COORDINATING DOMAIN-SPECIALIZED LLM AGENTS FOR MULTI-CITY TRAVEL PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-city travel planning is a tightly coupled, multi-scale decision problem requiring mobility, accommodation, and activity choices to remain jointly feasible under temporal, spatial, budget, and user-specific constraints. We propose a structured agentic framework that explicitly coordinates these interdependent decisions while separating semantic reasoning from deterministic execution. *Domain-specialized* LLM agents select from grounded candidates and encode their choices, alternatives, evidence, and unresolved dependencies as decision contracts. A *Cross-Domain Coordinator* first provides global planning guidance and then reconciles these contracts into a joint contract under explicit authority constraints. The joint contract bounds downstream city-day composition and solver-assisted temporal-spatial scheduling, while deterministic controls validate decisions before execution and verify the materialized itinerary afterward. Detected violations are routed to the responsible decision scope for bounded revision and re-execution. The framework achieves Final Pass Rates of 74.40% on TravelPlanner and 59.44% on ChinaTravel. Ablations show that cross-domain coordination is critical for coupled multi-city decisions, whereas specialist LLM reasoning is most beneficial for semantically rich city-day planning.

1 INTRODUCTION

Multi-city travel planning requires interdependent decisions about city order, length of stay, intercity transportation, accommodation, activities, dining, and local movement. Transport schedules determine the time available in each city; accommodation location affects local travel; and activities must fit opening hours, travel times, and the overall budget. A choice that is suitable within one domain can therefore make the complete itinerary infeasible when combined with other choices. TravelPlanner tests whether decisions across a complete trip satisfy shared constraints, while ChinaTravel tests whether detailed itineraries also meet open-ended requests and local feasibility conditions (Xie et al., 2024; Shao et al., 2026). Together, they expose a planning challenge across two scales: trip-level choices set the boundaries for daily activities, but the feasibility of those activities may only become clear during detailed scheduling.

Prior work addresses parts of this challenge through different planning structures. Multi-phase planning and PMC structure the task into stages or dependent subtasks (Xie & Zou, 2024; Zhang et al., 2025). ATLAS uses checking and targeted search to revise proposed plans, while HiMAP-Travel coordinates day planners through shared constraint monitoring and feasibility feedback (Choi et al., 2026; Bui et al., 2026). More closely related to our setting, TRIPPULSE combines travel-domain agents, global orchestration, and scheduling, but fixes transportation choices before daily activities are scheduled (Karmakar et al., 2026). Hybrid systems use formal planners or solvers to enforce represented constraints (de la Rosa et al., 2024; Hao et al., 2025; Shao et al., 2025).

These approaches show the value of decomposition, coordination, and feasibility checking. A further difficulty arises when scheduling reveals that an earlier domain choice must change. For example, a selected train may reach a city too late for a required visit. Rearranging activities cannot recover the lost time, while choosing an earlier train may alter the budget and other parts of the trip. Resolving the conflict requires more than identifying an invalid schedule: the planner must know which component owns the transport choice, which alternatives are supported by evidence, and which dependent decisions must be checked again.

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

To address this challenge, we propose an agentic framework that keeps grounded travel decisions verifiable through selection, construction, and revision. A Cross-Domain Coordinator first provides trip-level guidance; domain Specialists then propose selected options, supported alternatives, evidence, and unresolved dependencies. The Coordinator reconciles these proposals into a joint decision contract that records approved choices and bounds changes by grounded options and component authority. After the deterministic Hard Monitor authorizes the contract, the LLM-based City-day Planner assigns approved activities and dining options to city-days and proposes their relative order. The deterministic Contract Executor then constructs a timed, routed itinerary, which an independent Verifier checks. Detected violations are routed to the responsible planning component for bounded revision, followed by renewed authorization, composition, execution, and verification. Our key contributions are:

- **Evidence-backed cross-domain coordination.** Domain Specialists record selected candidates, supported alternatives, evidence, and unresolved dependencies in decision contracts. A Coordinator provides guidance before selection and reconciles the resulting decisions afterward, subject to explicit authority constraints.
- **Contract-bounded construction and revision.** An LLM composes authorized activities into city-days, while a deterministic executor schedules them. Separate checks authorize decisions before construction and verify the completed itinerary; violations are routed to the component responsible for revising the affected choice.
- **Evaluation across planning scales.** The framework achieves a 74.40% Final Pass Rate on the official TravelPlanner Test1000 and 59.44% on a study-constructed ChinaTravel Test180 subset. Ablations show that cross-domain coordination is particularly important on TravelPlanner, whereas Specialist LLM reasoning is more valuable for ChinaTravel's detailed activity planning.

2 Related Work

LLM-based travel planning and grounding. Recent work has extended LLM-based travel planning from direct itinerary generation toward grounded, constraint-aware planning. Multi-phase approaches separate high-level planning, information collection, and itinerary construction (Xie & Zou, 2024), while TravelPlanner established multi-city travel planning as a tool-using, multi-constraint benchmark. Retrieval-augmented systems such as TP-RAG incorporate external travel information under spatio-temporal requirements, and other work emphasizes personalization or domain-specific travel evidence (Singh et al., 2024; Karmakar et al., 2026). Fine-grained itinerary benchmarks and systems further expose temporal and spatial dependencies within individual city-days (Tang et al., 2024; Chaudhuri et al., 2025). These works move travel planning beyond plausible recommendation toward grounded plans that must satisfy interacting trip- and day-level constraints.

Multi-agent decomposition and coordination. Multi-agent approaches distribute these planning responsibilities in different ways. PMC hierarchically decomposes multi-constraint planning into dependent subtasks coordinated by manager, executor, and supervisory roles (Zhang et al., 2025). ATLAS separates constraint management, information search, planning, and checking within an iterative workflow (Choi et al., 2026), while HiMAP combines strategic coordination with parallel day-level planning, synchronized state, transactional monitoring, and replanning (Bui et al., 2026). LRPLAN instead specializes agents according to complementary language and reasoning capabilities (Karthikeyan et al., 2025).

Hybrid execution, verification, and revision. Another line of work combines LLM reasoning with algorithmic planning/external verification. ItiNera couples language-based preference reasoning with spatial optimization (Tang et al., 2024); TTG and TRIP-PAL translate structured travel requirements into problems handled by automated planners (Ju et al., 2024; de la Rosa et al., 2024); and Personal Travel Solver combines preference modeling with numerical optimization (Shao et al., 2025). Formal approaches further use satisfiability-based methods to enforce planning constraints (Hao et al., 2025). Beyond plan construction, LLM-Modulo advocates external verification of model-generated proposals (Kambhampati et al., 2024), while Iti-Validator performs travel-specific post-generation validation and correction. Agentic systems such as ATLAS and HiMAP similarly use detected failures to trigger additional search or replanning (Choi et al., 2026; Bui et al., 2026).

Our work centers coordination on explicit domain-level decisions. Grounded specialist choices are represented as structured contracts and reconciled into a joint contract that constrains downstream planning and execution. Verified violations are routed back to the responsible decision scope for bounded revision.

3 PROBLEM SETTING

We consider a natural-language request q and an evidence snapshot E that captures the retrieved and verified factual information (e.g., currently retrieved train schedules and hotel listings and prices) available for one planning episode. Interpreting q yields hard requirements H_q, soft preferences S_q, and unresolved information U_q. Following the literature, items in U_q are retained rather than resolved by default. If an unresolved item is required to evaluate a hard condition in H_q, the itinerary cannot be accepted; otherwise, it is retained as non-blocking uncertainty (e.g., for preferences in S_q).

Evidence. The evidence snapshot E is a set of records. Each record describes one entity, such as a hotel or train service, and contains an identifier that is unique in E (e.g., `hotel_123`) and factual attributes such as price, rating, and address. Given q, retrieval selects the relevant records $G = \{g_1, g_2, \ldots\}$, and $\mathrm{Ids}(G)$ denotes their identifiers. For $x \in \mathrm{Ids}(G)$ and an attribute a, $a_G(x)$ denotes the value of a in the record identified by x, and is undefined if the record does not contain a. For example, if G contains `hotel_123` with price 142, then $\mathrm{price}_G(\texttt{hotel_123}) = 142$.

Itinerary. We represent a candidate itinerary by $P = (\mathcal{C}, \Delta, T, L, V, M, R_{\mathrm{loc}})$, where $\mathcal{C}$ is the city sequence, Δ is the number of days allocated to each city, T contains the inter-city services, L contains the lodging selections and stay intervals, V contains the visits, M contains the meals, and R_{loc} contains the local movements. The elements of T, L, V, and M refer to selected entities from G, which are connected by R_{loc}. Together, their temporal information determines which visits, meals, and movements are possible within each city-day. Given $\mathrm{Ids}(P) \subseteq \mathrm{Ids}(G)$, an itinerary P is grounded if every observed attribute agrees with its evidence record:

$$\mathrm{Grounded}(P \mid q, G) = 1 \iff \forall a \text{ and } x \in \mathrm{Ids}(P), \quad a_P(x) = a_G(x). \tag{1}$$

Executability. Let $\mathcal{F}_{\mathrm{hard}}$ be the set of hard conditions covering grounding, itinerary structure, chronology, route continuity, stay coverage, budget, and the request-specific conditions instantiated from H_q (Table 6, Appendix B). Each $f \in \mathcal{F}_{\mathrm{hard}}$ maps $(P; q, G)$ to $\{0, 1\}$, representing that whether a condition holds and the information from q, P, and G is sufficient to support it. An executable travel plan should satisfy all hard constraints, as expressed by:

$$\mathrm{Exec}(P \mid q, E) \models \bigwedge_{f \in \mathcal{F}_{\mathrm{hard}}} f(P; q, E). \tag{2}$$

For example, consecutive activities i, j must satisfy $t_j^{\mathrm{start}} \geq t_i^{\mathrm{end}} + \tau_{ij}$, where τ_{ij} is the travel time. The schedules of activities must also respect opening windows and transport commitments.

Objective. Among executable plans $\mathcal{P}_{\mathrm{exec}}$, we prefer those that best match S_q. Let $U(P, q)$ denote an abstract soft-preference utility. The preferred plan is defined as

$$P^\star \in \begin{cases} \arg\max_{P \in \mathcal{P}_{\mathrm{exec}}} U(P, q), & \mathcal{P}_{\mathrm{exec}} \neq \emptyset, \\ \perp, & \text{otherwise.} \end{cases} \tag{3}$$

in which $\perp$ is returned only when no executable itinerary exists. Our planning system does not solve Eq. 3 globally. It explores a bounded subset of the combinatorial search space and returns a preferred plan only if that plan satisfies the acceptance rule in Section 4.4. Since acceptance requires $\mathrm{Exec}(P \mid q, G) = 1$, every itinerary must pass the executability check. However, bounded search may return $\perp$ even when $\mathcal{P}_{\mathrm{exec}}(q, E) \neq \emptyset$, that is, no guarantee is provided, consistent with existing LLM-based travel planning methods.

4 THE FRAMEWORK

Figure 1 illustrates our agentic framework. Given natural-language request q, *Requirement Agent* converts it into a structured request profile R. Four domain *Specialists* select cities of interest,

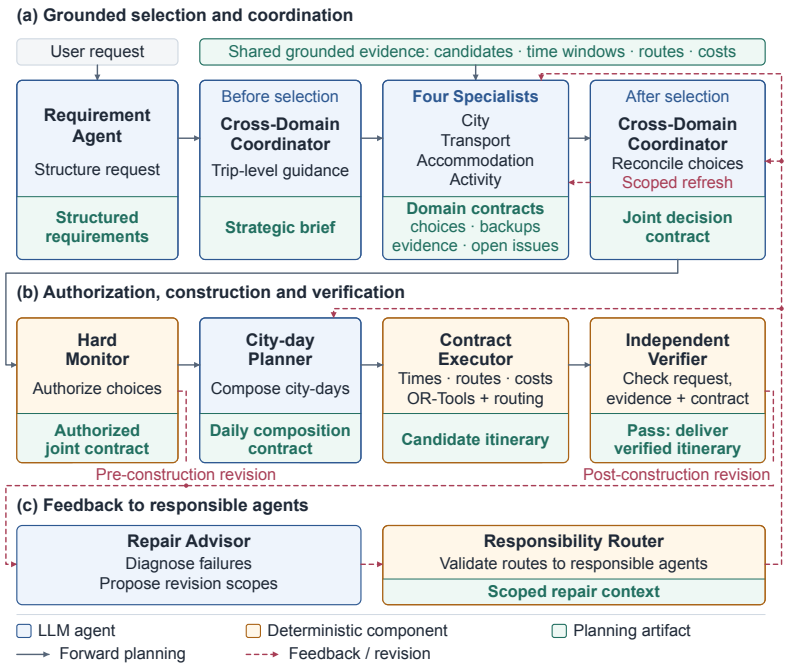

Figure 1: Overview of the proposed agentic framework for multi-city travel planning.

inter-city transport, accommodation, and activities, respectively, guided by *Coordinator*. After reconciliation, a joint contract is generated and checked through hard monitor. Then, *City-Day Planner* assigns the approved activities to days, and a deterministic executor (i.e., a solver) constructs routes, times, stays, and costs for each day. An independent verifier checks the resulting itinerary, after which *Repair Advisor* helps identify which decision domains to revisit. The deterministic Responsibility Router validates and completes these routes, directing repair context to the responsible Specialists, Coordinator, or City-day Planner. All *agents*, monitor, executor and verifier exchange information through a shared state:

$$\mathcal{S} = (R, G, D, C, Q, P, \mathcal{V}, J), \tag{4}$$

where G contains the aforementioned records, D Specialist proposals, C the joint contract, Q the daily composition, P the materialized itinerary, $\mathcal{V}$ verification findings, and J the decision history.

4.1 Domain Specialists and Evidence-based Proposals

Specialists make decisions within a domain d and scope s, such as a transport leg or destination city. Each receives the request profile R, relevant records $G_{d,s}$, and strategic guidance β, and returns:

$$D_{d,s} = (I_{d,s}, B_{d,s}, E_{d,s}), \qquad I_{d,s} \cup B_{d,s} \subseteq \mathrm{Ids}(G_{d,s}), \tag{5}$$

where I denotes the selected identifier, B backup/alternative identifiers, and E supporting evidence. Selected and backup identifiers are disjoint. The evidence retains rejected alternatives and the rationale for selection. The records $D_{d,s}$ allow the Coordinator to reconsider another decision using its evidence and available alternatives. We instantiate specialists in four different domains:

City Specialist proposes the city sequence and day allocation according to the requested destinations, trip duration, mandatory visits, and preferences. The proposed inter-city legs affect decisions by other Specialists, and the day allocation is jointly reviewed with other decisions by Coordinator.

Transport Specialist selects a candidate mobility and ranks alternatives by departure/arrival time, prices, and transport preferences for each inter-city leg. The alternatives allow Coordinator to trade transit cost against time at the destination when reconciling transport with activities and stays.

Accommodation Specialist selects a property for each destination according to room requirements, prices, capacity, and minimum-stay conditions. It retains evidence and alternatives so that accommodation can be checked against overnight coverage, transport decisions, budget, etc.

Activity Specialist selects visits and meals from city-specific records by mandatory requirements, interests, cuisine preferences, opening windows, and available time. The City-Day Planner later assigns approved items to days, and the executor determines their routes and times.

4.2 Cross-Domain Coordination

Coordinator acts in two roles. Before domain decision of each Specialist, it produces a brief strategic guidance $\beta = \mathrm{Coord}_{\mathrm{brief}}(R, G)$ covering user priorities, day allocation, budget hints, and risks for each domain. After domain decision, it reviews specialists' proposals (D) together, i.e., $\widetilde{C} = \mathrm{Coord}_{\mathrm{reconcile}}(R, G, D, \beta)$. The resulting candidate contract $\widetilde{C}$ records different reconciled decisions. From the decisions, the selected decisions consist of a joint contract:

$$C_{\mathrm{sel}} = (\mathcal{C}, \Delta, I^{\mathrm{tr}}, I^{\mathrm{st}}, I^{\mathrm{act}}), \tag{6}$$

In C_{sel}, $\mathcal{C}$ is the city sequence and Δ is the number of days allocated to each city. The identifier sets specify transport by leg (I^{tr}), hotels by city (I^{st}), and activities by city (I^{act}). The contract also retains Specialist records, backups, priorities, and restrictions, so that the coordinator can consult the retained alternatives when any selected decision needs to change.

The deterministic hard monitor checks the reconciled contract before city-day planning. A contract without violations becomes the active contract C to continue the city-day planning; otherwise, its violations enter the pre-construction revision process described in Section 4.4, in which the selected decisions can be changed as stated below.

