experiments

We implement two LLM-as-judge paradigms to evaluate pairs of candidate travel plans and evaluate agreement with expert labels as well as rank correlation (Kendall’s τ). Both paradigms use a structured rubric of hard/soft constraints and explicitly condition on the user request. To improve reproducibility, we use frozen prompts, temperature 0, and report results across 3 runs. And we report the results for multiple LLMs (GPT-5.4, DeepSeek-V3.2, Gemini-3-flash/pro).

Point-wise scoring. The LLM independently scores each plan in $[0, 100]$ under the same request and rubric. To mitigate instability reported in prior work (Bai et al., 2023), we adopt comparative prompting, anchoring scores with comparative references. The predicted winner is the plan with the higher score, and ties yield a neither decision. We report two point-wise metrics, namely tie-excluded accuracy and tie-inclusive accuracy. The full prompt is provided in Appendix P.6.

Pair-wise comparison. The LLM receives the request and both travel plans, first eliminates candidates with hard-constraint violations, then compares soft quality and preference matching. If still uncertain, it selects the clearer, more executable plan. The model must output exactly one token from route A, route B. The prompt is provided in Appendix P.7.

Ours scoring method. For our hybrid evaluation framework combining rules and LLMs, we apply the identical configuration used in point-wise scoring. Specifically, the plan with the higher score is designated as the winner, while tied scores result in no decision. We employ various LLMs for the LLM-based component. Because tie-excluded and tie-inclusive accuracies are nearly identical, we report only the tie-inclusive accuracy in Table 2.

G Experiment Details

G.1 Fine tuning

SFT. We fine-tune the pretrained model with pair-wise query-response supervision. The SFT training set is constructed by prompting GPT-5.4 to answer queries from the TripScore-M training split. Then we filter out samples that violate our format or commonsense constraints. This yields 2,094 training and 71 validation examples. As GPT-5.4 does not expose an explicit chain-of-thought, we train the Qwen3 base model in a non-thinking configuration. During training, we set the maximum sequence length to 115,000 tokens and training epochs to 3. As the context length is too long, which may lead to GPU out of memory, we set the sequence parallel to 8 to ensure training progress.

RFT (Yuan et al., 2023). We further refine the SFT-tuned Qwen3 models via RFT. Starting from the queries in the TripScore-M training split, we prompt the SFT model to generate five candidate routes per query and score each route with our own evaluator. The highest-scoring route is retained; queries for which all five candidates fail the quality bar are dropped. This curation leaves 2,372 clean training samples. We then fine-tune the SFT model for three epochs on this set, using the same hyperparameters as SFT.

GRPO (Shao et al., 2024b). We train the SFT model based on GRPO algorithm on queries from the TripScore-M training set. Given that the SFT model was trained without a thinking mode, we preserve this setting and continue fine-tuning it using the GRPO algorithm. For each query, the policy

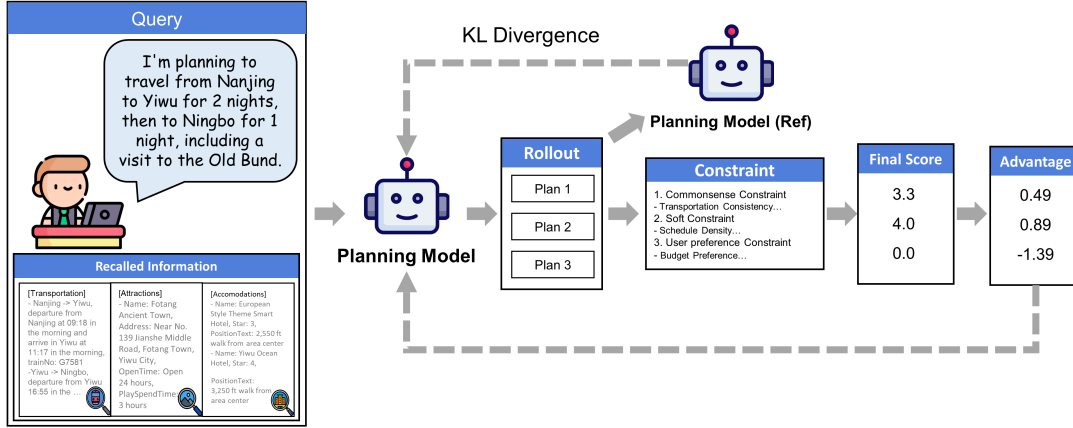


Figure 3: Demonstration of the Grouped Relative Policy Optimization (GRPO). GRPO estimates the baseline from group scores, and normalize them into a standard value. The planning model undergoes iterative optimization by maximizing the objective function $\mathcal{L}(\pi_\theta)$.

model generates 8 rollouts, and rewards are computed by our evaluation framework. To speed up training, we use only the rule-based components of our evaluator. During training process, the maximum prompt length is 79000 and the maximum answer length is 7000.

We perform early stopping by selecting the best-performing checkpoint for each task independently. The hyperparameters employed in fine-tuning baselines are presented in Table 10.

Method	Hyperparameter	Qwen3-8B	Qwen3-14B
SFT	learning rate	2e-5	1e-5
	scheduler type	cosine	cosine
	batch size	32	32
	training epoch	3	3
	warmup ratio	0.1	0.1
RFT	learning rate	2e-5	1e-5
	scheduler type	cosine	cosine
	batch size	32	32
	training epoch	3	3
	warmup ratio	0.1	0.1
GRPO	actor learning rate	1e-6	1e-6
	scheduler type	constant	constant
	batch size	64	64
	training epoch	1	1
	rollout temperature	1.2	1.2
	rollout times	8	8

Table 10: The hyperparameters we employ in baselines.

For training, we use the veRL (Sheng et al., 2025) framework and fine-tune the base models in 32 A100 GPUs. For deployment, we leverage vLLM (Kwon et al., 2023) to serve Qwen3-8B, Qwen3-14B and Qwen3-32B. To maximize inference speed, we set tensor parallelism to 8 and run each model on a single 8 A100 node. All training

experiments were conducted using a fixed random seed of 42 to ensure reproducibility. Due to the high computational cost of fine-tuning and testing, we report results from a single run.

G.2 Other Baselines

Direct. This approach involves inputting the query directly into the model, accompanied by comprehensive instructions detailing the task requirements and all relevant gathered information. The model is expected to generate a response based solely on this input. For Qwen3 base models, we disable the thinking mode for the direct method.

Zero-Shot Chain-of-Thought (ZS-CoT) (Wei et al., 2022). This method enhances the reasoning process by encouraging the model to articulate intermediate steps. Building upon the Direct method, we augment the prompt with the phrase ‘‘Let’s think step by step.’’ This addition is designed to elicit a more detailed, structured reasoning process from the model, potentially leading to more accurate and transparent outcomes. For Qwen3 base models, we enable the thinking mode for the ZS-COT method.