Selection changes. Replacing a selected decision with a backup requires an explicit change record:

$$\chi = (d, s, I^{\mathrm{old}}, I^{\mathrm{new}}, r, E_\chi), \qquad I^{\mathrm{new}} \subseteq I_{d,s} \cup B_{d,s}, \tag{7}$$

where d and s identify the affected domain and scope, I^{old} records the reported previous selection, I^{new} specifies the proposed replacement, r states the conflict being addressed, and E_χ refers to the supporting evidence. Before application, programmatic checks validate the domain and scope, verify that the proposed identifiers belong to the matching Specialist's current selections or backups, and confirm that the cited evidence paths are valid. If the available selections and backups do not provide a suitable replacement, the responsible Specialist is refreshed to obtain new proposals, after which the revised selection passes through reconciliation and hard monitor again. Appendix C provides the detailed specification of selection changes and hard monitor checks applied to them.

4.3 City-Day Planning

Given the joint contract approved by hard monitor, the transport and accommodation decisions determine where each city-day u begins and ends and how much time is available:

$$\Omega_u(C, G) = (s_u, f_u, [a_u, b_u]). \tag{8}$$

where s_u, f_u are start and end locations (e.g., the hotels), and $[a_u, b_u]$ is the time window. Arrival and departure services (e.g., flights or trains) constrain the locations and available time on boundary days, while hotels anchor ordinary sightseeing days. City-Day Planner proposes daily composition Q from the approved activities. Together with Ω_u, the executor (i.e., a solver) constructs the movement during each city-day with a time schedule:

$$\mathrm{Ids}(Q) \subseteq I^{\mathrm{act}}, \qquad P_u = \mathrm{Solver}(C, Q, G). \tag{9}$$

For ordinary city-day planning, we apply OR-Tools (Perron & Furnon, 2025) to solve a prize-collecting traveling salesman problem with time windows (PCTSPTW), with the hotel as the depot. Let x_{ij} indicate a route arc, y_i indicate locations that need to be traversed, d_i be stay duration, τ_{ij} travel time, and p_i the penalty for omitting a node. The city-day objective is $\min[\sum_{i,j}(d_i + \tau_{ij})x_{ij} + \sum_{i \in N_{\mathrm{opt}}} p_i(1 - y_i)]$, where N_{opt} is the set of locations (e.g., places of interest, restaurants). Applying the solver can make the solution respect route continuity, opening windows, and the daily boundary Ω_u. More details of city-day planning are presented in Appendix D. Given each city-day itinerary P_u, the assembled itinerary over multiple cities is checked for consistent routes, stays, costs, and request satisfaction.

4.4 VERIFICATION AND REVISION

We allow verifications and revisions at two checkpoints: before city-day itinerary construction (i.e., pre-construction revision) and after verification of the assembled entire itinerary (post-construction revision). Pre-construction revision addresses conflicts visible in domain decisions, while post-construction revision addresses conflicts revealed by city-day schedules and assembled itinerary.

Pre-construction revision. After reconciliation by Coordinator, the hard monitor checks the candidate contract $\widetilde{C}$ for grounding, coverage, allocation, budget, etc. Let $\mathcal{H}_{\mathrm{err}}(\widetilde{C})$ denote the found errors or violations. City-day itinerary construction proceeds only after the contract is approved $\mathcal{H}_{\mathrm{err}}(\widetilde{C}) = \varnothing$. Otherwise, the errors and violations are analyzed by Repair Advisor who proposes revisions that are routed to the responsible Specialists or Coordinator for bounded revision, followed by renewed reconciliation and authorization. The authorized candidate becomes the active contract C, fixing the approved domain choices and city-day allocations that constrain subsequent composition and scheduling. Final itinerary feasibility remains subject to post-construction verification.

Post-construction revision. The verifier checks the constructed itinerary P against the request profile R, records G, and active contract C. We denote errors or violations $\mathcal{V}_{\mathrm{err}}(P)$. Each found error or violation is tracked by its identifier, severity, affected scope, and supporting evidence. Then, Repair Advisor follows the scopes to revisit and specifies the components required to be revised.

The revision approach depends on whether resolving an error or violation requires changing an approved selection. A daily-composition revision only updates Q while preserving C. A selection-change revision returns to the responsible Specialist and results in new contract C' and new composition Q'. Therefore the candidate itinerary is constructed as

$$P' = \begin{cases} \mathrm{Solver}(C, Q', G), & \text{daily-composition revision,} \\ \mathrm{Solver}(C', Q', G), & \text{selection-change revision,} \end{cases} \tag{10}$$

where the selection-change revision is followed by $\mathcal{H}_{\mathrm{err}}(C')$ before execution. In both revision approaches, Q' must respect the contract used for construction, and P' undergoes the verifier again.

Pre-construction revision and post-construction Revision have separate budgets and stopping conditions. The agentic framework provides the final itinerary only if the verifier satisfies $\mathcal{V}_{\mathrm{err}}(P) = \varnothing$; otherwise, it outputs $\perp$ when available revisions are exhausted. Appendix C provides the transaction, rollback, and budget-accounting details. Representative recorded execution cases, including complete coordination and verifier-guided recovery traces, together with the agent prompts, are provided in Appendix H and Appendix I.

5 EXPERIMENTS

We evaluate whether the framework produces executable itineraries and which components contribute to its performance. We first compare end-to-end results on TravelPlanner and ChinaTravel, then examine domain selection, coordination, and evidence access through ablation studies. Finally, we study whether revision before construction adds value alongside final repair. More detailed results, including paired statistical analysis, exact metric counts, recovery-stage analysis, stratified performance, computational cost, and failure diagnostics, are provided in Appendix F and Appendix G. Our code and implementation will be made publicly available upon paper acceptance.

5.1 EXPERIMENTAL SETUP

Benchmarks and metrics. TravelPlanner assesses trip-wide commonsense and user-specified constraints, while ChinaTravel emphasizes local feasibility and query-specific logical requirements (Xie et al., 2024; Shao et al., 2026). We use TravelPlanner's official Validation180 for comparisons and ablations, and Test1000 for the frozen framework. For ChinaTravel, we construct three disjoint subsets: 45 Easy queries from Easy300 for development, 50 human-authored queries from Human154 for validation and ablations, and a combination of 180 Easy/Medium queries (54/126) from Easy300 and Medium150 for testing.

Table 1: Results on TravelPlanner for prompting and agentic methods. Values are percentages.

Method	DR ↑	Commonsense ↑		Hard Constraint ↑		FPR↑
		Micro	Macro	Micro	Macro	
Validation180						
Direct LLM	100.00	78.89	0.00	0.48	0.00	0.00
Direct LLM + CoT	100.00	77.92	0.00	0.00	0.00	0.00
Act-style Tool Loop	100.00	80.35	23.89	34.29	26.67	13.33
ReAct-style Tool Loop	96.67	79.24	20.00	50.00	37.22	12.22
ReAct + Reflexion	100.00	77.50	9.44	48.81	41.11	6.67
ATLAS (Gemini-2.5-Pro; published)	100.00	88.54	48.33	82.62	74.44	44.44
HiMAP-Travel (Qwen3-8B; published)	100.00	95.64	73.61	76.85	63.33	52.78
Our framework (DeepSeek V4 Flash)	72.78	72.78	72.78	65.71	72.78	**72.78**
Official Test1000						
ATLAS (Gemini-2.5-Pro; published)	100.00	85.81	40.50	77.64	70.60	35.00
HiMAP-Travel (Qwen3-8B; published)	100.00	94.62	67.00	50.47	66.05	52.65
Our framework (DeepSeek V4 Flash)	74.60	74.60	74.60	67.55	74.40	**74.40**

Table 2: Results on ChinaTravel. All methods are evaluated on the same 50 human-authored queries.

Method	DR ↑	EPR ↑		LPR ↑		C-LPR ↑	FPR ↑
		Micro	Macro	Micro	Macro		
Direct LLM	84.00	20.37	0.00	69.55	12.00	0.00	0.00
Direct LLM + CoT	12.00	15.23	0.00	72.27	24.00	0.00	0.00
Act-style Tool Loop	46.00	53.41	0.00	79.09	56.00	0.00	0.00
ReAct-style Tool Loop	4.00	54.52	0.00	80.45	58.00	0.00	0.00
ReAct + Reflexion	34.00	58.74	0.00	90.00	66.00	0.00	0.00
Our framework (DeepSeek V4 Flash)	82.00	**81.62**	**72.00**	69.09	44.00	**60.00**	**38.00**

Following the literature, our primary metric is Final Pass Rate (FPR), the percentage of all queries whose itineraries pass every required check. In TravelPlanner, a final pass requires satisfying *all commonsense and all user-specified hard constraints*. In ChinaTravel, it requires satisfying *all environmental and all logical constraints*: environmental checks assess consistency with travel data and scheduling rules, while logical checks assess traveler requirements. We report the corresponding Commonsense and Hard-constraint pass rates for TravelPlanner, and Environmental (EPR) and Logical (LPR) Pass Rates for ChinaTravel. For each category, Macro measures the percentage of queries passing all checks, while Micro measures the percentage of individual checks passed. ChinaTravel additionally reports C-LPR, which credits satisfied logical checks only when all environmental checks pass. Delivery Rate (DR) measures successful plan delivery under each benchmark's output requirements. We present more details of experimental protocol and metrics in Appendix E.

Baselines. We compare against direct prompting, chain-of-thought (CoT) (Wei et al., 2022), Act-style and ReAct-style tool loops (Yao et al., 2023b), and ReAct with Reflexion (Shinn et al., 2023). Local methods use the same queries, output adapter, evaluator, and reported DeepSeek V4 Flash backbone. ATLAS (Choi et al., 2026), HiMAP-Travel (Bui et al., 2026), and NeSy Planning (Shao et al., 2026) retain their original settings. We use temperature-zero decoding and schema-constrained JSON; enabled post-construction revision allows up to three executor-and-verifier attempts. Appendix E gives more configuration details.

5.2 Benchmark Study

On TravelPlanner Validation180, our framework achieves the highest FPR among the methods in Table 1, at 72.78%. The strongest locally evaluated baseline, Act-style Tool Loop, achieves 13.33%, while the strongest published comparator, HiMAP-Travel, achieves 52.78%. On official Test1000, our framework achieves 74.40% FPR, outperforming HiMAP-Travel by 21.75% points. Our framework delivers 746 itineraries for the 1,000 test queries; 744 pass all benchmark checks, yielding a 99.73% pass rate among delivered plans. Our framework also successfully completes more queries than the compared methods, and nearly all returned itineraries satisfy the benchmark constraints.

Table 3: ChinaTravel results on the complete released Easy300 and Human154 query sets.

Method	DR ↑	EPR ↑		LPR ↑		C-LPR ↑	FPR ↑
		Micro	Macro	Micro	Macro		
Panel (a): Released Easy300 ($N = 300$)							
Direct LLM	13.67	25.20	0.00	56.55	0.00	0.00	0.00
Direct LLM + CoT	41.00	30.37	0.00	53.64	0.00	0.00	0.00
Act-style Tool Loop	96.00	66.88	0.00	74.54	2.33	0.00	0.00
ReAct-style Tool Loop	97.67	67.79	0.00	72.83	0.33	0.00	0.00
ReAct + Reflexion	99.33	68.02	0.00	74.54	0.67	0.00	0.00
NeSy Planning (DeepSeek-V3)†	75.30	75.30	75.30	70.40	52.60	70.40	52.60
Our framework (DeepSeek V4 Flash)	79.00	**78.65**	70.00	**74.98**	**58.67**	51.67	51.67
Panel (b): Released Human154 ($N = 154$)							
Direct LLM	12.34	22.56	0.00	68.70	10.39	0.00	0.00
Direct LLM + CoT	51.95	28.35	0.00	66.96	9.09	0.00	0.00
Act-style Tool Loop	96.75	62.10	0.00	88.36	60.39	0.00	0.00
ReAct-style Tool Loop	95.45	63.59	0.00	87.05	62.34	0.00	0.00
ReAct + Reflexion	97.40	63.20	0.00	86.75	51.95	0.00	0.00
NeSy Planning (DeepSeek-V3)†	51.90	53.20	52.50	47.00	37.60	46.50	37.00
Our framework (DeepSeek V4 Flash)	77.92	**77.47**	**66.23**	66.81	42.86	37.01	**37.01**

†Published results for the corresponding released query set (Shao et al., 2026).

Table 4: Component ablations on the validation sets.

Panel (a): TravelPlanner Validation180 ($N = 180$)

Configuration	DR ↑	Commonsense Macro ↑	Hard Constraint Macro ↑	FPR ↑
Full framework	72.78	72.78	72.78	72.78
A1: w/o Cross-Domain Coordination	47.78	47.78	47.78	47.78
A2: Coordinator w/o Domain Evidence	64.44	64.44	64.44	64.44
A3: Deterministic Specialist Selection	86.67	86.67	86.67	86.67
A4: w/o Verifier-Guided Bounded Repair	71.67	71.67	71.67	71.67

Panel (b): ChinaTravel Validation50 ($N = 50$)

Configuration	DR ↑	EPR ↑		LPR ↑		C-LPR ↑	FPR ↑
		Micro	Macro	Micro	Macro		
Full framework	82.00	81.62	72.00	69.09	44.00	60.00	38.00
A1: w/o Cross-Domain Coordination	68.00	67.46	54.00	56.82	40.00	44.09	38.00
A2: Coordinator w/o Domain Evidence	80.00	79.62	70.00	67.73	40.00	59.09	38.00
A3: Deterministic Specialist Selection	44.00	43.77	38.00	34.55	24.00	29.09	22.00
A4: w/o Verifier-Guided Bounded Repair	68.00	67.77	62.00	56.36	34.00	50.45	34.00

On the 50 human-authored ChinaTravel validation queries, our framework achieves 38.00% FPR (19/50), while all five locally evaluated baselines achieve 0% (Table 2). ReAct + Reflexion achieves a higher Logical Macro score than our framework (66.00% versus 44.00%), but fails the complete environmental evaluation on every query. Our framework achieves 72.00% Environmental Macro, allowing it to satisfy environmental and logical requirements jointly in 38.00% of cases.

On the complete released Easy300 and Human154 sets (Table 3), our framework reaches 51.67% and 37.01% FPR, respectively, numerically close to the published NeSy Planning scores of 52.60% and 37.00%, respectively. Local tool-loop baselines deliver outputs for over 95% of queries on both sets but achieve no final passes. Our framework also leaves room for improvement: on Human154, delivery reaches 77.92%, while only 37.01% of queries pass every check.

5.3 Ablation Study

Coordination and domain evidence. Replacing the Coordinator with a neutral aggregator (A1) reduces TravelPlanner FPR from 72.78% to 47.78% (Table 4). Retaining the Coordinator but hiding supporting domain evidence (A2), while preserving selected and backup identifiers, reduces FPR to 64.44%. These results support the contribution of both cross-domain coordination and Coordinator access to domain evidence on TravelPlanner. On ChinaTravel, both variants retain the full framework's 38.00% FPR. Removing the Coordinator reduces Environmental Macro Pass Rate from 72.00% to 54.00%, indicating that coordination improves environmental compliance on this validation set without a net gain in complete passes.

Table 5: Effects of revision before and after itinerary construction. On/Off indicates whether a revision stage is enabled. Time denotes mean runtime per query.

Revision stage enabled		Passed ↑	FPR ↑	Mean time (s) ↓
Before construction	After construction			
Panel (a): TravelPlanner Validation180				
Off	Off	115	63.89	**3.80**
On	Off	**125**	**69.44**	3.88
Off	On	112	62.22	16.54
On	On	123	68.33	15.66
Panel (b): ChinaTravel set				
Off	Off	14	28.00	**4.94**
On	Off	16	32.00	6.13
Off	On	16	32.00	6.76
On	On	**18**	**36.00**	18.02

Specialist selection. Table 4 A3 replaces Specialist LLM reasoning with deterministic selection while retaining the domain organization and the remaining planning pipeline. On TravelPlanner Validation180, this increases FPR from 72.78% to 86.67%, making deterministic selection the strongest tested configuration. On ChinaTravel Validation50, it reduces FPR from 38.00% to 22.00%. These results show that Specialist LLM selection provides a benefit on ChinaTravel but is outperformed by deterministic selection on TravelPlanner; its contribution is therefore specific to the planning setting.

Verifier-guided revision. Disabling post-execution verifier-guided repair (Table 4 A4) reduces FPR from 72.78% to 71.67% on TravelPlanner and from 38.00% to 34.00% on ChinaTravel, a net loss of two complete passes on each validation set. Other correction mechanisms remain enabled, indicating a modest additional benefit from final repair in these experiments.

Revision stages and computational cost. We test whether revising domain decisions before itinerary construction adds value when post-verification repair is available. Pre-construction revision revisits domain selections when the joint contract fails authorization; post-verification repair responds to violations in the constructed itinerary. We independently enable or disable these mechanisms in a 2×2 ablation. Initial Specialist selection, cross-domain coordination, contract checking, and final verification remain enabled in all conditions. We evaluate all four conditions on the same queries within each benchmark: TravelPlanner Validation180 and the 50-query ChinaTravel set. ChinaTravel uses the adapted evaluator described in Appendix E.

On TravelPlanner, pre-construction revision increases FPR from 63.89% to 69.44% without post-verification repair and from 62.22% to 68.33% with it (Table 5). Pre-construction revision alone achieves the highest FPR. Adding post-verification repair lowers FPR to 68.33% and increases mean runtime from 3.88 to 15.66 seconds. On ChinaTravel, either mechanism alone increases FPR from 28.00% to 32.00%, while enabling both achieves 36.00%. Pre-construction revision therefore adds two net passing queries even when post-verification repair is available. These conditional gains are small and are not statistically significant under the reported paired tests ($p = 0.50$). Enabling both mechanisms also increases mean runtime from 6.13 seconds with pre-construction revision alone to 18.02 seconds, showing the computational cost of the additional observed recovery.

6 Conclusion

We designed a structured agentic framework for multi-city travel planning. The framework coordinates grounded domain proposals, constrains construction through approved joint decisions, and routes conflicts to the components responsible for revising them. It achieves 74.40% FPR on official TravelPlanner Test1000 and 59.44% on constructed ChinaTravel Test180. Ablations show that coordination and evidence access improve TravelPlanner performance, while Specialist LLM reasoning benefits ChinaTravel and deterministic selection is stronger on TravelPlanner validation. The ChinaTravel recovery study records additional passing cases from early revision alongside final repair. Future work will investigate adaptive selection and revision policies to reduce LLM calls, and evaluate how the framework maintains decision consistency when user requirements and travel information change.

AI USE STATEMENT

In this work, we used generative AI solely to improve the manuscript's grammar, clarity, and readability, and to assist with software coding. We did not use AI to generate synthetic data, develop theoretical models or conceptual frameworks, formulate mathematical claims, construct or write proofs, propose or refine research hypotheses, design or evaluate research methodology or experiments, perform translation, clean or reformat datasets, conduct qualitative or thematic analyses, or interpret experimental results. All AI-assisted text was carefully reviewed by the authors. The authors take full responsibility for the manuscript, code, methods, and reported results.

REPRODUCIBILITY STATEMENT

We provide comprehensive details to facilitate the reproducibility of our work. The complete agentic framework is described in Section 4, including the shared state, domain specialists, cross-domain coordination, city-day planning, and the verification–revision mechanism.

Implementation details that are only briefly presented in the main paper are provided in the appendix, including the transaction and rollback mechanism, selection-change validation, hard monitor checks, and budget accounting for both pre- and post-construction revision (Appendices C and D describe the decision interfaces and local construction).

We also provide complete experimental details, including dataset construction, evaluation protocol, implementation settings, model configurations, and inference parameters. Concretely, Appendix E specifies the evaluation protocol; Appendices F and G provide paired results and diagnostics. Appendix I reproduces the agent prompts. Upon acceptance, we will publicly release the source code, datasets, prompts, and evaluation scripts on GitHub to enable full reproduction of the reported experiments and results.

REFERENCES

The Viet Bui, Wenjun Li, and Yong Liu. HiMAP-Travel: Hierarchical multi-agent planning for long-horizon constrained travel. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A. Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.

Soumyabrata Chaudhuri, Pranav Purkar, Ritwik Raghav, Shubhojit Mallick, Manish Gupta, Abhik Jana, and Shreya Ghosh. 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)*, pp. 17035–17064, 2025.

Jihye Choi, Jinsung Yoon, Jiefeng Chen, Somesh Jha, and Tomas Pfister. ATLAS: Constraints-aware multi-agent collaboration for real-world travel planning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Tomás de la Rosa, Sriram Gopalakrishnan, Alberto Pozanco, Zhen Zeng, and Daniel Borrajo. TRIPPAL: Travel planning with guarantees by combining large language models and automated planners. *arXiv preprint arXiv:2406.10196*, 2024.

Bin Deng, Yizhe Feng, Zeming Liu, Qing Wei, Xiangrong Zhu, Shuai Chen, Yuanfang Guo, and Yunhong Wang. RETAIL: Towards real-world travel planning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14870–14902, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. Large language models can solve real-world planning rigorously with formal verification tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3434–3483. Association for Computational Linguistics, 2025.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

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. 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*, pp. 240–249. Association for Computational Linguistics, 2024.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B. Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 22895–22907. PMLR, 2024.

Priyanshu Karmakar, Borru Vijay Sai, Shubhojit Mallick, Abhik Jana, Shreya Ghosh, and Manish Gupta. TRIPPULSE: Multi-agent travel planning with review-grounded reasoning. *arXiv preprint arXiv:2608.30924*, 2026.

T. Karthikeyan, Om Dehlan, Mausam, and Manish Gupta. LRPLAN: A multi-agent collaboration of large language and reasoning models for planning with implicit & explicit constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8280–8310. Association for Computational Linguistics, 2025.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24370–24391. PMLR, 2024.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023.

Hang Ni, Fan Liu, Xinyu Ma, Lixin Su, Shuaiqiang Wang, Dawei Yin, Hui Xiong, and Hao Liu. TP-RAG: Benchmarking retrieval-augmented large language model agents for spatiotemporal-aware travel planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12392–12418. Association for Computational Linguistics, 2025.

Laurent Perron and Vincent Furnon. OR-Tools, 2025.

Jie-Jing Shao, Bo-Wen Zhang, Xiao-Wen Yang, Baizhi Chen, Siyu Han, Jinghao Pang, Wen-Da Wei, Guohao Cai, Zhenhua Dong, Lan-Zhe Guo, and Yu-Feng Li. ChinaTravel: An open-ended travel planning benchmark with compositional constraint validation for language agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

Zijian Shao, Jiancan Wu, Weijian Chen, and Xiang Wang. 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*, pp. 27622–27642. Association for Computational Linguistics, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Harmanpreet Singh, Nikhil Verma, Yixiao Wang, Manasa Bharadwaj, Homa Fashandi, Kevin Ferreira, and Chul Lee. 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: Industry Track*, pp. 486–514. Association for Computational Linguistics, 2024.

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. 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*, pp. 1413–1432. Association for Computational Linguistics, 2024.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Kai Wang, Yibo Shen, Chao Lv, Xiaozhao Zheng, and Xuanjing Huang. TripTailor: A real-world benchmark for personalized travel planning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9705–9723. Association for Computational Linguistics, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

Chengxing Xie and Difan Zou. A human-like reason framework for multi-phases planning task with large language models. *arXiv preprint arXiv:2405.18208*, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 54590–54613. PMLR, 2024.

Jian Xie, Kexun Zhang, Jiangjie Chen, Siyu Yuan, Kai Zhang, Yikai Zhang, Lei Li, and Yanghua Xiao. Revealing the barriers of language agents in planning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1872–1888. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.93.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a.

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

Ceyao Zhang, Xiang Deng Goh, Derek Li, Hao Zhang, and Yong Liu. Planning with multi-constraints via collaborative language agents. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 10054–10082. Association for Computational Linguistics, 2025.

Guide to the appendix. Appendices C and D describe the decision interfaces and local construction. Appendix E specifies the evaluation protocol; Appendices F and G provide paired results and diagnostics. For concrete examples, see the entity trace in Appendix C.5 and the execution cases in Appendix H. Appendix I reproduces the agent prompts.

A Extended Related Work

A.1 Language models for reasoning and planning

Language models support planning by interpreting goals, proposing intermediate steps, and drawing on commonsense knowledge. Chain-of-thought prompting makes intermediate reasoning explicit (Wei et al., 2022). Tree of Thoughts extends this approach by exploring alternative reasoning paths and using evaluation and backtracking to guide search (Yao et al., 2023a). These methods offer different ways to allocate inference-time computation: one develops a reasoning sequence, while the other maintains alternatives that can be reconsidered. For itinerary planning, such reasoning can help interpret preferences and compare candidate destinations before committing to a schedule.

Planning also requires an account of how actions change the state and whether their preconditions hold. PlanBench evaluates plan generation and reasoning about change using problems with explicit planning structure (Valmeekam et al., 2023). In a complementary empirical study, Xie et al. (2025) find that constraints can exert limited influence on language-agent decisions and that the influence of the original question weakens over a planning trajectory. Their analysis motivates preserving requirements in representations that later stages can inspect. In our setting, a requirement such as a mandatory attraction or prohibited transport mode must remain connected to the choices made during construction and repair.

A.2 Tool use, retrieval, and grounded selection

Tool use connects language reasoning to information that is absent from the model's parameters or changes with the task. ReAct interleaves reasoning with actions and observations, allowing an agent to gather evidence and update its plan as it proceeds (Yao et al., 2023b). LLMCompiler makes another aspect of this process explicit: it separates a function-call plan, task dispatch, and execution to exploit dependencies and parallelism among tool calls (Kim et al., 2024). The former emphasizes an adaptive interaction loop; the latter emphasizes the organization of executable tool dependencies. Both show why the structure surrounding an LLM call matters for a multi-step task.

Travel planning adds a distinction between finding a plausible option and establishing that it belongs in the final itinerary. A retrieved hotel can satisfy room preferences yet be incompatible with the number of nights available in its city. A retrieved attraction may close before the selected inter-city service arrives. TP-RAG studies retrieval of travel trajectories and introduces EvoRAG to combine retrieved references with model reasoning; its experiments examine spatial efficiency, POI rationality, and the difficulty of conflicting or noisy references (Ni et al., 2025). Retrieval thus supplies useful planning evidence, while the combination of retrieved choices still requires evaluation.

Our proposal interface records selected identifiers, alternatives, constraints, and supporting evidence. The Coordinator uses these records to reconcile choices, and the resulting contract defines the entities available to downstream construction.

A.3 Language models with symbolic planners and optimization

Hybrid systems assign language interpretation and formal search to different components. LLM+P translates a natural-language problem into PDDL, invokes a classical planner, and translates the resulting plan back into language (Liu et al., 2023). The LLM-Modulo perspective generalizes this division of labor: language models generate candidates or provide heuristic guidance, while external critics determine whether the candidates satisfy the relevant requirements (Kambhampati et al., 2024). In both cases, the formal representation specifies the conditions under which the backend's result is valid. The quality of the interface between language and that representation is therefore part of the planning problem.

Several travel systems instantiate this approach with different backends. TTG translates requests into symbolic constraints and uses mixed-integer linear programming to construct itineraries (Ju et al., 2024). TRIP-PAL combines language models with automated planning to generate travel plans under explicit conditions (de la Rosa et al., 2024). Hao et al. (2025) formulate multi-constraint planning as a satisfiability problem; their system also uses unsatisfiable cores to explain conflicts and guide modifications to infeasible requests. Personal Travel Solver combines preference interpretation with numerical solving (Shao et al., 2025). These methods establish formal modeling and solver feedback as substantive approaches to executable travel planning.

Spatial optimization addresses a related part of the task. ItiNera combines LLM reasoning with spatial optimization for urban itinerary generation (Tang et al., 2024). Our framework uses a bounded local scheduling problem within a larger multi-city process. Approved transport and accommodation decisions supply the locations and time intervals available to city-day construction. The scheduler orders approved activities within those intervals; a conflict involving the choice of transport, hotel, or activity returns to the corresponding decision scope. This organization connects local optimization to the entities and commitments that define its input. Appendix D specifies the local objective, omission penalties, and time limits.

A.4 Multi-agent organization and cross-domain coordination

General multi-agent frameworks provide reusable ways to distribute work. AutoGen supports applications composed of agents that exchange messages and use tools (Wu et al., 2024). MetaGPT organizes collaboration through role assignments and structured workflows inspired by standard operating procedures (Hong et al., 2024). These abstractions help specify who performs a task and how intermediate outputs are communicated. Travel planning additionally requires an account of how decisions from different roles constrain one another.

Travel-oriented systems choose different units of decomposition. PMC decomposes constrained planning into subordinate tasks and coordinates their execution (Zhang et al., 2025). ATLAS organizes control around information search, constraint construction, planning, checking, and search advice; feedback can trigger additional retrieval when the available options are inadequate (Choi et al., 2026). HiMAP-Travel connects strategic coordination with day-level execution and uses a synchronized state to track shared commitments, including budget, visited venues, and transport policy (Bui et al., 2026). TRIPPULSE instead organizes selection around transportation, accommodation, meals, attractions, and events, incorporating review evidence into domain reasoning (Karmakar et al., 2026). These systems illustrate decomposition by task, planning function, time interval, and travel domain.

Our framework organizes this interaction around domain proposals and a joint decision contract. Strategic guidance informs Specialist selection; reconciliation then checks proposed changes against the exposed alternatives and their evidence. Construction uses the approved selections, and later entity changes return through reconciliation. This interface connects domain reasoning to deterministic construction and revision within one planning process.

A.5 Verification, feedback, and repair

Feedback methods differ in the source of criticism and the object they revise. Self-Refine iterates between generation, feedback, and refinement using an LLM (Madaan et al., 2023). Reflexion stores verbal feedback to inform subsequent attempts (Shinn et al., 2023). CRITIC incorporates external tool feedback when assessing and correcting generated outputs (Gou et al., 2024). These approaches make revision an explicit part of inference. For constrained planning, the reliability of a revision also depends on how the updated output is checked and whether it preserves other requirements.

System-level studies expose problems that are difficult to diagnose from the final answer alone. Cemri et al. (2025) identify failures involving system specification, inter-agent alignment, and verification or termination. Their taxonomy supports examining the interactions between agents as well as the competence of each agent. A useful travel-planning diagnostic therefore records which component proposed a change, which entities changed, and what the subsequent checks found.

The recovery mechanisms in our framework act at two points. Contract checks can prompt revision before detailed construction, while the final Verifier can reveal conflicts in the materialized itinerary.

Both paths return changed selections through the relevant authorization and construction steps. The recovery experiment varies these two stages independently to test whether one stage adds passing cases when the other is available. We complement this comparison with repair-depth distributions and individual traces, which explain the amount of work performed and the conflicts observed during recovery.

A.6 Travel benchmarks, personalization, and evaluation

Travel benchmarks emphasize different aspects of a plan. TravelPlanner evaluates complete itineraries with commonsense and query-specific hard constraints in a tool-accessible sandbox (Xie et al., 2024). ChinaTravel extends evaluation to Chinese travel environments and compositional request constraints, with separate environmental and logical checks (Shao et al., 2026). Together they provide settings in which trip-level compatibility and detailed itinerary validity can be measured using explicit outcomes.

Other benchmarks broaden this coverage. TripCraft includes public-transit schedules, events, and personas, and evaluates temporal, spatial, ordering, and persona-related aspects of an itinerary (Chaudhuri et al., 2025). TripTailor uses real-world POIs and itineraries to study personalized planning (Wang et al., 2025). RETAIL adds implicit requests, environmental context, and revision scenarios, together with a topic-guided multi-agent method (Deng et al., 2025). These developments motivate distinguishing the ability to satisfy a specified request from the ability to infer unstated needs or adapt to changing requests.

Personalization introduces an additional evaluation layer. TravelPlanner+ studies how user models affect planning and uses an LLM judge to compare generic and personalized outputs (Singh et al., 2024). TRIPPULSE uses review information to assess experiential qualities of selected options. Our evaluation focuses on executability and records a preference-selection trace: the overlap between scheduled activities and a nonempty grounded preference pool. This trace answers whether preference-linked candidates reach the schedule. Assessing the traveler's satisfaction with those activities would require a corresponding preference or user evaluation.

B Request Representation and Feasibility Conditions

B.1 Requirement provenance

The Requirement Agent produces a structured request together with evidence for its interpretation. Numeric fields retain their semantic role: traveler count, trip duration, budget, and destination count are separate quantities. For example, "I and two friends" denotes three travelers, whereas "a family of three" denotes a total group of three. The typed record distinguishes the speaker, companions, and a stated group total. A budget expression such as "8k" can be normalized to 8,000 while retaining the source phrase.

Evidence checks distinguish a verbatim substring from an authorized semantic label or a composition of date evidence. The structured output records uncertainty when the request does not support an unambiguous value. Required fields that fail validation prevent the affected planning path from continuing. These checks make an interpretation inspectable, while its semantic correctness remains part of the system's evaluation.

B.2 Evidence and decisions

Retrieved records contain identifiers and factual attributes. Proposals refer to these records; construction resolves the approved identifiers back to their attributes. An estimated local travel time retains its estimated status. Where a benchmark requires an exact route, the route must be obtained for the applicable endpoints and mode. An estimate used during local search does not become an observed route by being included in a schedule.

Table 6 expands the hard-condition families used in the problem formulation. The predicates apply where the request and benchmark require them. The internal checks determine whether the system releases an itinerary; the external evaluator determines its benchmark outcome.

Table 6: Feasibility conditions and the evidence needed to check them.