ReAct (Yao et al., 2023). This method incorporates environmental feedback into the planning process. We provide the function FeasibleEnquiry to check day-level feasibility to help enhance the model’s reasoning ability. We develop specialized prompts to guide the LLM in generating daily travel plans and refining them based on the function feedback. Once the model calls the Finish function, the latest plan is returned as the final output. To control the time cost, LLMs are allowed to call the function up to 5 times. Otherwise, it is regarded as failing to generate the plan.

LLM-Modulo (Kambhampati et al., 2024). LLM-Modulo is a hybrid method where a large language model (LLM) generates structured representations from a user’s request, and a symbolic planner verifies them and flags issues in a feedback loop. The method benefits from the bidirectional interaction: The symbolic module actively gives feedback to the LLM, and the LLM refines the itinerary iteratively.

To implement the LLM-Modulo into our benchmark, we develop specialized prompts to guide the LLM in generating initial itineraries and refining them based on evaluator feedback. We reuse our existing constraint evaluators to automatically score itineraries and provide structured feedback to the LLM, and for fairness we only utilize the rule-based part in our evaluation framework. We also add a controller that coordinates iterations between the LLM and the evaluators until constraints are satisfied or a timeout is reached. We set a maximum of 3 iteration steps and stop the refinement process if the reward score exceeds 3.5.

HyperTree (Gui et al., 2025). This baseline employs a hypertree-structured reasoning paradigm that recursively decomposes a travel query into transportation, accommodation, and attraction sub-tasks. By employing this recursive decomposition strategy, the baseline ensures that each aspect of the travel plan can be meticulously crafted and seamlessly integrated, resulting in a highly personalized and adaptable itinerary.

To adapt HyperTree to our dataset, we implement several modifications. We remove all the hypertree rules and nodes related to Dining to align with our schema. We also refactor the hypertree library to handle an arbitrary-length destination list, dynamically generating city-specific subtrees and their corresponding inter-city transportation segments, with no upper bound on the number of cities. Additionally, as the repository does not provide prompts, we write them from scratch based on the descriptions in the paper.

TTG (Ju et al., 2024). TTG (To The Globe) is a hybrid travel-planning system that translates natural language requests into structured JSON constraints using a fine-tuned LLM. It then solves these constraints with a Mixed Integer Linear Programming (MILP) solver to guarantee feasible and near-optimal itineraries in under 5 seconds. It combines the LLMs’ natural language abilities with the mathematical guarantees of MILP solvers.

To implement TTG into our benchmark, we de-

fine a symbolic type to represent feasible travel constraints, serving as the intermediate format for MILP problem generation. Then we translate travel requests into MILP formulations that encode itinerary feasibility and optimization objectives. Finally, we integrate the Python PuLP library to solve the constructed MILP problems using appropriate solver strategies.

NESY (Shao et al., 2024a). The NESY baseline scheme consists of a two-stage process: (1) In the NL2DSL translation stage, natural language queries are converted into logical and preference DSL requirements; (2) In the interactive search stage, a neuro-symbolic solver sequentially arranges activities under the guidance of a symbolic sketch and LLM-driven POI recommendations, generating a multi-day itinerary with DSL validation.

To adapt NESY to our dataset, we implement the following modifications:

- **Dataset Adaptation.** We streamline the planning process by removing redundant logical modules irrelevant to our study, such as restaurants, ensuring the workflow aligns with the available data support.
- **DSL and Concept Function Reconstruction.** We reconstruct the concept functions and augmented the DSL statements with commonsense constraints to match the specific format of our itinerary results. Specifically, we removed DSL statements related to restaurants, people number, dishes, and room types, while expanding commonsense constraints. Ultimately, 32 concept functions were obtained, and the input of each function includes the planned itinerary elements as well as relevant information.
- **Multi-Destination Planning.** To address the limitations of the original NESY framework, which only supported single-destination planning and lacked mechanisms for sequencing multiple destinations, we implement a two-phase modification: (1) For **determining the destination sequence**, we prioritize transportation data (e.g. direct accessibility, travel duration) to determine the planning order. In the absence of such data, we default to the sequence extracted from user input via LLMs. (2) For **day allocation**, we introduce a “cyclic allocation method”: each destination is initially assigned one day, with

additional days distributed sequentially according to the predetermined order until meeting the total day requirement.

- **Transportation Mode Adaptation.** To address the original method’s limitation of deeming planning infeasible when large-scale transportation data (e.g. flights, trains) are unavailable, we implement a more flexible approach. In the absence of such data, we now default to a “self-driving” mode. This adaptation accommodates scenarios like local trips and self-driving tours, maximizing the utilization of POI resources in areas lacking comprehensive transportation data. We establish fixed time windows for self-driving based on typical travel patterns: outbound trips are scheduled from 9:00 to 11:00, and return trips from 18:00 to 20:00.

H Case Study

To examine how plan quality is assessed, we present several head-to-head cases in Figure 4. Although all plans are feasible, side-by-side comparison under our criteria clearly exposes which route is superior. This underscores the need for fine-grained comparative evaluation rather than relying solely on rules or single-pass LLM judgments. Reliable evaluation must account for structural validity, spatio-temporal coherence, semantic value (iconicity and diversity), and user preferences. A practical evaluator should fuse rule-based diagnostics with preference signals, quantify uncertainty, and return actionable feedback.

I Evaluator Ablations and Robustness Protocol

This section presents the results of the robustness protocol described above. We re-scored the outputs of all models from Table 3 using the evaluator variants and analyzed the stability of rankings and conclusions.

To quantify the agreement between different evaluator variants, we compute the system-level Kendall’s τ correlation. To ensure a fair comparison of plan quality independent of feasibility failures, we adopt a pairwise intersection protocol. For each planning method, we compare it against a reference baseline (Direct GPT-5.4) and identify the subset of queries where both methods produced feasible plans (passing format and commonsense checks). We then compute the average score of the target method on this common subset under each

evaluator variant. The Kendall’s τ coefficient is calculated between the system ranking induced by the Full TripScore and the ranking induced by the ablated variant, measuring how well the ablated metric preserves the relative order of model performance on valid plans.

I.1 Ranking Stability Results

Evaluator Variant	TripScore-M (τ)	TripScore-F (τ)
A1. Component Ablation		
Rule-only (Feasibility + Rule Soft)	0.85	-0.07
LLM-only (Feasibility + LLM Soft)	0.60	0.88
A2. Leave-One-Out (LOO)		
LOO-Schedule Density	0.85	0.96
LOO-Hotel Consistency	0.95	0.96
LOO-Daytime Utilization	0.96	0.96
LOO-Unique Attractions	0.97	0.97
LOO-Location Clustering	0.95	0.96
LOO-Iconic Landmarks	0.93	0.99
LOO-Attraction Diversity	0.91	0.84
LOO-Structured Preference	0.47	1.00
LOO-Request Fulfillment	1.00	0.01

Table 11: System-level ranking correlations (Kendall’s τ) between Full TripScore and ablated variants.