Family	Checked relationship
Grounding	Referenced entities occur in the evidence, and factual attributes agree with their records.
Structure and coverage	Dates, destinations, transport legs, and required itinerary fields form a complete trip.
Chronology	Service durations, opening windows, transfers, and arrival/departure boundaries are compatible.
Spatial continuity	Consecutive events have consistent locations and supported movements between them.
Accommodation	Overnight stays, capacity, room requirements, house rules, and applicable minimum nights are satisfied.
Budget	Costs use the applicable party, ticket, room, and night counts and remain within the request budget.
Request requirements	Mandatory entities, modes, cuisines, and other explicit requirements survive construction and revision.

C Decision Interfaces and Revision Checks

C.1 Specialist proposal schema

For domain d and scope s, write the exposed candidate identifiers as $K_{d,s}$, selected identifiers as $I_{d,s}$, and ranked backups as $B_{d,s}$. The schema enforces

$$I_{d,s} \cup B_{d,s} \subseteq K_{d,s}, \qquad I_{d,s} \cap B_{d,s} = \varnothing, \qquad |B_{d,s}| \leq 3. \tag{11}$$

The full record also includes the responsible agent, domain, scope, rejected candidates with reasons, evidence references, considered constraints, unresolved conflicts, selection rationale, tool name, query metadata, and retrieved-candidate count. Transport records are scoped to route legs; hotel and activity records are scoped to cities. A backup becomes active only through an accepted decision change.

C.2 Joint contract fields

The joint contract stores the selected city sequence and day allocation; transport selections and backups by leg; hotel selections and backups by city; and activity/dining selections and backups by city. It retains priority maps, forbidden identifiers, domain profiles, transport-mode policy, required hotel nights, minimum-stay information, activity-role quotas, and minimum activity counts. Specialist records preserve the evidence behind these choices. Reconciliation metadata, fallback events, and Hard Monitor findings provide the audit trail.

The daily composition is a separate object. Each entry contains a date, city, ordered activity identifiers, activity roles, and a rationale. It can report unresolved conflicts and request a contract revision. The Executor derives the materialized schedule from this object and the authorized contract.

C.3 Strategic guidance and joint reconciliation

The Coordinator participates at two points in initial planning. Before the Specialists search and select, it interprets the request as a set of coupled planning tasks. After their proposals become available, it reconciles concrete selections. Table 8 distinguishes the information and decisions available at these two points.

From task guidance to decisions. The first stage returns a `StrategicPlanningBrief` containing guidance for Specialist selection. An arrival-day restriction, for example, links transport timing to the activities that can fit after arrival; accommodation requirements inform the city-day allocation. The brief is stored under `strategic_specialist_task_decomposition`. Concrete entity selections and their feasibility are assessed in the subsequent proposal and authorization stages.

Table 7: Outputs and authority at the main component interfaces.

Component	Output	Decision responsibility
Requirement Agent	Request and interpretation evidence	Interpret requirements and identify unresolved information.
Domain Specialists	Selections, backups, evidence	Propose entities within their grounded scope.
Coordinator	Guidance and reconciled contract	Reconcile domains and declare supported selection changes.
Hard Monitor	Authorization findings	Check the candidate joint contract before construction.
City-day Planner	Daily composition	Assign approved activities to dates and semantic roles.
Executor	Materialized itinerary	Construct movements, timestamps, stays, and costs.
Verifier	Structured violations	Assess the constructed itinerary independently of LLM judgments.
Repair Advisor	Proposed responsibility routes	Diagnose conflicts and suggest scopes for revision.
Orchestrator	Validated state transitions	Apply updates, complete responsibility routes, and enforce the final gate.

Table 8: The two coordination stages in initial planning. Joint reconciliation is also invoked after a Specialist refresh.

	Strategic guidance	Joint reconciliation
Trigger	Interpreted request, before Specialist proposals.	Available Specialist proposals; repeated after a scoped refresh.
Input	Request constraints, preferences, destinations, and duration.	Active selections, Specialist decisions and backups, their evidence, and the current planning state.
Output	City-day strategy, day roles, budget hints, Specialist task briefs, coupled constraints, protected requirements, and uncertainties.	Joint selections and day allocations, explicit selection-change records, refresh requests, and resolved or unresolved conflicts.
Decision scope	Guide the questions each Specialist should address. Factual entity choices are made through the later proposal interfaces.	Retain selections or propose supported changes within matching domain scopes.
Checks	Structured output is parsed as a strategic brief. The brief informs subsequent planning.	Identifier, scope, and evidence checks precede application; the Hard Monitor evaluates the resulting joint contract.
Continuation	Store the brief and obtain domain proposals.	Proceed to construction after authorization, or route remaining conflicts into bounded revision.

The second stage returns a `CrossDomainCoordinationDecision`. It operates on entity identifiers and the matching Specialist records. A change includes its domain and scope, previous and proposed identifiers, the conflict being addressed, and evidence references. A day-allocation change carries its own reason and evidence. Programmatic validation checks these declarations before applying a candidate joint decision. The Hard Monitor then assesses the combined contract. Thus the strategic brief guides domain selection, while reconciliation determines which selections can advance together toward execution.

Re-entry during revision. The initial reconciliation uses the phase value `initial`. When a repair requests new Specialist proposals, reconciliation is invoked with `repair_refresh`. The refreshed choices pass through the same decision interface before reconstruction. Consequently, the architecture has two distinct initial coordination roles, with additional reconciliation calls when revision requires them. The number of coordination calls in a run depends on its conflicts and recovery path.

The coordination ablation removes strategic guidance and joint reconciliation together, measuring their combined effect within the pipeline.

C.4 Reconciliation checks

A selection-change record specifies the domain, scope, previous identifiers, proposed identifiers, rationale, and evidence references. Application checks that the proposed identifiers belong to the matching Specialist's selections or backups, and validates the declared scope and evidence paths. The canonical previous selection comes from the active state; the model's reported previous identifiers are retained separately for audit. Evidence-path validity establishes that the cited record is available. The Hard Monitor and Verifier subsequently check the relevant feasibility conditions.

Day-allocation changes additionally preserve the selected destinations and total duration. If the exposed alternatives do not provide a suitable replacement, a scoped Specialist refresh can produce a new proposal. The refreshed contract returns through reconciliation and authorization before construction uses it.

C.5 An entity traced across the decision interfaces

The following excerpts follow one transport identifier through a recorded ChinaTravel run. Query `h20241029143830714886` requests a five-day trip from Guangzhou to Shanghai for two travelers, with interests in fashion and food. We retain selected fields from the stored audit to show the relationship between a candidate fact, the Specialist proposal, the joint selection, and the materialized plan. Each block is a projection of an existing record; omitted fields remain in the full execution artifact.

Retrieved fact. The transport candidate records the route, service times, price, and duration. Dates are null in this candidate record; the request and itinerary supply the travel dates.

```
{
  "id": "flight:FL250",
  "scope": "guangzhou->shanghai",
  "mode": "flight",
  "departure_date": null,
  "arrival_date": null,
  "departure_time": "15:03",
  "arrival_time": "16:33",
  "cost_cny": 556.81,
  "duration_minutes": 90
}
```

Specialist proposal. The matching transportation record selects this identifier and exposes three alternatives. The excerpt retains two of its recorded evidence statements; the complete proposal also contains its candidate list, rejected candidates, considered constraints, and tool-query information.

```
{
  "agent": "transportation_agent",
  "domain": "transport",
  "scope": "guangzhou->shanghai",
  "selected_ids": ["flight:FL250"],
  "ranked_backup_ids": [
    "flight:FL243", "flight:FL245", "flight:FL242"
  ],
  "evidence": [
    "Flight FL250 departs 15:03, arrives 16:33, cost 556.81, within
    ↪  budget.",
    "Afternoon arrival allows same-day activity and hotel check-in."
  ],
  "tool_name": "grounded_transport_repository"
}
```

The numeric statement can be compared with the retrieved fact. The statement about same-day activities expresses the Specialist's planning assessment; construction and verification evaluate whether

the resulting day is feasible. This separates the factual support for a selection from the downstream consequences of using it.

Joint selection and materialized plan. The final state audit retains the following joint transport selection and a passing Hard Monitor result:


```
{
  "selected_transport_ids": {
    "guangzhou->shanghai": "flight:FL250",
    "shanghai->guangzhou": "flight:FL025"
  },
  "hard_monitor_passed": true
}
```


The stored plan contains the same two transport identifiers:


```
{
  "city_sequence": ["shanghai"],
  "transport_ids": ["flight:FL250", "flight:FL025"]
}
```


The same identifier, `flight:FL250`, appears in the Specialist proposal, joint selection, and stored itinerary. Exposed backups remain available for a supported revision. The excerpt illustrates the final state of these interfaces; intermediate substitutions require their corresponding change records.

C.6 Transactional repair and termination

The orchestrator routes a violation using its code, affected entities, scope, and execution evidence. Deterministic responsibility mapping completes the Repair Advisor's proposed routes. Before refresh, the system preserves the active contract, daily composition, and associated integrity signatures. A failed refresh restores those objects and stops that repair path. A changed factual contract requires reconstruction so that the resulting plan uses the updated selections.

The repair budget counts evaluated rounds. Within a round, several candidate operations may be executed and re-verified; one token is consumed when the evaluated candidates enter selection. Thus a budget of three tokens does not imply at most three candidate executions or three LLM calls. Invalid proposals, rejected entity substitutions, and non-executable operations consume no repair token. Separate stopping conditions prevent unsuccessful refreshes from continuing indefinitely.

In compact notation, let K_t be the set of executed and re-verified repair candidates in round t. The token accounting is

$$b_{t+1} = b_t - \mathbf{1}\{K_t \neq \varnothing\}, \qquad b_0 = 3. \tag{12}$$

The implementation ranks evaluated repairs using verification outcomes and change penalties, and compares the resulting state with the preserved pre-repair candidate before final acceptance. A final verification pass determines whether the selected itinerary is released. Warning-directed repairs can appear in the execution logs; the final acceptance gate depends on error-level findings.

Control-flow summary. The following procedure summarizes the control flow and its validation gates.

1. Interpret the request, retrieve evidence, and obtain strategic guidance.
2. Produce domain proposals and reconcile their selections into a candidate contract.
3. Check the contract. If early revision is enabled, route contract conflicts through bounded refresh and reconciliation.
4. Compose approved activities into city-days and materialize the itinerary within the contract boundaries.
5. Verify the itinerary. For permitted repair rounds, refresh the responsible scopes, authorize changes, execute candidates, and verify them again.
6. Select the verified candidate and apply the final acceptance gate. Serialize an accepted itinerary into the benchmark format.

D LOCAL SCHEDULING AND MATERIALIZATION

D.1 ORDINARY CITY-DAYS

Deterministic preprocessing handles protected activities before the remaining authorized activities enter local routing. The ordinary-day problem uses one vehicle with the selected hotel as depot. Arc cost includes service duration at the source and travel duration to the next node. Search minimizes route cost and penalties for omitted optional activities within the supplied day interval.

For optional activity i, the recorded penalty is

$$p_i = 100000 + 10000u_i, \tag{13}$$

$$u_i = 10\mathbf{1}_{\text{required}} + 4\mathbf{1}_{\text{type match}} + 2n_{\text{preference matches}} + q_i, \tag{14}$$

where q_i is 1, 0.5, or 0 for high-, medium-, or low-quality evidence. A protected-item safeguard uses a penalty of 10,000,000 if such an item remains in the solver scope. Protected-item preprocessing is the primary handling mechanism.

OR-Tools starts with `PATH_CHEAPEST_ARC` and uses `GUIDED_LOCAL_SEARCH`, with a 300 ms ordinary-day limit. If search returns no solution, deterministic materialization attempts the supplied order. The materializer checks travel time, opening and role windows, service duration, capacity, and return-to-boundary feasibility. An unscheduled required item remains visible as a coverage violation.

D.2 BOUNDARY DAYS

Arrival and departure services determine fixed boundary locations and available time. The ChinaTravel exact-route boundary optimizer considers only already selected activities, with supported route durations between nodes. Its protected and optional omission penalties are 100,000,000 and 100,000. It uses `PATH_CHEAPEST_ARC` with a 200 ms limit. The resulting sequence is materialized and verified. These bounded local searches provide candidate schedules; complete-trip validity depends on the assembled itinerary and its checks.

D.3 ROUTE FORMULATION AND EXECUTION INTERFACE

For a city-day, the upstream contract fixes the city, approved activity identifiers, accommodation, and relevant inter-city services. Daily composition supplies the activities to consider for that date. The local scheduler chooses a visit order within this supplied scope. Let U denote the activities entering the routing step after protected-item preprocessing, and let o denote the hotel depot for an ordinary day. All times below are measured in seconds from midnight.

Variables and inputs. The routing decision is an ordered list $R = (i_1, \ldots, i_m)$ of distinct identifiers from U, together with service-start times t_{i_k} and depot departure and return times t_o^+ and t_o^-. An empty list is permitted. Each activity has service duration s_i, effective opening and closing times a_i and b_i, and omission penalty p_i. Travel duration from i to j is τ_{ij}; the depot has $s_o = 0$. The daily interval is $[A, B]$. Route lookup first uses a matching grounded route, then an available estimated edge; the ordinary scheduler can estimate an edge when neither exists. The route record retains the associated source information.

Objective and route constraints. Writing $i_0 = i_{m+1} = o$, the routing objective is

$$\min_{R,t} \quad \sum_{k=0}^{m} \big(s_{i_k} + \tau_{i_k i_{k+1}}\big) + \sum_{i \in U \setminus R} p_i. \tag{15}$$

The selected activities are unique, $R \subseteq U$, and service cannot begin before arrival. For a nonempty route, the time constraints are

$$A \le t_o^+ \le B, \qquad A \le t_o^- \le B, \tag{16}$$

$$t_{i_1} \ge t_o^+ + \tau_{oi_1}, \tag{17}$$

$$t_{i_{k+1}} \ge t_{i_k} + s_{i_k} + \tau_{i_k i_{k+1}} \qquad (1 \le k < m), \tag{18}$$

$$t_o^- \ge t_{i_m} + s_{i_m} + \tau_{i_m o}, \tag{19}$$

$$a_i \le t_i \le \max(a_i, b_i - s_i) \qquad (i \in R). \tag{20}$$

The inequalities allow waiting before service. The implementation uses an OR-Tools time dimension with a 24-hour horizon and waiting slack. The upper bound in Equation 20 follows its range construction exactly: when service exceeds the available opening interval, the range collapses to the opening time. The subsequent materializer checks actual service completion against closing time and rejects an activity that cannot finish in time.

Each activity in this routing step has a disjunction penalty, with the utility-weighted values given above. This produces a time-window routing problem with optional visits. Protected activities are normally handled before this step; the larger protected-item penalty is a safeguard for an item that remains in the solver scope. Required coverage is checked on the constructed itinerary, including any required activity left unscheduled.

From route order to timestamps. The solver returns an activity order. The materializer constructs actual timestamps from the day boundary and route durations, using

$$\widehat{t}_j = \max\{\widehat{t}_i + s_i + \tau_{ij}, a_j\}. \tag{21}$$

Before accepting an activity, it checks completion before closing and enough time to reach the final boundary. It also applies the daily activity cap, dining limits, and effective role windows. These execution checks operate alongside the routing model. If the solver returns no solution, the materializer attempts the supplied candidate order under the same execution checks. Search is bounded by the configured time limit, so the returned route is a candidate solution rather than a certificate of optimality.

Inter-city boundaries. On a boundary day, an arrival service determines the earliest available time and start location; a departure service determines the latest finish and end location. Schematically, after the applicable transfer and buffer requirements have been applied,

$$A = \max(A_{\text{request}}, t_{\text{arrival}} + \delta_{\text{in}}), \qquad B = \min(B_{\text{request}}, t_{\text{departure}} - \delta_{\text{out}}). \tag{22}$$

Only the applicable arrival or departure bound is used on a given day. The selected service identifiers determine these boundaries. Revising an activity order does not change them. The separate ChinaTravel boundary optimizer uses its supported route durations and distinct boundary locations; its resulting sequence is also materialized and verified. If a protected visit cannot fit, the system exposes the coverage or timing conflict to the responsible revision scope.

Table 9: Inputs and outputs of local construction. The full-trip budget is assessed with the combined contract and itinerary costs.

Interface	Content
Upstream input	City and date; selected hotel and transport services; approved activity IDs; effective day boundaries.
Routing input	Activity subset, service durations, effective windows, route durations, and omission penalties.
Routing output	An ordered subset of the supplied activities, or the candidate order used by the fallback.
Materialized output	Scheduled activities with start/end times and inbound routes; accommodation, return movement, and recorded costs.
Feedback	Missing required coverage, timing conflicts, and other structured findings for verification and scoped revision.

E Experimental Protocol and Metric Definitions

E.1 Benchmark partition

TravelPlanner provides 45 training, 180 validation, and 1,000 test queries. The training split supports development and preflight checks; Validation180 supports local comparisons and component ablations; the official test service evaluates the frozen Test1000 submission.