Table 11 highlights the necessity of hybrid evaluation. In the minimal-constraint setting (TripScore-M), both rule-based and LLM-based components correlate moderately with the full score (0.85 and 0.60), and removing structured preferences drops correlation to 0.47, indicating a balanced contribution of feasibility rules and preference matching.

I.2 Claim Stability: GRPO vs. GPT-5.4

We further investigated the nature of the improvements brought by RL fine-tuning. While Table 4 compares GRPO against Direct (Qwen3-14B), a more ambitious goal is to bridge the gap with proprietary SOTA models. We compared the conditional quality scores (on feasible plans in TripScore-F) of GRPO (Qwen3-14B) against Direct (GPT-5.4) across evaluator variants.

Evaluator Variant	GRPO (Qwen3-14B)	Direct (GPT-5.4)	Gap
Full TripScore	2.42	2.43	≈ 0.00
Rule-only (Feasibility + Rule Soft)	1.91	1.88	+0.03
LLM-only (Feasibility + LLM Soft)	2.11	2.15	-0.04

Table 12: Average conditional scores on TripScore-F (Generalized) for feasible plans. GRPO (14B) matches GPT-5.4 on the full score, effectively closes the gap by surpassing GPT-5.4 on rule-based execution while remaining competitive on semantic understanding.

As shown in Table 12, GRPO achieves parity with GPT-5.4 on the Full TripScore (2.42 vs 2.43). The decomposition reveals the mechanism: **GRPO outperforms GPT-5.4 on Rule-only metrics** (1.91

Location Clustering	Schedule Density	Hotel Consistency
Query: Plan a laid-back 3-day Yantai trip, departing from Chengde. Good Plan: Day1: Morning: G219 (transportation) -> Evening: Knights-International-Ocean Hotel (hotel) Day2: Morning: Penglai Pavilion (attraction) -> Afternoon: Sanxian Mountain Scenic Area (attraction) -> Evening: Knights-International-Ocean Hotel (hotel) Day3: Morning: Yantai Mountain Scenic Spot (attraction) -> Afternoon: G1092 (transportation) Score: 3.00 Bad Plan: Day1: Morning: G219 (transportation) -> Afternoon: Penglai Pavilion (attraction) -> Evening: Hampton by Hilton (hotel) Day2: Morning: Yangma Island (attraction) -> Evening: Manting Hotel (hotel) Day3: Morning: Sanxian Mountain Scenic Area (attraction) -> Afternoon: G1092 (transportation) Score: 2.91 Reason: The bad plan has unnecessary backtracking: Day 1: Penglai. Day 2: Opposite end. Day 3: Back to Penglai.	Query: I'm planning to stay in London for 6 days and at the meantime i m going to watcha MU game in Manchester. Good Plan: Day1: Morning: SQ308 (transportation) -> Evening: Royal National Hotel (hotel) Day2: Morning: Buckingham Palace (attraction) -> Afternoon: Westminster Abbey (attraction), Big Ben (attraction), Palace of Westminster (attraction) -> Evening: London Eye -> Royal National Hotel (hotel) Day3: Morning: The British Museum (attraction) -> ... Score: 2.98 Bad Plan: Day1: Morning: SQ308 (transportation) -> Afternoon: Royal National Hotel (hotel) -> Evening: Hyde Park (attraction) Day2: Morning: The British Museum (attraction) -> Afternoon: Natural History Museum (attraction) -> Royal National Hotel (hotel) Day3: Morning: Windsor Resort (attraction) -> ... Score: 2.84 Reason: The bad plan is overly relaxed and lacks attraction diversity.	Query: I want to do 5 days 4 night trip at Qingdao. Good Plan: Day1: Morning: CZ8619 (transportation) -> Afternoon: Zhangqiao Park (attraction), St. Michael's Cathedral (attraction) -> Evening: Minguo Hotel (hotel) Day2: Morning: Laoshan Scenic Spot (attraction) -> Evening: Minguo Hotel (hotel) Day3: Morning: Badaguan (attraction) -> Afternoon: Signal Hill Park (attraction), Tsingtao Beer Museum... Score: 2.99 Bad Plan: Day1: Morning: CZ8619 (transportation) -> Afternoon: Badaguan (attraction) -> Evening: Holiday Inn Express (hotel) Day2: Morning: Laoshan Scenic Spot (attraction) -> Evening: Minguo Hotel (hotel) Day3: Morning: Qingdao Wildlife World (attraction) -> Evening: Holiday Inn Express West Coast (hotel) Score: 2.85 Reason: Staying overnight at the zoo on the west coast is a hassle - it's far out and misses the main attractions.

Figure 4: Case studies of the comparison of the plan quality.

vs 1.88), indicating superior adherence to strict constraints (format, operating hours, schedule density). On **LLM-only metrics** (iconic, diversity, request fulfillment), GRPO trails slightly (2.11 vs 2.15) but remains highly competitive. This confirms that RL fine-tuning effectively internalizes complex planning rules, enabling a 14B model to rival GPT-5.4's quality on real-world queries, driven primarily by robust constraint satisfaction.

J Neuro-Symbolic Solver Case Study

The high failure rate of NeSy-based approaches stems from a fundamental mismatch between rigid symbolic logic and the ambiguous, flexible nature of real-world travel planning. While NeSy systems excel at verifiable logic, they often lack the semantic flexibility to handle “soft” constraints, translating vague user intents into overly strict, unexecutable logical predicates. We illustrate this with three specific failure modes observed in our experiments:

J.1 Semantic Misinterpretation

Users frequently request activities that are not discrete physical locations (POIs) in a database, such as “attend a meeting” or “rest.” NeSy translators, lacking semantic grounding, often erroneously map these abstract concepts to rigid POI-visit constraints.

Example Query: “Trip around Beijing. I have a meeting all day tomorrow and want to rest on Thursday”

Generated DSL Constraint:

```
# The solver interprets "Meeting" and "Rest" as
specific POI names to be visited
```

```
result_day2 = (day2_attraction_names_set <=
{'Meeting'})
result_day3 = (day3_attraction_names_set <=
{'Rest'})
```

Failure Cause: The solver attempts to find attractions literally named “Meeting” or “Rest” within the candidate set. Since no such entities exist and the solver cannot produce entities outside the candidate set, it returns “Unsatisfiable.” In contrast, an end-to-end LLM would correctly interpret this as a directive to leave a time slot empty or instantiate a custom event outside the provided candidate set.

J.2 Rigid Entity Matching

Symbolic solvers typically enforce strict set membership or equality checks. If a user provides a generic term (e.g., “Jiuzhaigou hotels”) while the database uses specific official names, the rigid constraint fails to bridge the semantic gap.

Example Query: “Jiuzhaigou 3-day trip. Jiuzhaigou hotels and attractions.”

Generated DSL Constraint:

```
# The solver requires the selected hotel's name
to be strictly "Jiuzhaigou"
result = ({'Jiuzhaigou'} <= hotel_names_set)
```

Failure Cause: Real-world database entries are specific (e.g., “InterContinental Resort Jiuzhaigou Paradise”). The above strict logical condition evaluates to False. The solver lacks the flexibility to perform semantic fuzzy matching or understand that the user implies “any hotel located in Jiuzhaigou.”

J.3 Oversimplified Logic Generation

NeSy translators often struggle to convert complex, implicit constraints (like “overnight train”) into accurate logical predicates, resorting to brittle

heuristics that fail to capture the true condition.

Example Query: “Trip starting from Guangzhou, March 27th-31st. I want to spend the night on the train.”

Generated DSL Constraint:

```
# The solver simplifies "overnight on train" to
a naive time check
overnight_train_found = False
if departure_time >= '23:00' or arrival_time <=
'06:00':
    overnight_train_found = True
result = overnight_train_found
```

Failure Cause: The generated logic is logically flawed or too restrictive. It naively assumes “overnight” means departing after 11 PM or arriving before 6 AM. It misses standard overnight trains that might depart at 8 PM and arrive at 8 AM the next day. Consequently, valid overnight options are filtered out, leading to unnecessary plan failures.

These examples highlight that hard constraints demand a level of precision in prompt-to-logic translation that is inherently brittle in open-domain scenarios. NeSy solvers are blocked by their own logical strictness when facing ambiguity, whereas LLMs can employ soft reasoning to resolve these gaps.

K Code Agent Case Study

This appendix provides the full system prompt used for the Claude Code experiment in Section 5.6. To give the agent harness its best chance, we explicitly modified the system prompt to encourage tool usage:

[Tool Usage] You have access to a code execution environment. For complex planning, constraint checking, budget calculation, or data processing, you may choose to write and execute Python code to assist you. However, using code is optional. You may also reason and generate the itinerary directly. Regardless of the method you choose, your final output must strictly be the requested JSON.

All other settings (the planning task instructions, the input context, and the JSON output schema) are identical to the Direct setting, ensuring that the only difference between the two conditions is the agent harness itself.

L Error Distribution Breakdown

To analyze the pass rate, Figure 5 breaks down violations of format and commonsense constraints on the TripScore-F set. For clarity, we report the top 5

most frequently broken constraints for each method. Furthermore, we found that the consistently dominant errors across all methods stem from format issues related to information accuracy and response structure, along with commonsense violations regarding operating hours and chronological order. Notably, Qwen3-8B exhibits the highest proportion of information-accuracy failures. These typically manifest as hallucinations, such as mismatched attraction IDs and names. Both SFT and GRPO training methodologies demonstrate significant efficacy in mitigating this type of hallucinations.

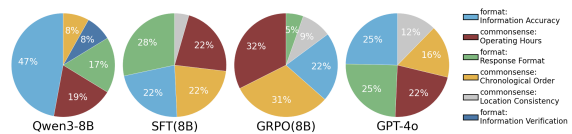


Figure 5: Format and commonsense constraints error distribution on the TripScore-F test set.

M Performance on Various Trip Durations

Figure 6 reports model performance as task complexity increases (longer trip durations). To maintain statistical significance, we focus on results for trips ranging from 1 to 5 days, excluding longer durations with insufficient data points. As the trip duration extends from 1 to 5 days, we observe a notable degradation in performance across vanilla baselines. This decline is evident in both format constraint (Average FPR) and commonsense constraint (Average CPR), with a corresponding decrease in reward. In contrast, GRPO (Qwen3-14B) generally achieves the highest FPR and CPR scores while exhibiting the smallest performance decline as trip duration increases, indicating superior stability in long-horizon planning scenarios. This suggests that GRPO learns to maintain consistency and constraint satisfaction while preserving valid output structure, leading to stronger performance when scheduling becomes more challenging.

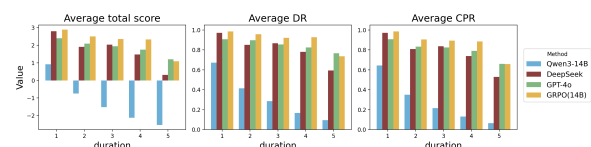


Figure 6: Performance distribution across varying trip durations on the TripScore-M test set.

N Soft Constraint Score Rule

The soft-constraint vector $\mathbf{S}_{\text{soft}} = (S_i)_{i=1}^7$ comprises seven sub-scores defined below: schedule density (S_{schedule}), hotel consistency (S_{hotel}), daytime utilization (S_{daytime}), unique attractions (S_{unique}), location clustering (S_{location}), iconic landmark (S_{iconic}), and attraction diversity ($S_{\text{diversity}}$).

1. Schedule Density. This metric assesses the temporal feasibility of daily itineraries by checking minimum and maximum activity hours against day-specific thresholds. Scoring is by day-level violation (at most one penalty per day):

$$S_{\text{schedule}} = 1 - \frac{|d_v|}{D} \quad (9)$$

where D is the total number of days. d_v is the set of days where the total activity hours exceed the upper bound (10 hours) or fall below the lower bound (4 hours).

2. Hotel Consistency. The hotel consistency metric applies penalties to unnecessary hotel changes within a single city:

$$S_{\text{hotel}} = 1 - \frac{|S_{\text{switches}}|}{|H|} \quad (10)$$

where S_{switches} represents the number of nights involving a switch to a nearby hotel (within 100 km) in the same city, and $|H|$ is the total number of hotel nights.

3. Daytime Utilization. This constraint ensures the efficient utilization of daytime hours, with a focus on avoiding idle time or overcrowded scheduling:

$$S_{\text{daytime}} = 1 - \frac{\sum_{d=1}^D \mathbf{1}[\text{violation}_d]}{D} \quad (11)$$

where violation_d occurs when there is not arranged any activities in the morning and afternoon.

4. Unique Attractions. The uniqueness score employs a combination of linear and exponential penalties to quantify the degree of redundancy among selected attractions:

$$S_{\text{unique}} = \max \left(0, 1 - \frac{|A_{\text{dup}}|}{|A_{\text{total}}|} - \sum_{a \in A_{\text{dup}}} \frac{(n_a - 1)^2 \cdot 0.05}{|A_{\text{total}}|} \right) \quad (12)$$

where A_{dup} represents non-consecutively duplicated attractions. n_a is the number of times an activity appears. A_{total} represents the total attractions in the itinerary.

5. Location Clustering. This metric optimizes the spatial distribution of daily activities by applying penalties proportional to inter-attraction distances:

$$S_{\text{location}} = \max \left(0, 1 - \frac{\mathcal{P}_{\text{total}}}{|A_{\text{total}}|} \right) \quad (13)$$

where $\mathcal{P}_{\text{total}}$ represents the count of consecutive activity pairs for the same day that are among the top 20% farthest of all activity pairs in the itinerary. $|A_{\text{total}}|$ is the total number of activities.