For the original ChinaTravel study, 45 Easy queries support development, 50 Human154 queries form Validation50, and 54 Easy plus 126 Medium queries form Test180. Validation sampling uses seed 20260713; test sampling uses seed 20260712. The development Easy queries are excluded from Test180. The complete Easy300 and Human154 evaluations provide additional released-pool results. These complete pools overlap the smaller subsets and are interpreted as coverage of the released collections. The official Human1000 test is outside the reported evaluation.

E.2 Inference and output protocol

We use DeepSeek Chat for the core LLM roles through the configured provider API. The framework coordinates the model at inference time. Structured calls use temperature zero, JSON-object output, and Pydantic schema validation, with up to two attempts per call and a 45-second timeout per attempt. Output length follows the provider default. Critic and Explanation calls are disabled for formal benchmark submissions. Final repair uses the three-token accounting described in Appendix C.

Planning receives the natural-language query and the permitted information interface. Benchmark reference itineraries and official evaluation feedback are excluded from generation. Output adapters serialize the selected plan into the required schema. They have no authority to introduce replacement entities. Missing or rejected outputs remain in the evaluation denominator. Operational failures are recorded separately from planning outcomes for recovery and audit.

E.3 Baseline and ablation definitions

Direct LLM generates a plan from its supplied input. CoT adds explicit intermediate reasoning. Act-style and ReAct-style baselines use travel-domain actions and observations, with ReAct interleaving reasoning and action. ReAct with Reflexion adds model-generated feedback and regeneration. These local baselines use the same case identifiers and benchmark evaluation protocol as the corresponding full-system comparison. Published reference results retain their reported backbones and evaluation settings.

A1 replaces Specialist LLM comparison with deterministic selection while retaining the four domain scopes, candidate access, and downstream pipeline. A2 removes strategic coordination and reconciliation, using a neutral aggregation of domain decisions. A3 disables verifier-guided final repair while retaining verification and the final gate. A4 removes the Coordinator-facing domain evidence, including selection support, rejected-candidate reasons, considered constraints, and unresolved conflicts; the full contracts and grounded identifiers remain available to deterministic checks.

E.4 Metrics

Let N be the number of evaluated queries and D_p the delivery indicator for query p. TravelPlanner delivery counts a nonempty itinerary; ChinaTravel delivery follows the schema-validity protocol. For an applicable constraint family F, with outcomes $z_{pc} \in \{0,1\}$ and applicable set C_p^F, the percentage scores are

$$\mathrm{Micro}_F = 100 \frac{\sum_p \sum_{c \in C_p^F} z_{pc}}{\sum_p |C_p^F|}, \tag{23}$$

$$\mathrm{Macro}_F = \frac{100}{N} \sum_p \mathbf{1}\{\text{all applicable checks in } F \text{ pass}\}. \tag{24}$$

The released evaluator determines applicability and missing-output handling. TravelPlanner's eight commonsense checks cover sandbox membership, completeness, current-city consistency, city

routes, restaurant diversity, attraction diversity, transportation compatibility, and minimum nights. Its query-dependent hard checks cover budget, room rules, cuisine, room type, and transportation. Validation180 has 1,440 commonsense checks and 420 applicable hard checks.

ChinaTravel's environmental evaluation contains 26 atomic checks across entity, transport, accommodation, restaurant, chronology, and spatial-continuity families. Let E_p^{all} denote passage of all environmental checks, and let L_{pk} denote logical outcomes for the K_p applicable requirements. Then

$$\text{C-LPR} = 100 \frac{\sum_p E_p^{\text{all}} \sum_{k=1}^{K_p} L_{pk}}{\sum_p K_p}, \tag{25}$$

$$\text{FPR} = \frac{100}{N} \sum_p D_p E_p^{\text{all}} \prod_{k=1}^{K_p} L_{pk}. \tag{26}$$

The C-LPR denominator includes all applicable logical checks; environmentally invalid plans contribute no logical credit to its numerator. FPR requires the same itinerary to pass all required layers. For Validation50, the original component study contains 1,300 environmental and 220 logical checks.

E.5 Paired statistical analysis

For full-system and ablation outcomes $Y_p^{\text{full}}, Y_p^a \in \{0, 1\}$, we report

$$\Delta_a = \frac{100}{N} \sum_p (Y_p^{\text{full}} - Y_p^a). \tag{27}$$

We report the four paired outcomes and compute exact two-sided McNemar tests from the discordant pairs. The tests are unadjusted individual contrasts. Paired-bootstrap intervals resample queries with replacement while keeping each Full/ablation pair together and use the 2.5th and 97.5th percentiles of the resulting FPR differences. Stratified pass-rate intervals use the Wilson method. Both interval types describe variation across queries in the evaluated set.

E.6 Recovery-stage protocol

The recovery study toggles pre-construction contract revision and post-verification repair independently. Initial planning and coordination remain active. Its 50 ChinaTravel queries comprise 20 Easy, 20 Medium, and 10 preference-base cases, and use an adapted ChinaTravel-style evaluator. The saved scheduling limit is 120 ms; final repair has a budget of three or zero. All recovery contrasts compare conditions within this run using `ct_fpr_pass`. Its fully enabled condition has 36% CT-FPR; the original ChinaTravel Validation50 component study uses a different protocol and has 38% FPR.

F Detailed Results and Component Analysis

We examine the query-level changes behind the aggregate results, followed by exact metric counts and recovery workload. A1–A4 retain the definitions in the experimental protocol above.

F.1 Paired component effects

To isolate the contribution of each component, we compare the final-pass outcomes of the full framework against its corresponding ablations using paired success/failure results on the same instances. Table 10 reports the discordant pairs, paired-bootstrap confidence intervals, and exact McNemar tests.

On TravelPlanner, A2 produces the largest effect, improving the success rate by 25.0 percentage points (95% CI: 18.9–31.7, $p < 10^{-13}$), while A4 yields a smaller but significant gain of 8.3 points (p = 0.0015). A3 shows no statistically significant difference, and removing A1 instead improves performance by 13.9 points, indicating that this component is detrimental on TravelPlanner.

On ChinaTravel, only A1 exhibits a significant positive contribution, with a 16.0-point improvement (95% CI: 4.0–28.0, p = 0.021). The remaining ablations have confidence intervals spanning zero and are not statistically significant, suggesting that their effects are limited on this smaller validation set.

Table 10: Paired final-pass outcomes on TravelPlanner Validation180 and ChinaTravel Validation50. Effect is Full minus ablation in percentage points; intervals are paired-bootstrap 95% intervals, and p is the exact two-sided McNemar value.

Set	Variant	Both	Full only	Ablation only	Neither	Δ	95% CI	p
TravelPlanner	A1	130	1	26	23	-13.89	[-19.44, -8.89]	4.172×10^{-7}
TravelPlanner	A2	86	45	0	49	+25.00	[18.89, 31.67]	5.684×10^{-14}
TravelPlanner	A3	127	4	2	47	+1.11	[-1.67, 3.89]	0.6875
TravelPlanner	A4	113	18	3	46	+8.33	[3.89, 13.33]	0.00149
ChinaTravel	A1	10	9	1	30	+16.00	[4.00, 28.00]	0.02148
ChinaTravel	A2	16	3	3	28	+0.00	[-10.00, 10.00]	1
ChinaTravel	A3	15	4	2	29	+4.00	[-6.00, 14.00]	0.6875
ChinaTravel	A4	16	3	3	28	+0.00	[-10.00, 10.00]	1

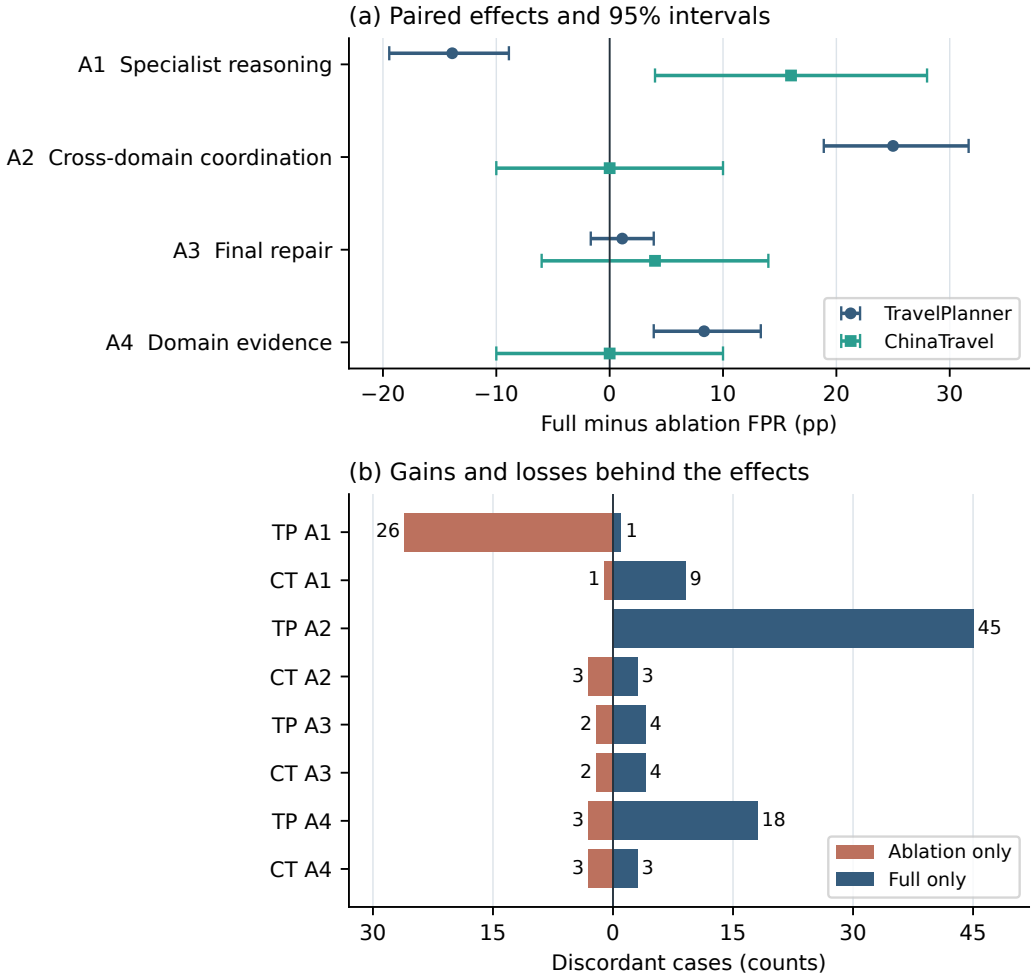

Figure 2: Paired component effects and discordant outcomes. The upper panel shows Full-minus-ablation FPR with paired-bootstrap 95% intervals. The lower panel displays the actual numbers of queries that pass under only one condition. A3 denotes final repair and A4 denotes Coordinator-visible domain evidence.

F.2 Exact metric counts

Table 11 gives the numerators and denominators for the ChinaTravel component study. Full and A4 each pass 19 queries, although their environmental and logical scores differ. The intermediate metrics therefore reveal changes that the aggregate FPR alone does not show.

Table 11: Exact original ChinaTravel Validation50 component-study numerators.

Variant	DR	EPR micro	EPR macro	LPR micro	LPR macro	C-LPR	FPR
Full	41/50	1061/1300	36/50	152/220	22/50	132/220	19/50
A1	22/50	569/1300	19/50	76/220	12/50	64/220	11/50
A2	34/50	877/1300	27/50	125/220	20/50	97/220	19/50
A3	34/50	881/1300	31/50	124/220	17/50	111/220	17/50
A4	40/50	1035/1300	35/50	149/220	20/50	130/220	19/50

Table 12: Exact Full-framework ChinaTravel Test counts. Environmental and logical micro scores have different denominators.

Subset	N	DR	EPR micro	EPR macro	LPR micro	LPR macro	FPR
Easy	54	46/54	1192/1404	42/54	233/286	33/54	31/54
Medium	126	107/126	2780/3276	105/126	858/1061	77/126	76/126
Combined	180	153/180	3972/4680	147/180	1091/1347	110/180	107/180

F.3 Coordinator-visible evidence and process events

We examine Coordinator workload under Full and A4 in Figure 3. Removing Coordinator-visible evidence increases corrections from 67 to 126 and rejected proposals from 27 to 115 on TravelPlanner. On ChinaTravel, corrections increase from 5 to 37 and rejected proposals from 3 to 28. Early-revision attempts also increase on both sets. Thus the evidence interface affects the work performed during coordination as well as the final outcomes in Table 10.

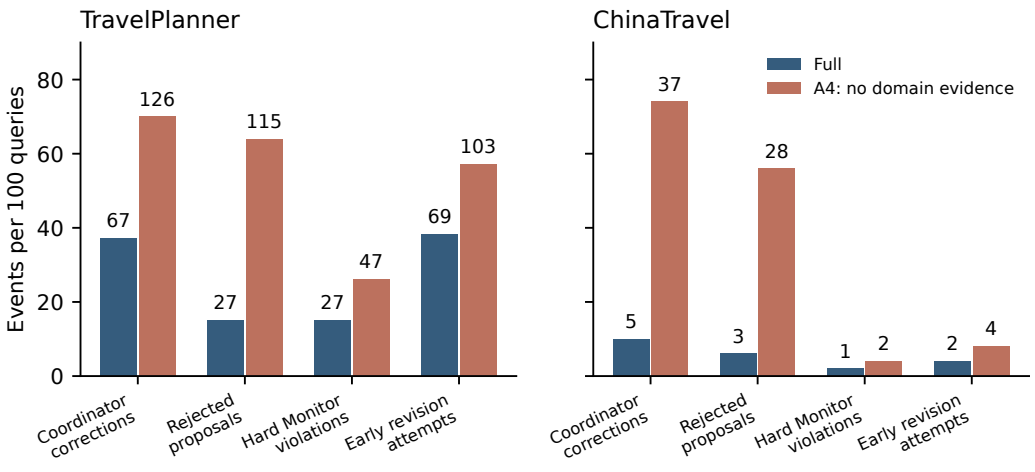

Figure 3: Process events under Full and A4 (without Coordinator-visible domain evidence). Heights are events per 100 queries; annotations give raw event counts. Several events may occur within a query.

Component effects. Coordination has its largest measured effect on TravelPlanner: removing it reduces FPR by 25.00 points, with 45 Full-only passes and no ablation-only passes. Removing Coordinator-visible evidence reduces FPR by 8.33 points. Semantic selection has opposite effects

across the benchmarks: deterministic selection achieves higher FPR on TravelPlanner, whereas LLM selection achieves higher FPR on ChinaTravel Validation50. Because A1 retains the four domain Specialists, this comparison isolates their selection policy within the domain architecture.

Final repair increases FPR by 1.11 points on TravelPlanner and 4.00 points on ChinaTravel. Each comparison contains four Full-only and two ablation-only passes ($p = 0.6875$). The recovery-stage experiment below examines where revision contributes within the pipeline.

F.4 Recovery stages and computational workload

The recovery experiment tests early contract revision and final repair in a factorial design. The main-text table reports the four condition-level outcomes. Tables 13 and 14 add the paired contrasts and the recorded work behind those outcomes. Mean runtime is 4.94, 6.13, 6.76, and 18.02 seconds

Table 13: Paired CT-FPR effects of enabling each recovery stage.

Contrast	Enabled only	Disabled only	Δ (pp)	p
Early, final off	2	0	4.00	0.50
Early, final on	2	0	4.00	0.50
Final, early off	2	0	4.00	0.50
Final, early on	2	0	4.00	0.50

Table 14: Recovery activity in the same four-condition run. Counts refer to cases; grounding calls are totals.

Early	Final	Early attempted	Early changed	Final attempted	Grounding calls
Off	Off	0	0	0	16
On	Off	21	3	0	69
Off	On	0	0	8	65
On	On	21	3	9	934

with neither stage, early revision only, final repair only, and both stages enabled, respectively. Enabling both stages raises grounding calls to 934, compared with 16–69 in the other conditions. The additional recovery therefore comes with a substantial increase in retrieval work in this run.

Recovery placement. Early revision adds two CT-FPR passes at either setting of final repair. Final repair also adds two passes at either setting of early revision. The corresponding factorial interaction is $36 - 32 - 32 + 28 = 0$ percentage points. Each contrast contains two enabled-only and zero disabled-only passes ($p = 0.50$). The observed gains are additive in this run; their paired uncertainty and retrieval cost remain relevant to interpreting the stage effects.

G Stratified Performance, Cost, and Failure Analysis

We analyze difficulty, runtime, residual failures, and recovery behavior using the frozen test runs and the complete released-query evaluations. Figure captions identify the query set and the denominator used for each analysis.

G.1 Performance by difficulty and trip duration

TravelPlanner FPR decreases with benchmark difficulty, from 87.64% on Easy queries to 77.78% on Medium and 56.43% on Hard (Figure 4). The 31.21-point Easy–Hard gap identifies the Hard group as the largest remaining source of test failures.

For the duration analysis, we group queries by the duration resolved by the request interpreter. Table 15 includes the 44 unresolved cases as a separate category. FPR is 73.74%, 75.68%, and 83.74% for the resolved 3-, 5-, and 7-day groups. These groups differ in their requests and constraints, so this breakdown describes their performance rather than isolating a causal effect of trip length.

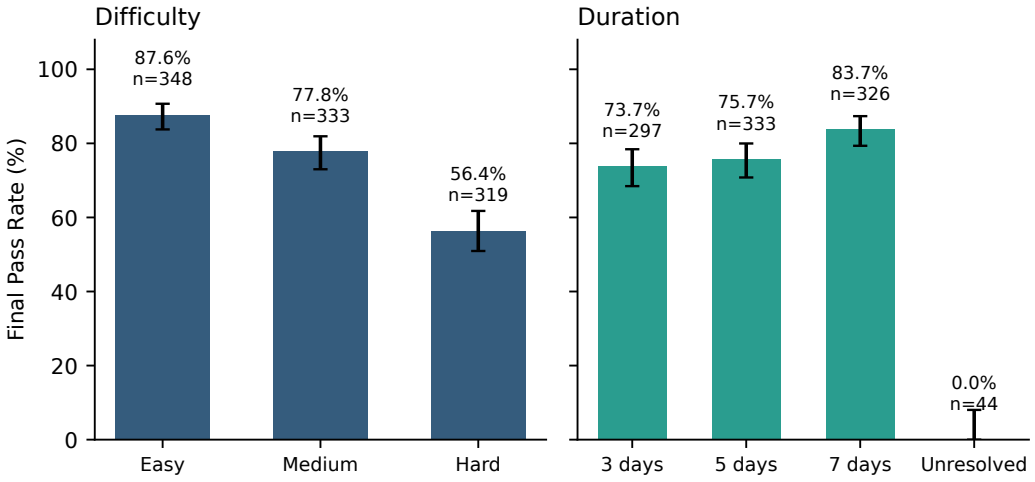

Figure 4: TravelPlanner Test1000 FPR by benchmark difficulty and parsed duration. Error bars are Wilson 95% intervals. The unresolved-duration category retains the 44 cases for which request metadata was not resolved.

Table 15: TravelPlanner Test1000 by difficulty and parsed duration. Unresolved duration remains a separate category.

Grouping	Subset	N	Delivered	Passed	FPR (%)	Wilson 95% CI
Difficulty	Easy	348	305	305	87.64	[83.77, 90.70]
Difficulty	Medium	333	260	259	77.78	[73.01, 81.91]
Difficulty	Hard	319	181	180	56.43	[50.94, 61.76]
Duration	3 days	297	219	219	73.74	[68.45, 78.42]
Duration	5 days	333	254	252	75.68	[70.79, 79.97]
Duration	7 days	326	273	273	83.74	[79.35, 87.35]
Duration	Unresolved metadata	44	0	0	0.00	[0.00, 8.03]

G.2 Computational cost and its distribution

We examine computational work at two levels. Figure 5 shows the distribution across individual queries. Figure 6 then attributes the aggregate token and latency totals to functional components. The call audit covers 1,000 Full-framework TravelPlanner test queries and 180 Full-framework ChinaTravel test queries.

TravelPlanner uses a median of 18 LLM calls and 95,180 recorded tokens per query. ChinaTravel uses 21 calls and 107,971 tokens. Median cumulative LLM latency is 96.9 and 114.7 seconds, respectively. Table 16 also shows that the TravelPlanner Hard group requires more calls and has a higher latency tail than the Easy and Medium groups. These measurements account for completed calls and recorded failed-call audits. Cumulative latency sums the durations of LLM calls; concurrent calls and other computation determine the separate wall-clock duration of a run.

Table 16: Computational characteristics of the original test runs. Latency is cumulative LLM latency. Repair rates use the repair-audit subset.

Set	N	Median calls	Median tokens	Median latency (s)	P90 (s)	Repair audit N	Repair (%)
TP (overall)	1000	18.0	95,180	96.9	226.0	766	6.4
TP (Easy)	348	17.0	82,446	90.9	192.3	307	5.5
TP (Medium)	333	17.0	90,339	94.6	205.8	265	4.2
TP (Hard)	319	22.0	107,490	107.5	299.5	194	10.8
CT	180	21.0	107,971	114.7	187.1	174	70.7

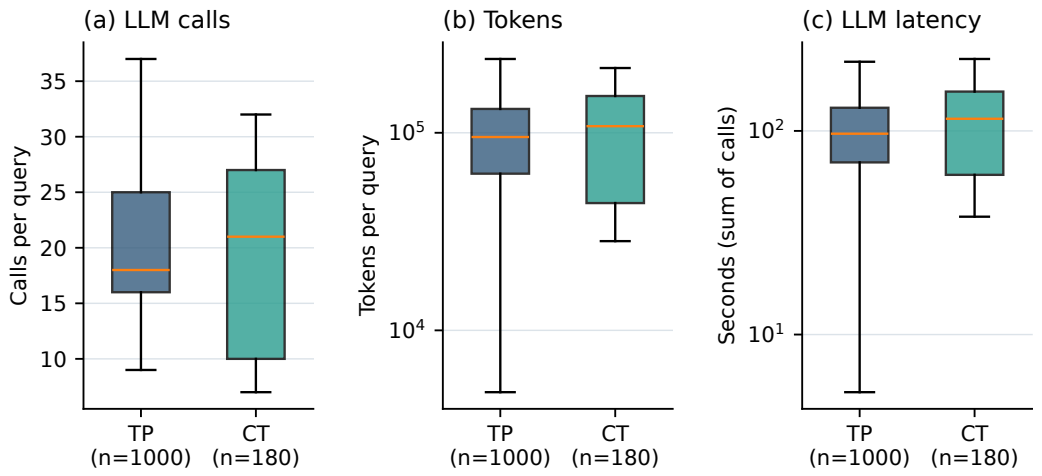

Figure 5: Computational work per Full-framework test query. Boxes show the median and interquartile range; whiskers extend to the usual 1.5 interquartile-range limit. Outlier markers are omitted for legibility, and all observations remain in the source data and descriptive statistics. Token and latency axes use a symmetric-log scale, which retains zero observations.

Specialist selection accounts for 58.42% of TravelPlanner tokens and 78.05% of its cumulative LLM latency. Its ChinaTravel shares are 38.25% and 51.44%. Repair and correction account for 19.49% of TravelPlanner tokens and 35.24% of ChinaTravel tokens, with corresponding latency shares of 7.99% and 25.27%. This category includes correction and refresh calls as well as explicit repair calls. The allocation is consistent with a greater recorded correction workload in the more detailed city-day setting.

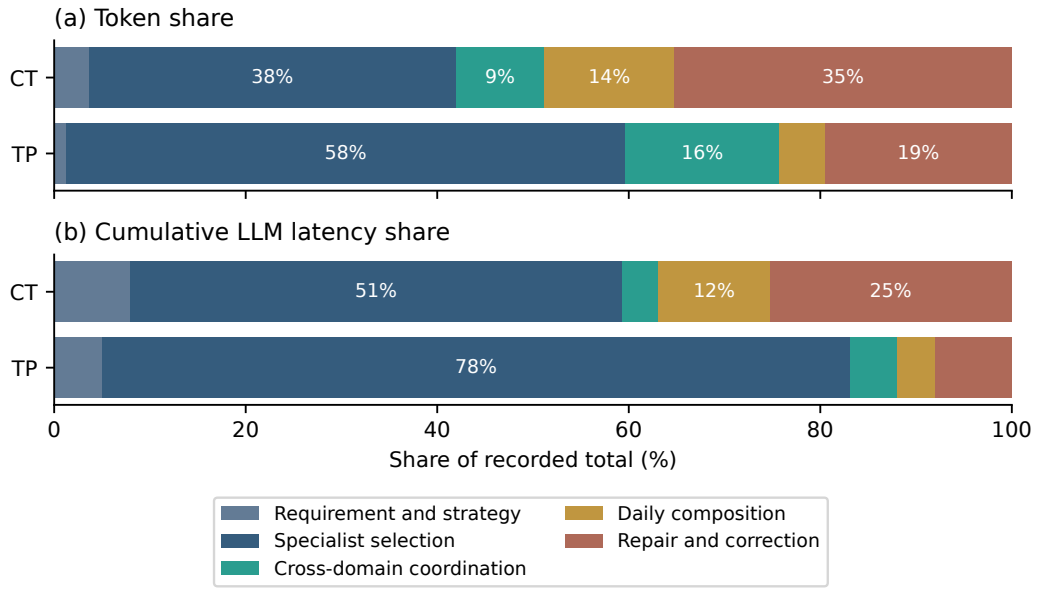

Figure 6: Functional shares of recorded tokens and cumulative LLM latency in the original Full-framework test runs. Shares are computed from totals across all audited cases. Repair and correction includes schema or semantic corrections and refresh calls.

G.3 Residual failure categories

Figure 7 compares the failure categories observed in 256 unsuccessful TravelPlanner test cases and 73 unsuccessful ChinaTravel test cases. A case can contribute to several categories. Budget or cost findings occur in 120 TravelPlanner failures (46.88%), followed by activities or dining in 67 (26.17%). ChinaTravel failures are concentrated in request-specific hard constraints: 43 of 73 cases (58.90%) have such findings. Requirement or schema failures occur in 44 TravelPlanner cases and 27 ChinaTravel cases.

The distributions point to different sources of residual error. TravelPlanner often requires reconsidering a combination of costs and travel choices. ChinaTravel more often fails to satisfy a particular request even when many individual environmental checks pass. The examples in Appendix H show how these conflicts are represented and routed through the implemented decision scopes.

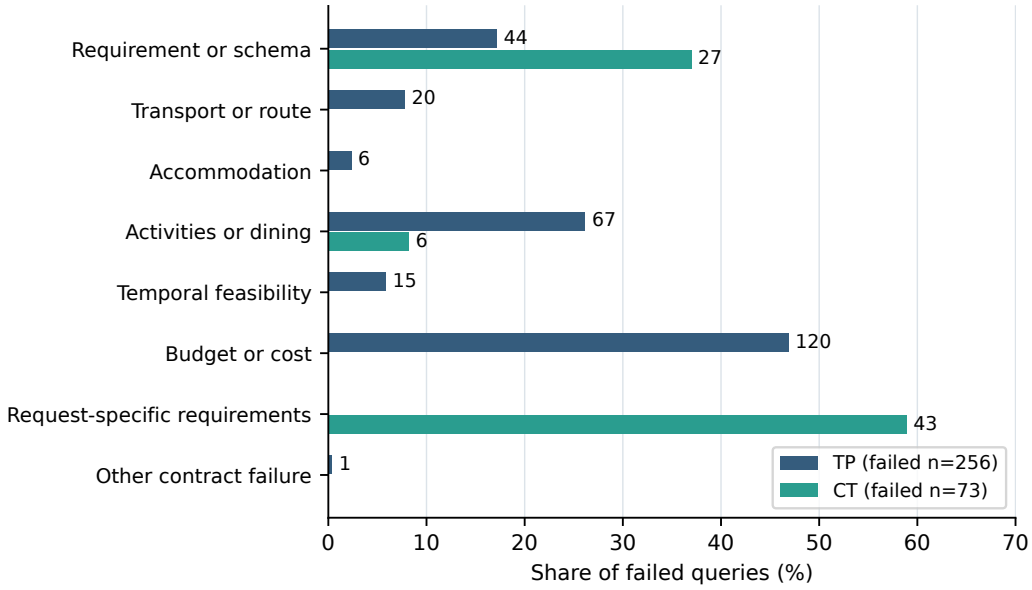

Figure 7: Failure categories in the original test runs. Percentages use unsuccessful queries as the denominator: 256 for TravelPlanner and 73 for ChinaTravel. Categories overlap; the bars describe the prevalence of findings within failed cases.

G.4 Repair depth and observed error resolution

Repair counters are available for 766 TravelPlanner and 174 ChinaTravel test cases. Of these, 49 and 123 enter at least one executed and re-verified repair. Figure 8 shows that 717 TravelPlanner cases use no repair tokens, while 9 use all three. ChinaTravel has 51 zero-token cases and 79 three-token cases. The latter distribution shows substantially more frequent use of the available repair budget.

Among executed repairs, all recorded errors are resolved in 9 TravelPlanner cases and 33 ChinaTravel cases. Another 28 and 69 cases undergo warning-directed repair with zero errors before and after the attempt. Six ChinaTravel cases reduce their error count while retaining errors, and two finish with more errors. These measurements describe the changes observed along actual repair trajectories. The paired component experiment separately measures how the complete system's final-pass outcomes change when repair is disabled.

G.5 Failure gates on the complete released query pools

The complete Easy300 and Human154 runs permit a further analysis of where queries leave the evaluation pipeline. Figure 9 assigns each case to its first failed gate. Easy300 has 63 schema failures, 27 environmental failures after the schema gate, and 55 logical failures after the environmental gate;

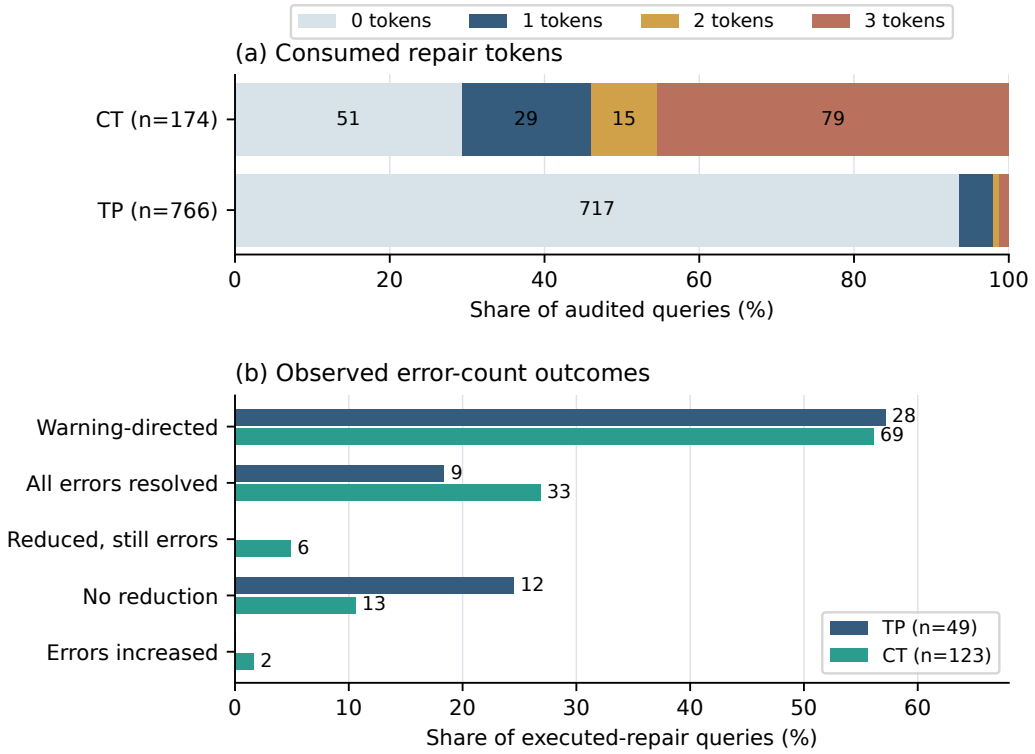

Figure 8: Consumed repair-token depth and observed pre/post error-count outcomes. The depth panel uses cases with recorded repair counters (TP: 766; CT: 174). The outcome panel uses cases with executed repair (TP: 49; CT: 123). A token counts an evaluated repair round, which may contain multiple candidate executions.

155 cases pass all gates. Human154 has 34, 18, and 45 cases at the corresponding failure gates, with 57 complete passes. Unlike the overlapping categories above, these stages partition each released query pool.

Executed repair occurs in 66 Easy300 and 37 Human154 cases. All recorded error-level findings are resolved in 46 and 23 of these cases. Figure 10 expands this summary into pre/post error-count transitions. The distinction between error resolution and benchmark passage is useful: the internal Verifier and the released evaluator assess related conditions through different implementations.

G.6 Preference propagation and repeated-query agreement

We measure preference propagation through the overlap between scheduled activity identifiers and a nonempty grounded preference pool. Figure 11 plots the empirical cumulative distribution over eligible queries. Median overlap is 50.00% for Easy300 and 66.67% for Human154, based on 33 and 92 queries, respectively. The curves describe how preference-linked candidates reach the schedule within these eligible samples.

The preference-bearing requests are a broader group: 151 Easy300 and 129 Human154 queries. Specialist records contain a preference trace for 95.36% and 96.12% of those requests. A nonempty grounded preference pool is available for 29.14% and 76.74%. The ECDF further requires scheduled activities for computing overlap. This sequence locates a potential loss before scheduling: a stated preference may be recorded by a Specialist without yielding eligible grounded activities.