6. Iconic Landmark and Attraction Diversity. These two metrics are assessed using a 5-point Likert scale, with evaluations conducted by LLMs. The resulting scores are normalized to the range $[0, 1]$ so that they are comparable to other soft sub-scores. The corresponding prompts are provided in Appendix P.2 and Appendix P.1. The normalization applied to both metrics is:

$$S = \frac{\text{rating} - 1}{4}, \quad \text{rating} \in \{1, 2, 3, 4, 5\}. \quad (14)$$

This mapping ensures that the lowest rating (1) corresponds to a normalized score of 0, while the highest rating (5) yields a normalized score of 1.

O Preference Constraint Score Rule

TripScore-M Datasets. The preference evaluator measures how well an itinerary aligns with the user's stated preferences. We consider four dimensions: $\mathbf{S}_{\text{pref}} = (S_{\text{budget}}, S_{\text{pacing}}, S_{\text{attraction}}, S_{\text{effort}})$.

1. Budget Preference (S_{budget}): The budget preference metric calculates the proportion of consistency between users' budget-corresponding expected hotel star ratings and actual hotel star ratings, aiming to measure the matching degree of budget and hotel star:

$$S_{\text{budget}} = \frac{1}{|H|} \sum_{h \in H} g(h; \text{pref}). \quad (15)$$

where H is the set of hotel nights, and $g(h; \text{pref}) \in \{0, 1\}$ indicate whether night h matches the budget preference (e.g., 0-2 stars for "cost-effective", 3-4 stars for "comfortable" and 5 stars for "High-end").

2. Pacing Preference (S_{pacing}): Pacing Preference quantifies the consistency between the actual itinerary arrangement and users' pace preferences based on key indicators such as play duration and the number of activities:

$$S_{\text{spacing}} = \frac{1}{D} \sum_{d=1}^D f(d; \text{pref}). \quad (16)$$

where D is the number of days and $f(d; \text{pref}) \in [0, 1]$ is a day-level compliance function. Specifically, $f(d; \text{pref}) = 1$ if the daily activity time is within the preferred range (e.g., 6-8 hours for "Moderate"), $f(d; \text{pref}) = 0.5$ if it deviates by less than 2 hours, and 0 otherwise.

3. **Attraction Preference ($S_{\text{attraction}}$):** We map user attraction preferences to POI tags and measure coverage among visited POIs. It is calculated by the proportion of qualified attraction:

$$S_{\text{attraction}} = \frac{1}{|A_{\text{total}}|} \sum_{a \in A_{\text{total}}} m(a; \text{pref}). \quad (17)$$

where A_{total} represents all visited attractions and $m(a; \text{pref}) \in \{0, 1\}$ indicating whether attraction a matches any preferred tag.

4. **Physical Effort Preference (S_{effort}):** Calculates activity's physical effort value by rules, determines itinerary exertion type via high-effort activity and duration ratios, and quantifies consistency with users' physical effort preferences.

$$S_{\text{effort}} = \frac{1}{D} \sum_{d=1}^D h(d; \text{pref}). \quad (18)$$

where $h(d; \text{pref}) \in \{0, 1\}$ indicate whether the day d matches the preferred physical effort tag. A day is labeled strenuous if (i) its single-day physical-exertion score is greater than 2, where hiking, theme-park and mountain-climbing each contribute 1, cycling contributes 2, and all other activities contribute 0; or (ii) it requires physical exertion on two consecutive days. A day is light if it involves no exertion and neither adjacent day involves exertion. All remaining cases are moderate.

TripScore-F Datasets. For real-world datasets, we employ LLM-based evaluation to assess overall user request compliance. $S_{\text{pref}} = \{S_{\text{user}}\}$. The corresponding prompt is provided in Appendix P.3.

$$S_{\text{user}} = \alpha \text{LLM}(\text{prompt}, \text{response}, \text{user_request}) \quad (19)$$

where α represents the scaling weight (0.2) employed to normalize the score within the range of 0 to 1.

P Prompt List

P.1 Iconic Landmarks Evaluation Prompt

ICONIC_LANDMARKS_EVALUATION_PROMPT

Please evaluate whether the attractions in the following itinerary cover the classic must-visit attractions of corresponding destination.

Itinerary: {answer_text}

Please evaluate based on the following criteria:

1. Does it include the most famous landmark attractions of the destination.
2. Does it cover different types of classic attractions (historical culture, natural scenery, modern architecture, etc.).
3. The popularity and recommendation level of the attractions.
4. Please consider the number of days in the itinerary. If some secondary attractions cannot be covered due to insufficient days, you can relax the evaluation criteria.

Please return the evaluation result in JSON format:

```
{
  "score": score (integer rating 1-5, where 1=no classic attractions, 2=only a few classic attractions, 3=some classic attractions, 4=most classic attractions, 5=all classic attractions),
  "missing_attractions": ["list of missing important classic attractions"],
  "explanation": "detailed explanation"
}
```

P.2 Attraction Diversity Evaluation Prompt

ATTRACTION_DIVERSITY_EVALUATION_PROMPT

Please evaluate the richness and diversity of the following itinerary to determine if there are homogenization issues.

Itinerary: {answer_text}

Please evaluate based on the following criteria:

1. Diversity of attraction types (historical culture, natural scenery, entertainment, shopping & dining, etc.).
2. Richness of activity experiences (sightseeing, hands-on experiences, interactive, leisure, etc.).
3. Reasonable pace arrangement (balance of active/quiet, indoor/outdoor).
4. Avoiding repetitive or homogeneous activities.
5. Please consider the main attraction types of the destination. If the main attractions of the destination are of a single type, you can relax the evaluation criteria for homogeneity issues.

Please return the evaluation result in JSON format:

```
{
  "score": score (integer rating 1-5, where 1=homogenization problem accounts for more than 80% of the itinerary, 2=homogenization problem accounts for about 60% of the itinerary, 3=homogenization problem accounts for about 40% of the itinerary, 4=homogenization problem accounts for about 20% of the itinerary, 5=homogenization problem is small or nonexistent),
  "diversity_issues": ["list of identified homogenization or monotony issues"],
  "explanation": "detailed explanation"
}
```

P.3 User Request Fulfillment Evaluation Prompt

USER_PREFERENCE_CONSTRAINT_PROMPT

You are a professional travel itinerary evaluation expert, responsible for evaluating whether the generated itinerary meets the user's specific request and expectations. Please evaluate the following travel itinerary based on the assessment criteria to determine whether it meets the user's specific request and expectations.

****User Request: ****
{user_request}

****Generated Itinerary Response: ****
{answer_text}

You need to carefully analyze the user's requirements and evaluate the itinerary's alignment based on the following aspects: Departure/Destination, Schedule/Timing, Mode of Transportation, Number of Travelers, Accommodation Requirements, Coverage of Attractions, Activity Types, Pace of the Trip, Budget, Other Requirements.

****Scoring Criteria****

5 points: Excellent. The itinerary fully meets all the user's requirements and considers potential personalized needs, providing a travel plan that exceeds expectations.

4 points: Good. The itinerary fully meets all the user's core requirements; however, there are details that could be further optimized.