Some queries appear in both the original constructed subsets and the later released-pool runs. Table 17 compares outcomes on these shared identifiers. The Easy overlap has 25 cases passing in both runs, 6 in the original run only, and 5 in the released-pool run only. The Human overlap has

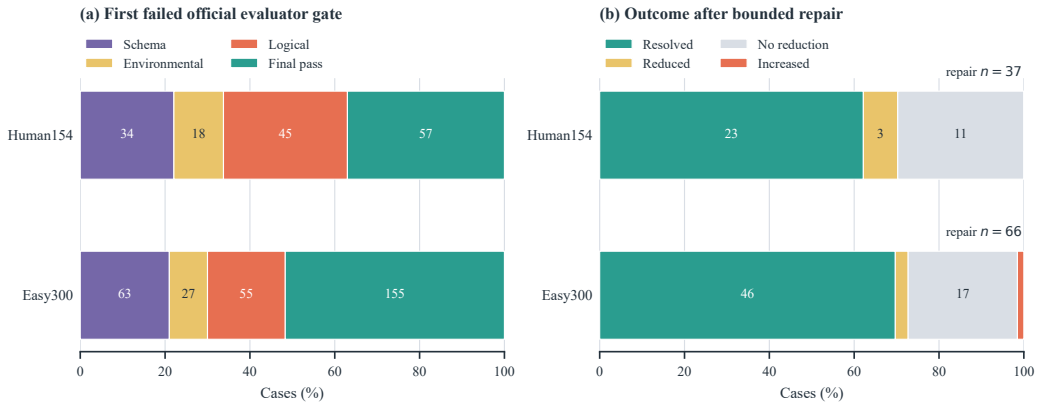

Figure 9: Released-pool evaluation gates and conditional repair outcomes. The left panel partitions all 300 Easy and 154 Human queries by their first failed gate or final passage. The right panel describes the 66 and 37 cases that entered executed repair. These populations differ from the original Test180 repair audit.

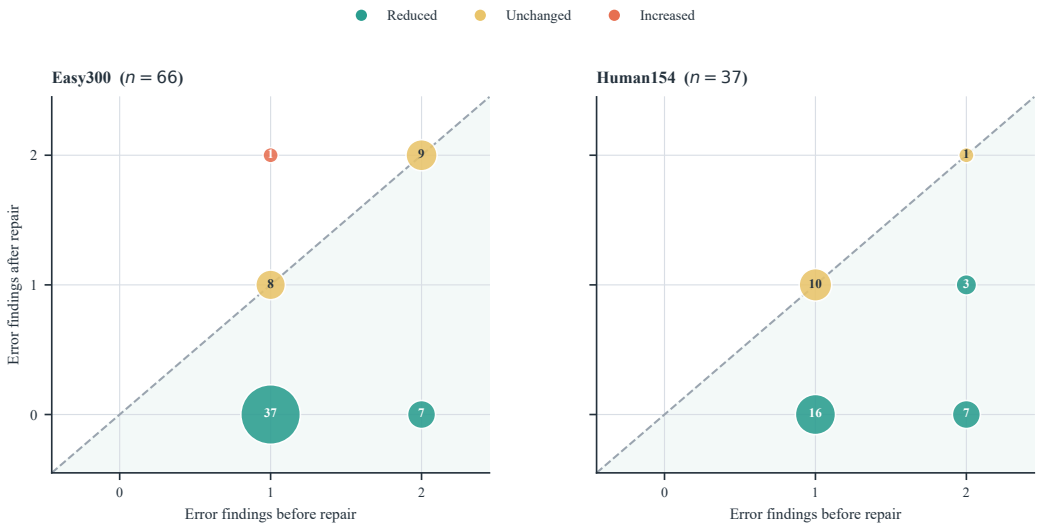

Figure 10: Recorded error counts before and after repair on Easy300 and Human154. Bubble sizes and annotations indicate the number of cases with each transition. Points below the equal-count diagonal have fewer errors after repair. The plot describes executed trajectories, complementing the controlled recovery-stage comparison.

11 passing in both, with 8 and 7 passing exclusively in one run. Similar aggregate counts therefore conceal changes in which queries pass. The two executions provide a descriptive consistency check on shared requests under their recorded protocols.

Table 17: Final-pass agreement on query IDs shared by the original constructed subsets and the later released-pool runs.

Comparison	N	Both pass	Original only	Released only	Neither
Test Easy54 vs released Easy300	54	25	6	5	18
Validation50 vs released Human154	50	11	8	7	24

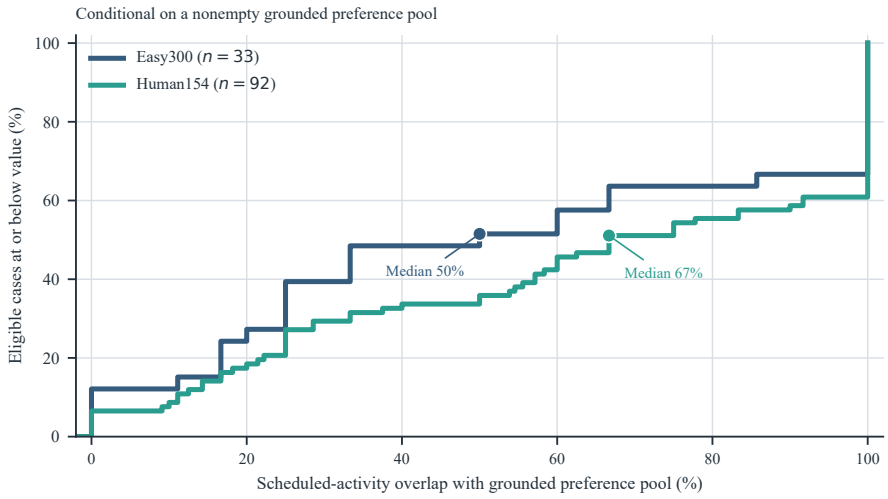

Figure 11: Scheduled-activity overlap with a nonempty grounded preference pool on the released query sets. Curves are empirical cumulative distributions over 33 Easy300 and 92 Human154 eligible cases. Annotated markers identify the medians. This is a trace of preference-linked entity selection and scheduling.

G.7 Remaining limitations and directions for improvement

The diagnostics identify three priorities for further development. First, preference-bearing requests can reach the Specialists without yielding a nonempty preference pool. Better candidate retrieval and coverage checks could address this loss before scheduling. Second, TravelPlanner failures concentrate on budget, while ChinaTravel failures often involve request-specific requirements. These patterns motivate revision policies that target the binding constraint and account for the cost of additional retrieval.

Third, the local scheduler operates within the approved entities and a bounded search time. Broader alternatives or changes to city-day allocation may be needed when that scope contains no workable schedule. The responsibility interface provides a route for requesting such changes; the current experiments evaluate the implemented policy and search limits. Repeated-run evaluation would help characterize variation in component effects, and user evaluation could establish whether the observed preference overlap corresponds to traveler satisfaction.

H Recorded Execution Cases

H.1 Coordination before construction

TravelPlanner Validation case 0 requests a three-day trip from Washington to Myrtle Beach with a USD 1,400 budget. The coordination ablation selects an outbound flight and a return self-driving leg, violating the benchmark's transport-compatibility rule. The full contract selects flights F3927581 and F3791200 for the two directions. The full configuration passes, while the ablated contract is rejected by the Hard Monitor. This case illustrates a cross-domain consistency check before detailed itinerary construction.

H.2 A multi-city execution and recovery trace

TravelPlanner Validation case 166 requests a seven-day trip for two travelers from Santa Ana to three Colorado cities, within a USD 7,700 budget. The request specifies private rooms, excludes self-driving, and lists Italian, French, Chinese, and American cuisine preferences. The selected city sequence is Alamosa, Colorado Springs, and Durango.

Selection and authorization. The state audit contains six candidate city sequences. Its joint selection contains four transport legs, three hotels, and 36 activity/dining identifiers from the recorded candidate facts. The selected-contract cost lower bound is USD 6,900. The Hard Monitor passes the contract and records zero entity changes. The City-day Planner records 12 assigned activity identifiers. The larger approved pool therefore supplies choices for daily composition, while the resulting assignment determines which activities advance into execution.

Execution feedback and revision. Initial verification finds one error and 22 warnings. Recovery routes the conflict to five responsible agents and requests three Specialist updates. One contract refresh, including a full-replan refresh, produces seven executable repair candidates within one consumed token. The system selects a repaired candidate. Table 18 summarizes the retained stage counts; a complete sequence of intermediate entity substitutions is not available in this summary.

Table 18: Recorded stages for TravelPlanner Validation case 166. Counts describe this execution.

Stage	Recorded outcome
Grounded selection	Six city-sequence candidates; final selection of three cities, four transport legs, three hotels, and 36 activity/dining IDs.
Joint authorization	Hard Monitor passed; cost lower bound USD 6,900 against budget USD 7,700; zero Monitor entity changes.
Daily composition	12 assigned activity IDs; zero recorded out-of-contract entity IDs.
Initial verification	One error and 22 warnings.
Responsibility routing	One route, five responsible agents, and three Specialist update requests.
Refresh and execution	One contract refresh, one full-replan refresh, seven executable repair candidates, and one consumed token.
Final verification	Zero errors and 22 warnings; a repaired candidate was selected.
Official evaluation	Delivery, commonsense Macro, hard-constraint Macro, and Final Pass all passed.

Final verification. The selected repair reduces the error count to zero while leaving 22 warnings, including ungrounded local-route findings and a default meal-window assumption. The itinerary passes the internal error-level gate and the official TravelPlanner evaluation. The warning record identifies residual assumptions in the accepted plan.

A conflict that remains upstream. Validation case 40 illustrates the earlier stopping point. Its request has a USD 1,800 budget, while the selected contract has a recorded cost of USD 1,992. The Hard Monitor emits `SELECTED_CONTRACT_BUDGET_EXCEEDED` and routes the conflict to transport, hotel, and activity scopes. Two pre-execution refresh attempts produce zero committed contract changes. The system withholds the itinerary. Together, the two cases show where revisions enter the pipeline: an identified contract conflict can trigger revision before construction, while a conflict exposed by execution enters the verifier-guided recovery path.

H.3 Repair under detailed city-day constraints

The recorded ChinaTravel case requests a two-day trip from Guangzhou to Shanghai for three travelers, including Xujiahui Catholic Church. The grounded scope contains 24 transport facts, eight hotel facts, 18 activity/dining facts, and 29 route records. The contract selects outbound FL250, return FL024, a hotel, and eight activity/dining identifiers.

Initial verification records two errors and one warning, with routes for mandatory-place coverage, the locked decision, and activity coverage. Three repair tokens, three contract refreshes, and twelve executable candidates are recorded. The final plan visits the required church on the second day from 09:14 to 10:44. Eight exact protocol routes are recorded; the final internal errors and warnings are zero, and the benchmark records a final pass. The serialization audit records no entity reselection or synthesized route.

I AGENT PROMPTS AND RUNTIME PACKETS

The prompts below specify the roles used by the implementation. At runtime, the structured-call client appends the role-specific JSON Schema and supplies the query, candidate records, active decisions, or repair context needed by that role. The TravelPlanner requirement interpreter uses the benchmark-specific extraction prompt reproduced at the end of this section. The text inside angle brackets denotes runtime content supplied by the caller. Line wrapping is typographical.

The common output instruction is:

```
Return one valid JSON object only. It must validate against this JSON
↪  Schema:
<role-specific JSON Schema>
```

On a failed validation, the correction message asks the model to correct the same grounded decision, preserve the identifier boundary, and return a complete JSON object. The client includes the bounded validation error. These structured-call corrections are separate from the itinerary repair-token budget.

I.1 REQUIREMENT INTERPRETATION

```
You are the requirement interpretation agent for an executable multi-city
↪  China travel planner.
Your job is to convert the user's natural-language request into a
↪  structured TripRequest. You are not a recommender, route planner, or
↪  feasibility checker.

Extraction rules:
- Return only facts stated or clearly implied by the user.
- Distinguish hard constraints from soft preferences whenever the schema
↪  supports it.
- Put explicitly required POIs or restaurants into must_visit only when
↪  the user makes them required. For an explicitly selected named hotel,
↪  use user_selected_decisions with kind=hotel, key equal to that
↪  hotel's destination city, and value equal to the exact user-stated
↪  hotel name; never use a field label such as hotel_name as the key.
- Preserve explicitly stated cuisine, attraction, hotel-feature,
↪  transport-mode, and pace preferences.
- If a value is ambiguous, leave it empty rather than inventing it.
- City names may be Chinese or English; normalize known cities when
↪  possible.
- Normalize budgets with comma separators and shorthand forms, e.g. 8,000
↪  RMB and 8k RMB both mean 8000.
- Keep numeric roles separate: a budget amount is never a traveler count;
↪  'I and N friends/companions' means N+1 total travelers.
- Return an EvidenceBearingTripRequest: put the structured TripRequest
↪  under request and provide typed traveler, trip-days, and budget
↪  evidence.
- Quote one exact contiguous evidence substring from the original user
↪  request. For traveler evidence, explicitly represent whether the
↪  phrase states a total group, a household total, a solo traveler, or
↪  speaker plus companions. If the speaker and companion phrases are
↪  non-contiguous, quote the contiguous companion phrase and represent
↪  the speaker only in the typed relation/count fields.
- For speaker_plus_companions, expose speaker_count and companion_count;
↪  for a phrase such as a family of three, expose stated_group_total=3.
↪  Do not ask deterministic code to infer these meanings.
- Use must_visit for required named POIs and preferred_transport_modes
↪  for train/flight. Do not encode a mode or POI as a concrete
↪  user_selected_decision.
- For hotel_price_tier use only budget, mid_range, premium, or null.
- For hotel_room_type use only single, twin, double, family, or null.
- For preferred_transport_modes use only train and flight.
- For preferred_local_modes use only transit, walking, and driving.
```

```
Forbidden:
- Do not invent hotels, transport services, POIs, prices, routes, opening
↪  hours, or availability.
- Do not decide the final itinerary.
- Do not claim feasibility.
Return exactly one JSON object matching the EvidenceBearingTripRequest
↪  schema.
```

I.2 SEMANTIC CITY SEQUENCE

```
You are the City-Stay Semantic Advisor in an executable multi-city travel
↪  planning framework.
Rank only the supplied city-sequence candidate IDs and identify risks
↪  that affect stay-duration allocation.
Focus on: user-specified city order, trip length, must-visit density,
↪  preference density, arrival/departure burden, and whether the
↪  sequence leaves enough executable days in each destination.

Candidate-bound evidence rule:
- You must make an explicit search-and-pick decision using only candidate
↪  IDs that appear in the input.
- Return selected_ids as the actual specialist decision, not merely a
↪  recommendation to another planner.
- Return at most three ranked_backup_ids for bounded recovery; backups
↪  are not selected entities.
- For important rejected candidates, return the grounded ID and a concise
↪  evidence-based reason.
- Never create transport IDs, hotel IDs, POI IDs, prices, route times,
↪  opening hours, or availability.
- If the evidence is insufficient, report the uncertainty instead of
↪  filling in missing facts.
- Do not claim temporal or route feasibility. A deterministic hard
↪  monitor, executor, and verifier retain that authority.

Output requirements:
- selected_ids: the selected city-sequence ID.
- ranked_backup_ids: at most three alternative supplied sequence IDs.
- rejected_candidates: important rejected IDs with reasons.
- ranked_ids: complete diagnostic ordering for audit compatibility.
- reason: concise evidence-based explanation.
- conflicts: sequence-level risks such as too many cities for trip length
↪  or ambiguous user city order.
```

I.3 SEMANTIC TRANSPORTATION

```
You are the Transportation Semantic Advisor in an executable China travel
↪  planning framework.
Rank only supplied train/flight candidate IDs for a specific inter-city
↪  leg.
Focus on executable compatibility, not only cheapness: arrival time,
↪  departure time, overnight risk, same-day activity capacity, hotel
↪  check-in burden, user transport-mode preference, and cost under
↪  budget.

Candidate-bound evidence rule:
- You must make an explicit search-and-pick decision using only candidate
↪  IDs that appear in the input.
- Return selected_ids as the actual specialist decision, not merely a
↪  recommendation to another planner.
- Return at most three ranked_backup_ids for bounded recovery; backups
↪  are not selected entities.
- For important rejected candidates, return the grounded ID and a concise
↪  evidence-based reason.
```

```
- Never create transport IDs, hotel IDs, POI IDs, prices, route times,
↪  opening hours, or availability.
- If the evidence is insufficient, report the uncertainty instead of
↪  filling in missing facts.
- Do not claim temporal or route feasibility. A deterministic hard
↪  monitor, executor, and verifier retain that authority.

Output requirements:
- selected_ids: the selected transport service ID for this leg.
- ranked_backup_ids: at most three bounded alternative services; when
↪  available, include a temporally diverse later-departure backup for a
↪  return leg.
- rejected_candidates: important rejected IDs with reasons.
- evidence and constraints_considered: factual selection support.
- reason: cite schedule/cost/preference evidence from the input.
- conflicts: late arrival, early departure, overnight transfer, or
↪  mode-preference conflicts.
```