3 points: Average. The itinerary satisfies most user requirements, such as mandatory budget, schedule, and number of travelers, but some aspects are not adequately addressed.

2 points: Poor. The itinerary fails to meet the user's main requirements, with most elements misaligned with their preferences.

1 point: Very Poor. The itinerary completely fails to meet the user's expectations and is irrelevant to their request.

0 points: The user did not provide any specific information (e.g., "Plan a trip for me"), in which case any itinerary offered can be considered as meeting the user's needs.

****Instructions for Scoring****

1. Your evaluation should focus on determining whether the provided itinerary meets the user's expectations.

2. If IDs are provided for transportation, POIs, or hotels, you may assume these details are authentic and reliable.

3. Before assigning a score, analyze the itinerary and the user's request, explaining why you assigned that score.

4. If the user's request changes midway, base your evaluation on the latest requirements.

5. You only need to evaluate the current itinerary. If the user requests multiple or alternative options, this should not result in a deduction.

6. Strictly follow the JSON format below when providing the evaluation result

Output format:

```
{
  "detailed_feedback": "Detailed evaluation feedback",
  "final_score": Final score (0-5)
}
```

P.4 Itinerary Generation Prompt

ITINERARY_GENERATION_PROMPT

You are a travel planning expert, skilled at generating detailed travel itineraries based on user's needs and preferences, and ultimately outputting them in JSON format. Attention: User's requirements may change, you need to adapt to the latest query of the user.

[Itinerary Arrangement Rules]

1. Arrange the itinerary according to the user's requirements. Don not change the user's travel plan, including the number of days, travel cities, travel dates and etc.

2. The itinerary should be arranged in chronological order, and the time period should be divided into Morning/Afternoon/Evening.

3. A well-designed itinerary should include transportation arrangements, accommodation and key attractions, all organized in a proper chronological sequence to ensure a smooth travel experience. Pay special attention to the restrictions on the opening hours of attractions (openTime-Calendar field) to avoid scheduling visits during times when the attractions are closed or not allowing entry.

4. For transportation, prioritize choosing main stations and pay special attention to the timing of transportation arrangements, ensuring they are scheduled within intended time period. Avoid mismatches such as planning morning departures for afternoon or evening time period.

5. For accommodation, find the most suitable hotel from the most suitable hotel area. One city only need one hotel and do not change hotel in the same city.

6. The hotel should be arranged at the end of the day, and hotel arrangements should be indicated every night, except on the last day.

7. It is especially important to pay attention to the time requirements for attractions and transportation to avoid time conflicts, which could lead to an unreasonable or unachievable itinerary.

8. Ensure that the travel schedule and physical exertion are moderate, and avoid arranging too many activities in the same time period.

9. Transportation for the outbound and return trips is required, and be careful not to mix up the outbound and return trips.

10. If the user's travel duration is short (less than 2 days), it is recommended to focus on visiting the core attractions within the city.

[Reference Data Rules]

Provided reference data may include:

Attractions: poi (id: poiId, name: poiName)

Hotels: hotel (id: hotelid, name: hotelname)

Transportations: train/flight/bus/driving/ferry/ship (id: planid, name: trainNo/flightNo/shipName)

you must select hotels and transportation from reference data. But attractions are allowed to use external resources if more suitable.

[Json Format Instruction]

1. Extract exact name, type, and id when using reference data:

For attractions type: 'poi', name: poiName, id: poiId.

For hotels type: 'hotel', name: hotelname, id: hotelid;

For transportations type: 'transportation', name: trainNo/flightNo, id: planid.

2. Attractions which is not in the reference data are allowed, but must set "id": "" and when mentioning the attraction in the description, use the format ****attraction****. transportation and hotel must be chosen from the reference

data.

3. for the items has the same id, just output one item, do not repeat items with different name but same id.

4. the Json format is as follows:

```
{
  "itineraryName": "itinerary name like: 3 Days's Travel Itinerary: Shanghai to Beijing",
  "recommendReason": "the reason why this itinerary is recommended, and make user feel that this itinerary is very suitable for him/her requirements. recommend reason should be no more than 50 words",
  "dayInfos": [
    {
      "day": "the order of days,a integer number starting from 1",
      "scheduleTitle": "today's schedule title",
      "scheduleDetail": [
        {
          "period": "the time period when the schedule begins, must choose one from Morning/Afternoon/Evening(Capitalized Initial Letter)",
          "description": "Mention all attractions/hotels/transportations using the specified markdown syntax: For attractions and hotels,you should point out names and ids, use the format **[PoiName](poiId)** or **[HotelName](hotelId)** , if the attractions is not in the reference data, use the format **[PoiName]**."
        }
      ],
      "detailList": [
        {
          "type": "transportation/poi/hotel",
          "id": "planid/poiId/hotelid",
          "name": "trainNo/flightNo/poiName/hotelname"
        }
      ]
    }
  ],
  "tips": {
    "title": "tips title",
    "info": "the tips's total content should be within 50 words"
  }
}
```

[References]

[transportation arrangements]
{transportation_information}

[attractions reference information]
{attraction_information}

[hotels reference information]
{hotel_information}

[User Query]
{user_request}

P.5 ReAct Prompt

REACT_PROMPT

You are a travel planning expert, skilled at producing a detailed itinerary based on the given transportation scheme and the user's requirements and preferences.

Please complete the task by alternating between Thought, Action, and Observation steps.

The Thought phase is used to reason about the current situation. The Action phase has two types:

(1) FeasibleEnquiry [Sub Plan]: used to assess whether a sub-plan is basically feasible. You must input the sub-plan in JSON format. The sub-plan should cover a full day.

(2) Finish [Final Plan]: used to indicate task completion. You must submit the complete and final plan (in JSON format) as the argument.

[Itinerary Arrangement Rules]

1. Arrange the itinerary according to the user's requirements. Don not change the user's travel plan, including the number of days, travel cities, travel dates and etc.

2. The itinerary should be arranged in chronological order, and the time period should be divided into Morning/Afternoon/Evening.

3. A well-designed itinerary should include transportation arrangements, accommodation and key attractions, all organized in a proper chronological sequence to ensure a smooth travel experience. Pay special attention to the restrictions on the opening hours of attractions (openTime-Calendar field) to avoid scheduling visits during times when the attractions are closed or not allowing entry.

4. For transportation, prioritize choosing main stations and pay special attention to the timing of transportation arrangements, ensuring they are scheduled within intended time period. Avoid mismatches such as planning morning departures for afternoon or evening time period.

5. For accommodation, find the most suitable hotel from the most suitable hotel area. One city only need one hotel and do not change hotel in the same city.

6. The hotel should be arranged at the end of the day, and hotel arrangements should be indicated every night, except on the last day.

7. It is especially important to pay attention to the time requirements for attractions and transportation to avoid time conflicts, which could lead to an unreasonable or unachievable itinerary.

8. Ensure that the travel schedule and physical exertion are moderate, and avoid arranging too many activities in the same time period.