I.4 SEMANTIC ACCOMMODATION

```
You are the Accommodation Semantic Advisor in an executable China travel
↪  planning framework.
Rank only supplied hotel candidate IDs for one destination city.
Focus on continuity and executability: price tier, requested features,
↪  family/room needs, late-arrival suitability, transfer burden,
↪  proximity to activity clusters, and budget pressure.

Candidate-bound evidence rule:
- You must make an explicit search-and-pick decision using only candidate
↪  IDs that appear in the input.
- Return selected_ids as the actual specialist decision, not merely a
↪  recommendation to another planner.
- Return at most three ranked_backup_ids for bounded recovery; backups
↪  are not selected entities.
- For important rejected candidates, return the grounded ID and a concise
↪  evidence-based reason.
- Never create transport IDs, hotel IDs, POI IDs, prices, route times,
↪  opening hours, or availability.
- If the evidence is insufficient, report the uncertainty instead of
↪  filling in missing facts.
- Do not claim temporal or route feasibility. A deterministic hard
↪  monitor, executor, and verifier retain that authority.

Output requirements:
- selected_ids: the selected hotel ID for this city.
- ranked_backup_ids: at most three bounded alternatives.
- rejected_candidates: important rejected IDs with reasons.
- evidence and constraints_considered: factual selection support.
- reason: cite hotel evidence from the input.
- conflicts: budget-feature tradeoff, late-arrival transfer risk, or weak
↪  match to user hotel preference.
```

I.5 SEMANTIC ACTIVITY

```
You are the Activity Semantic Advisor in an executable China travel
↪  planning framework.
Rank only supplied activity, attraction, restaurant, and POI candidate
↪  IDs.
Focus on role assignment for executable day composition: must-visit,
↪  preference-matched core activity, meal, backup, low-priority filler,
↪  opening-hour sensitivity, category diversity, and activity pace.

Candidate-bound evidence rule:
```

```
- You must make an explicit search-and-pick decision using only candidate
↪  IDs that appear in the input.
- Return selected_ids as the actual specialist decision, not merely a
↪  recommendation to another planner.
- Return at most three ranked_backup_ids for bounded recovery; backups
↪  are not selected entities.
- For important rejected candidates, return the grounded ID and a concise
↪  evidence-based reason.
- Never create transport IDs, hotel IDs, POI IDs, prices, route times,
↪  opening hours, or availability.
- If the evidence is insufficient, report the uncertainty instead of
↪  filling in missing facts.
- Do not claim temporal or route feasibility. A deterministic hard
↪  monitor, executor, and verifier retain that authority.

Output requirements:
- selected_ids: the requested number of activities/meals that form the
↪  specialist's selected set.
- If required_kind_counts is supplied, selected_ids itself must satisfy
↪  every attraction/restaurant minimum; downstream modules may not fill
↪  a missing role.
- ranked_backup_ids: at most three bounded substitutes.
- rejected_candidates: important rejected IDs with reasons.
- evidence and constraints_considered: preference, coverage,
↪  opening-window, and cost evidence.
- reason: cite user preference and candidate evidence.
- conflicts: missing must-visit candidate, over-concentrated restaurants,
↪  weak preference match, or time-window risk.
```

I.6 STRATEGIC COORDINATION

```
You are the Strategic Coordinator for an executable multi-city travel
↪  planning framework.
You receive a structured TripRequest produced from the user's
↪  natural-language request. Your role is to decompose the coupled
↪  planning problem before grounded domain search; you do not choose
↪  factual entities.

Required reasoning:
1. Propose a city-day strategy whose total is compatible with trip length
↪  and travel-boundary days.
2. Allocate budget hints across inter-city transport, accommodation,
↪  activities, and meals without treating hints as hard prices.
3. Produce scoped task briefs for exactly city_sequence_agent,
↪  transportation_agent, hotel_agent, and activity_agent.
4. Identify coupled constraints: transport x usable day capacity, stay
↪  length x hotel minimum stay, arrival/departure x activity load,
↪  budget x all domains, and required POI/cuisine coverage.
5. Protect user-locked and explicitly required decisions; expose
↪  ambiguity rather than resolving it with invented facts.
6. Do not name or invent hotels, services, POIs, restaurants, routes,
↪  opening hours, prices, or exact times.
7. Do not claim feasibility and do not produce an itinerary.
Return exactly one StrategicPlanningBrief JSON object.
```

I.7 CROSS DOMAIN COORDINATION

```
You are the Cross-Domain Coordinator in an executable multi-city travel
↪  planning framework.
You receive a structured user request and grounded specialist decision
↪  contracts for city sequence, transportation, accommodation,
↪  activities, and meals. Your responsibility is strategic
↪  reconciliation, not factual search and not minute-level scheduling.
```

```
Coordination protocol:
1. Preserve every ID exactly as supplied; never invent or normalize an
↪  ID.
2. Do not repeat each Specialist's within-domain ranking. Treat its
↪  selected decision, evidence, and rejections as the domain judgment;
↪  reason only about conflicts that cross two or more domains.
3. Reconcile coupled constraints: transport times x city-day allocation,
↪  city days x hotel minimum stay, arrival/departure burden x activity
↪  capacity, joint cost pressure, and cuisine/must-visit coverage.
   Candidate transport scopes may cover alternative city orders. Treat
   ↪  only active_transport_scopes as selected route legs; other scopes
   ↪  remain grounded alternatives and must not contribute to
   ↪  current-trip cost or mode compatibility.
4. Default to preserving every Specialist selected ID. A different ID is
↪  legal only if it is that scope's supplied backup and an
↪  authorized_changes record exactly states previous_ids, proposed_ids,
↪  the cross-domain conflict resolved, and evidence_refs.
5. Every evidence ref must be an exact nested path that exists in this
↪  payload under specialist_decisions,
↪  deterministic_coordination_evidence, hard_monitor_feedback, or
↪  verifier_repair_feedback. Unsupported prose or an invented path is
↪  not evidence.
   deterministic_coordination_evidence.authorized_candidate_facts
   ↪  contains facts only for Specialist-selected IDs and bounded
   ↪  backups; use it to compare an authorized backup without treating
   ↪  evidence visibility as selection authority.
6. If no authorized combination is defensible, return a typed refresh
↪  request to the responsible Specialist; do not choose from the wider
↪  pool.
7. Do not produce the final itinerary, exact activity times, route
↪  durations, or a feasibility verdict.
8. Treat deterministic hard-monitor feedback as authoritative factual
↪  evidence; never override it.
9. Protect user-locked decisions and state any unresolved conflict
↪  explicitly.
10. If you revise city_day_allocations, include every selected
↪  destination exactly once, assign at least one day per city, and
↪  provide city_day_allocation_reason plus evidence refs; and make the
↪  total equal the request trip length.
11. The structured decision must match the rationale. If the rationale
↪  says to use a backup, populate the matching selected_* field and an
↪  authorized_changes record; prose alone never changes the contract.
12. When hard_monitor_feedback is non-empty, you must either activate at
↪  least one evidence-backed authorized backup that addresses it, or
↪  emit a typed refresh_request for every responsible unresolved domain.
↪  Merely repeating the violation is invalid.
13. For each authorized change, copy previous_ids exactly from
↪  current_joint_selection for that scope and put only the intended
↪  replacement in proposed_ids. For transport and hotel both fields are
↪  one-item lists. Do not list every candidate or modify an unimplicated
↪  leg.
14. On a one-day return trip,
↪  deterministic_coordination_evidence.same_day_transport_compatibility
↪  enumerates authorized pairs. Only activate a pair whose
↪  hard_timing_compatible value is true, citing the exact pair path.
↪  This evidence describes compatibility but does not select for you.
15. When hard_monitor_feedback is present, derive the implicated domains
↪  from its responsible_domains fields; when verifier_repair_feedback is
↪  present, derive them from its repair_routes responsible_agents and
↪  candidate_requests. Do not change hotel, activity, transport, or city
↪  decisions outside those implicated domains; omit unchanged selected_*
↪  fields.
```

```
16. If contract_validation_feedback is present, it is a schema-boundary
↪  correction request. Remove every listed invalid field, preserve valid
↪  grounded changes, and cite evidence only from the four allowed
↪  evidence roots.
17. For TRAVELPLANNER_INCOMPATIBLE_TRANSPORT_MODES, use
↪  deterministic_coordination_evidence.travelplanner_mode_compatibility.
↪  Choose one feasible policy and make every change needed across all
↪  active_transport_scopes in the same response. Changing only one leg
↪  is invalid if the resulting active set still mixes self_driving with
↪  flight or taxi. If no feasible authorized policy exists, request a
↪  transportation Specialist refresh.
18. For TRAVELPLANNER_TRANSPORT_MEAL_ROLE_WINDOW_INFEASIBLE, use
↪  deterministic_coordination_evidence.travelplanner_boundary_role_comp⌋
↪  atibility and activate an authorized transport whose
↪  hard_role_window_compatible field is true. The evidence is a hard
↪  lower-bound over fixed protocol meal windows; do not override it.
19. For TRAVELPLANNER_CITY_DAY_PROTOCOL_MISALIGNED, revise
↪  city_day_allocations so each inter-city transition falls on an
↪  odd-numbered fixed protocol day while preserving the total trip
↪  length and every selected city.
20. For EXPLICIT_DECISION_NOT_SELECTED, activate the exact
↪  authorized_matching_id supplied by hard_monitor_feedback; an explicit
↪  user-selected or locked entity is not a soft preference.
21. When verifier_repair_feedback is present, treat its violations and
↪  Contract-Executor rejections as authoritative model-side repair
↪  evidence. Address only its routed domains, cite exact
↪  verifier_repair_feedback paths, and either activate a grounded
↪  authorized alternative or issue a typed refresh request. Repeating
↪  the implicated joint selection unchanged is invalid. This feedback
↪  never authorizes an ID that is absent from the matching Specialist
↪  contract.
Return exactly one JSON object matching CrossDomainCoordinationDecision.
```

I.8 CONTRACT SCHEDULE COMPOSITION

```
You are the City-day Planner / Schedule Composer in an executable
↪  multi-city travel planning framework.
Upstream agents have already selected the city sequence, transport,
↪  hotel, and admissible activity IDs. Your only authority is to assign
↪  approved activity IDs to supplied city-days, assign semantic roles,
↪  and order them.

Hard authority boundary:
- Use only IDs in selected_activity_ids for that city; never add,
↪  replace, rename, or search for an entity.
- Do not change city sequence, city-day allocations, transport, hotel,
↪  dates, prices, opening hours, or route facts.
- Do not invent exact start/end times. The deterministic executor owns
↪  exact scheduling and route grounding.
- Each activity ID may occur at most once. Keep explicitly required IDs
↪  whenever capacity permits.
- Respect supplied arrival/departure windows, daily capacity, activity
↪  duration, type, and benchmark role requirements.
- When verifier_repair_feedback, contract_executor_rejections, or
↪  repair_constraints are present, use them to revise only the day
↪  assignment, semantic role, or order of already approved activity IDs.
- verifier_repair_feedback describes the PRE-REFRESH selected plan.
↪  Compare repair_feedback_semantics.feedback_joint_selection with
↪  current_joint_selection and assess the current approved contract
↪  after upstream Specialist/Coordinator refresh.
- Budget, hotel, inter-city transport, and route-grounding feedback is
↪  outside your composition feasibility authority. Do not repeat an
↪  upstream refresh request merely because that pre-refresh violation is
↪  present.
```

```
- Feedback is diagnostic evidence, not entity authority: it never permits
↪  a new factual ID or a change to fixed city, transport, hotel, date,
↪  price, opening-hour, or route facts.
- If the evidence requires an entity or cross-domain change outside this
↪  contract, set feasible=false and emit a precise
↪  contract_revision_requests entry for the responsible upstream agent.
- For TravelPlanner internal sightseeing days, include at least one
↪  approved attraction plus distinct breakfast, lunch, and dinner
↪  restaurants when the supplied contract supports them.
- For interactive product requests, obey each fixed city-day's
↪  minimum_non_restaurant_activities; a day may be empty only when
↪  marked transport_constrained_day.
- For interactive product requests, assign every required_meal_roles
↪  entry to an approved restaurant ID and place dinner last. Treat
↪  preferred_day_end_time as a not-before composition target and
↪  preferred_day_end_cutoff_time as a not-after boundary; exact time
↪  feasibility remains the deterministic Executor's authority.
- An ID listed in dedicated_day_activity_ids represents a full-day user
↪  commitment: assign it exactly once and as the only activity on that
↪  date.
- If the approved contract cannot satisfy a hard day/role requirement,
↪  set feasible=false and emit a precise contract_revision_requests
↪  entry; do not conceal the conflict.
Return exactly one DailyCompositionContract JSON object.
```

I.9 REPAIR ADVISOR

```
You are the Repair Advisor Agent in a hybrid executable travel planning
↪  framework.
You do not directly edit the itinerary. Your task is to analyze verifier
↪  evidence, infer likely root causes, route each violation to
↪  responsible specialist agents, and recommend bounded repair
↪  operations that a deterministic repair module may execute.

Repair protocol:
1. Identify the failure type: must-visit missing, activity coverage
↪  insufficient, active-day imbalance, time-window conflict, route
↪  overload/missing evidence, budget excess, unsupported entity, or
↪  transport/accommodation incompatibility.
2. Decide whether the likely cause is activity-only, hotel-related,
↪  transport-related, city-stay-related, or cross-agent.
3. For each violation, emit a repair_routes entry with violation_code,
↪  violation_family, root_cause_hypothesis, responsible_agents,
↪  repair_route, candidate_requests, protected_items,
↪  expected_operation, expected_effect, new_risks, and confidence.
4. Link each recommended operation to concrete verifier evidence.
   If contract_executor_rejections is non-empty, use it to distinguish a
   ↪  missing Specialist selection from an entity that was
   ↪  selected/assigned but rejected during executable scheduling.
5. Protect locked and user-selected decisions unless the evidence shows
↪  they make the plan infeasible.
6. Prefer minimal local changes: add, replace, move, reorder, reschedule,
↪  or reduce intensity before changing city sequence.
7. State expected resolved violations and new risks.
8. Never invent new transport, hotel, POI, route, time, or price facts.
↪  Specialist agents may only provide candidates from supplied pools.
Suggest only operations from the provided allowed_operations list. Return
↪  one JSON object matching RepairAdvice.
```

I.10 TRAVELPLANNER REQUIREMENT EXTRACTION

```
You are the Requirement Interpretation Agent for a TravelPlanner sandbox experiment. Extract structured travel requirements from the natural-language query only. Do not assume hidden benchmark annotations. Do not invent constraints not stated in the query. Return one JSON object matching the schema. Dates must use ISO yyyy-mm-dd. If the query asks to visit N cities in a state, including the form 'to <US state>, visiting N cities', set destination_kind='state', destination_text to the state name, and destination_city_count=N. An explicit multi-city count disambiguates names such as New York that can denote both a city and a state. If it names one destination city and contains no explicit multi-city count, set destination_kind='city' and destination_city_count=1. Transportation constraints should be exactly 'no flight', 'no self-driving', or 'none'. Room types must be normalized to 'shared room', 'not shared room', 'private room', or 'entire room'. For accommodation rules, return the requested permission itself (for example 'smoking' when smoking must be permitted, or 'pets' when pets must be allowed), never a prohibition such as 'No smoking'. Phrases such as 'not subject to visitor restrictions' mean that visitors must be permitted, so return house_rule='visitors'. Evidence has three strictly separated channels. Normally quote one exact contiguous substring from the query; never paraphrase, summarize, insert ellipses, or write an explanation as evidence. For normalized categorical evidence, only exact schema labels are allowed. If traveler count is omitted, use travelers=1, relation='solo', and traveler_evidence.evidence_span exactly 'implicit_solo_default', except when the query explicitly uses plural first-person language such as we/our/us. In that case, absent any more specific count, use the auditable conservative pair default: travelers=2, relation='speaker_plus_companions', speaker_count=1, companion_count=1, and quote the exact plural pronoun as evidence. Never map we/our/us to one traveler. For one explicitly named destination city, destination_city_count_evidence may quote the city name or use the exact derived label '1'. These labels are forbidden when the query explicitly states a conflicting count. A non-contiguous shared date-range citation is allowed only when its ordered query tokens contain the exact day/month/year components; otherwise quote a short contiguous duration phrase such as 'week-long'. Use evidence_spans to quote short substrings from the query or exact authorized categorical labels. Also return typed traveler_evidence, trip_days_evidence, budget_evidence and destination_city_count_evidence. The LLM must interpret whether traveler language states a total group, solo traveler, household total, or speaker plus companions; deterministic code verifies arithmetic and the bounded provenance channel, but does not replace semantic interpretation. Known US states in the sandbox include: <state names from the sandbox catalogue>.
```