9. Transportation for the outbound and return trips is required, and be careful not to mix up the outbound and return trips.

10. If the user's travel duration is short (less than 2 days), it is recommended to focus on visiting the core attractions within the city.

[Reference Data Rules]

Provided reference data may include:

Attractions: poi (id: poiId, name: poiName)

Hotels: hotel (id: hotelid, name: hotelname)

Transportations: train/flight/bus/driving/ferry/ship (id: planid, name: trainNo/flightNo/shipName)

you must select hotels and transportation from reference data. But attractions are allowed to use external resources if more suitable.

[Complete Plan JSON Specification]

1. Extract exact name, type, and id when using reference data:

For attractions type: 'poi', name: poiName, id: poiId.
 For hotels type: 'hotel', name: hotelname, id: hotelid;
 For transportations type: 'transportation', name: trainNo/flightNo, id: planid.

2. Attractions which is not in the reference data are allowed, but must set "id": "" and when mentioning the attraction in the description, use the format **attraction**.

transportation and hotel must be chosen from the reference data.

3. for the items has the same id, just output one item, do not repeat items with different name but same id.

4. the Json format is as follows:

```
{
  "itineraryName": "itinerary name like: 3 Days's Travel Itinerary: Shanghai to Beijing",
  "recommendReason": "the reason why this itinerary is recommended, and make user feel that this itinerary is very suitable for him/her requirements. recommend reason should be no more than 50 words",
  "dayInfos": [
    {
      "day": "the order of days,a integer number starting from 1",
      "scheduleTitle": "today's schedule title",
      "scheduleDetail": [
        {
          "period": "the time period when the schedule begins, must choose one from Morning/Afternoon/Evening(Capitalized Initial Letter)",
          "description": "Mention all attractions/hotels/transportations using the specified markdown syntax: For attractions and hotels,you should point out names and ids, use the format **[PoiName](poiId)** or **[HotelName](hotelId)**; if the attractions is not in the reference data, use the format **[PoiName]**.",
          "detailList": [
            [
              {
                "type": "transportation/poi/hotel",
                "id": "planid/poiId/hotelid",
                "name": "trainNo/flightNo/poiName/hotelname"
              }
            ]
          ]
        }
      ]
    }
  ],
  "tips": {
    "title": "tips title",
    "info": "the tips's total content should be within 50 words"
  }
}
```

***** Example *****

Input: I want to travel to Beijing, departing from Shanghai, for 3 days. Please design a detailed travel itinerary.
 You may call FeasibleEnquiry like FeasibleEnquiry[{
 "day": 1,
 "scheduleTitle": "Arrivel in Beijing and begins the tour with Beijing Zoo",
 "scheduleDetail": [
 {
 "period": "Morning",
 "description": "Take the train **T110** from Shanghai to Beijing, the train duration is approximately 6 hours",

```
    "detailList": [
      {
        "type": "transportation",
        "id": "1001",
        "name": "T110"
      }
    ],
    {
      "period": "Afternoon",
      "description": "Exploring
**[Beijing Zoo](0001)** and experience the vitality of the wildlife.",
      "detailList": [
        {
          "type": "poi",
          "id": "0001",
          "name": "Beijing Zoo"
        }
      ]
    },
    {
      "period": "Evening",
      "description": "Stay in the Tiananmen Square Area for its proximity to magor attractions. Recommend
**[Beijing Tiantan Manssion Hotel](0002)** for its convenient transportation",
      "detailList": [
        {
          "type": "hotel",
          "id": "0002",
          "name": "Beijing Tiantan Manssion Hotel"
        }
      ]
    }
  ]
}
```

You may call Finish like Finish[{
 "itineraryName": "3 Days's Travel Itinerary: Shanghai to Beijing",
 "recommendReason": "Based on your request, with just three days, traveling from Shanghai to Beijing allows you to experience both cultural landmarks and modern attractions. I recommend this itinerary to explore Beijing's highlights at a comfortable pace.",
 "dayInfos": [
 "day": 1,
 "scheduleTitle": "Arrivel in Beijing and begins the tour with Beijing Zoo",
 "scheduleDetail": [
 {
 "period": "Morning",
 "description": "Take the train **T110** from Shanghai to Beijing, the train duration is approximately 6 hours",
 "detailList": [
 {
 "type": "transportation",
 "id": "1001",
 "name": "T110"
 }
],
 },
 {
 "period": "Afternoon",
 "description": "Exploring
[Beijing Zoo](0001) and experience the vitality of the wildlife.",
 "detailList": [
 {
 "type": "poi",
 "id": "0001",
 "name": "Beijing Zoo"
 }
],
 },
 {
 "period": "Evening",
 "description": "Stay in the Tiananmen Square Area for its proximity to magor attractions. Recommend
[Beijing Tiantan Manssion Hotel](0002) for its convenient transportation",
 "detailList": [
 {
 "type": "hotel",
 "id": "0002",
 "name": "Beijing Tiantan Manssion Hotel"
 }
],
 }
],
 "tips": {
 "title": "tips title",
 "info": "the tips's total content should be within 50 words"
 }
 }
]
 }

```

{
  "type": "poi",
  "id": "0001",
  "name": "Beijing Zoo"
}
],
{
  "period": "Evening",
  "description": "Stay in the Tiananmen Square Area for its proximity to major attractions. Recommend **[Beijing Tiantan Manssion Hotel](0002)** for its convenient transportation",
  "detailList": [
    {
      "type": "hotel",
      "id": "0002",
      "name": "Beijing Tiantan Manssion Hotel"
    }
  ]
},
{
  "day": 2,
  "scheduleTitle": "Explore Universal Beijing Resort and rediscover the joys of childhood ",
  "scheduleDetail": [
    {
      "period": "Morning",
      "description": "After breakfast, take bus to **[Universal Beijing Resort](0003)** which is a large theme park resort located in Tongzhou District, including the Universal Studios Beijing theme park, two resort hotels, and a comprehensive commercial area",
      "detailList": [
        {
          "type": "poi",
          "id": "0003",
          "name": "Universal Beijing Resort"
        }
      ]
    }
  ],
  "day": 3,
  "scheduleTitle": "Cultural Immersion and Departure",
  "scheduleDetail": [
    {
      "period": "Morning",
      "description": "Visit **[temple of Heaven](0004)**: Explore this UNESCO World Heritage Site, where emperors prayed for good harvests",
      "detailList": [
        {
          "type": "poi",
          "id": "0004",
          "name": "temple of Heaven"
        }
      ]
    }
  ],
  "tips": {
    "title": "Tips for Enhanced Travel Experience",
    "info": [

```

```

    "Use a private car or join a guided tour for the Universal Beijing Resort",
    "Book tickets in advance to secure your entry",
    "Pack comfortable walking shoes"
  ]
}
]

```

***** End of Example *****

You must call Finish to indicate you have completed the task. Each action must call exactly one function; do not call multiple functions at the same time. Please make sure call Finish before 5 rounds of dialogue.

[References]
[transportation arrangements]
{transportation_information}

[attractions reference information]
{attraction_information}

[hotels reference information]
{hotel_information}

[User Query]
{user_request}

P.6 Point-wise evaluation Prompt

POINT_WISE_EVALUATION_PROMPT

You are a travel itinerary quality reviewer. Please rate a single itinerary based on the following criteria (0-100), without introducing external information or speculation.

[Evaluation Criteria] (in order of priority from high to low)

1. Format and Facts (hard constraints, severe violations directly Inferior)

Response structure: The output must strictly follow the requested schema. Missing or misplaced elements are non-compliant.

Information verification: Transportation/hotels/attractions must come from the given text; introducing external facts or conjecture is treated as hallucination and deemed invalid.

Information accuracy: Details such as names/times are consistent;

Information relevance: Description matches corresponding attractions/events.

2. Common Sense and Feasibility (hard constraints)

Complete information: Each destination must include necessary accommodation, essential transportation, and key activities to ensure executability.

Correct time sequence: Activities must be listed in temporal order; days cannot backtrack, and intra-day sequences must be non-decreasing in time.

Location consistency: A traveler cannot be in multiple cities/locations simultaneously; any change of location must be justified by an explicit transport step.

Feasible operating hours: Visits must occur within confirmed opening hours; closed days/times invalidate scheduled activities.

Transportation block: No activities scheduled during transport intervals;

Early transportation rule: If departure time is before 10 AM, no earlier activities scheduled that day;

Transportation continuity: Smooth movement between cities/attractions, no repeated backtracking.

3. Soft Constraints

Moderate pace density: Daily pacing should be balanced—neither overpacked nor overly sparse—with reasonable buffers for transition and rest.

Hotel Consistency: Within the same city, prefer a single hotel to minimize check-in/out overhead and travel friction.

Daytime Utilization: Prioritize activities during daylight; reserve evenings for appropriate activities or rest, avoiding unproductive daytime gaps.

Unique Attractions: Avoid repeated visits to the same (or effectively identical) attractions.

Location Clustering: Group nearby attractions to reduce transit time and improve route efficiency.

Iconic Landmarks: When feasible, include representative, must-see landmarks to improve coverage and recognizability.

Attraction Diversity: Maintain variety across categories (e.g., cultural, natural, museums, landmarks) to avoid monotony.

4. Preference Matching (only considered if preferences appear in the text, otherwise treated neutrally)

Budget Preference: Select hotels/activities aligned with the stated budget profile (e.g., premium, budget-conscious, value-oriented).

Pacing Preference: Match the requested pacing (relaxed, moderate, compact) by adjusting daily activity counts and durations.

Attraction Prioritization: Prioritize categories explicitly favored by the user and ensure requested items are covered.

Physical Effort Preference: Align walking distance and intensity with the specified level (light, moderate, strenuous), managing elevation and high-exertion activities accordingly.

User Request Fulfillment: Satisfy explicit user constraints (e.g., must-visit/avoid, time windows, ordering). If no preferences are stated, no penalty or credit is applied.

[Scoring Approach]

First apply compliance deductions based on hard constraints, then provide a total score considering soft constraints and preference matching.

[Scoring Anchors]

90-100: Comprehensive, factually accurate, highly actionable, well-paced, and strongly aligned with preferences.

70-85: Largely complete, with occasional minor flaws that do not impede execution or user experience.

50-65: Moderate quality; contains several issues but remains executable.

30-45: Significant flaws (e.g., temporal/spatial conflicts, missing elements) requiring substantial revision.

0-25: Numerous severe issues or a large amount of fabricated/irrelevant information; largely unusable/inactionable.

[Output Requirement]

Strictly output 'score' (0-100), no explanations or additional text.

[User Query]

{user_request}

[Itinerary]
{itinerary}

P.7 Pair-wise evaluation Prompt

PAIR_WISE_EVALUATION_PROMPT

You are a travel itinerary quality reviewer. Your task is to compare two candidate itineraries under strict evaluation criteria, without introducing external information or speculation.

[Evaluation Criteria] (in order of priority from high to low)

1. Format and Facts (hard constraints, severe violations directly Inferior)

Response structure: The output must strictly follow the requested schema. Missing or misplaced elements are non-compliant.

Information verification: Transportation/hotels/attractions must come from the given text; introducing external facts or conjecture is treated as hallucination and deemed invalid.

Information accuracy: Details such as names/times are consistent;

Information relevance: Description matches corresponding attractions/events.

2. Common Sense and Feasibility (hard constraints)

Complete information: Each destination must include necessary accommodation, essential transportation, and key activities to ensure executability.

Correct time sequence: Activities must be listed in temporal order; days cannot backtrack, and intra-day sequences must be non-decreasing in time.

Location consistency: A traveler cannot be in multiple cities/locations simultaneously; any change of location must be justified by an explicit transport step.

Feasible operating hours: Visits must occur within confirmed opening hours; closed days/times invalidate scheduled activities.

Transportation block: No activities scheduled during transport intervals;

Early transportation rule: If departure time is before 10 AM, no earlier activities scheduled that day;

Transportation continuity: Smooth movement between cities/attractions, no repeated backtracking.

3. Soft Constraints

Moderate pace density: Daily pacing should be balanced—neither overpacked nor overly sparse—with reasonable buffers for transition and rest.

Hotel Consistency: Within the same city, prefer a single hotel to minimize check-in/out overhead and travel friction.

Daytime Utilization: Prioritize activities during daylight; reserve evenings for appropriate activities or rest, avoiding unproductive daytime gaps.

Unique Attractions: Avoid repeated visits to the same (or effectively identical) attractions.

Location Clustering: Group nearby attractions to reduce transit time and improve route efficiency.

Iconic Landmarks: When feasible, include representative, must-see landmarks to improve coverage and recognizability.

Attraction Diversity: Maintain variety across categories (e.g., cultural, natural, museums, landmarks) to avoid monotony.

4. Preference Matching (only considered if preferences appear in the text, otherwise treated neutrally)

Budget Preference: Select hotels/activities aligned with the stated budget profile (e.g., premium, budget-conscious, value-oriented).

Pacing Preference: Match the requested pacing (relaxed, moderate, compact) by adjusting daily activity counts and durations.

Attraction Prioritization: Prioritize categories explicitly favored by the user and ensure requested items are covered.

Physical Effort Preference: Align walking distance and intensity with the specified level (light, moderate, strenuous).

User Request Fulfillment: Satisfy explicit user constraints (e.g., must-visit/avoid, time windows, ordering).

[Decision]

First pay attention to hard constraints, severe violations are inferior; if both comply, then compare soft constraints and preference matching. If difficult to distinguish, choose the clearer and more executable one.

[Output Requirement]

Only output "Route A" or "Route B", no other characters or explanations.

[User Query]

{user_request}

[Route A]

{route_A}

[Route B]

{route_B}