DYNAPLANBENCH:BENCHMARKING LONG-HORIZON AGENTIC PLANNING IN DYNAMIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based agents are increasingly expected to solve long-horizon tasks through interaction with dynamic real-world environments. However, existing benchmarks largely evaluate plan generation or task completion in static settings, or introduce dynamics only as isolated state changes and disruptions. Such settings do not fully capture a defining property of real-world execution: the environment evolves over time, and previously executed actions may create persistent commitments that continue to constrain future decisions. To address this gap, we introduce **DynPlanBench**, an executable benchmark for long-horizon agentic planning and execution in dynamic environments. Across 153 tasks spanning logistics, retail, and travel, DynPlanBench models two complementary dynamics: Time-Indexed Dynamics, where environment states and action outcomes vary with execution time, and Event-Driven Dynamics, where task-specific disruptions arise during execution. Together, these dynamics create a continuous planning-and-execution process, requiring agents to perform temporal planning over world states at future execution times and commitment-aware replanning under persistent commitments from prior execution. Evaluation of 20 frontier LLMs shows that even the best-performing model achieves only a 48.7% success rate. Further analysis reveals two recurring failure modes: global temporal inconsistency across dependent steps and commitment-state drift across successive repairs. These findings identify temporal consistency across evolving world states and commitment-state maintenance across successive plan repairs as key challenges for reliable long-horizon agents.

1 INTRODUCTION

Recent advances in large language models (LLMs) have improved their reasoning and tool-use capabilities (OpenAI, 2026; DeepSeek-AI, 2026; Qwen Team, 2026), accelerating the deployment of agents in real-world service scenarios (Yao et al., 2023; Schick et al., 2023; Liu et al., 2024). As agents move into such settings, they must plan and act over extended horizons in an evolving world rather than a static environment (Froger et al., 2026; Wen et al., 2026; Ma et al., 2026; Zhou et al., 2024; Trivedi et al., 2024). In real-world service tasks, relevant world states may evolve over time, requiring agents to reason about the world states expected at future execution times under dynamic conditions rather than rely on a static snapshot. Meanwhile, execution unfolds incrementally and may be disrupted after partial completion (Hui et al., 2026; Liu et al., 2026b;a), requiring agents to repair the remaining plan while respecting commitments accumulated through prior execution. Reliable agents in such environments therefore require two capabilities: temporal planning over evolving world states and commitment-aware replanning after partial execution.

However, a substantial gap remains between existing agent benchmarks and the systematic evaluation of long-horizon planning and execution in realistic dynamic environments. Existing planning benchmarks incorporate dates, time windows, deadlines, and cross-step temporal constraints (Ding et al., 2025; Shao et al., 2026; Zhang et al., 2026; Drouin et al., 2024), but typically treat time as a constraint for validating plan feasibility rather than as a factor that dynamically changes the underlying world state. As a result, they provide limited assessment of whether agents can reason about how world states evolve over time and plan against the conditions that will hold when actions are actually executed. In parallel, dynamic interactive benchmarks introduce asynchronous updates, tool failures, and cost changes to elicit plan revision (Froger et al., 2026; Wen et al., 2026; Hui et al., 2026; Liu et al., 2026b;c;a). However, these changes are typically introduced as individual

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.

perturbations rather than systematically modeled as task-specific event dynamics during execution, while persistent commitments created by completed actions are rarely treated as central constraints on subsequent plan repair. Consequently, existing benchmarks provide limited evaluation of whether agents can jointly perform reliable temporal planning over evolving world states and commitment-aware replanning after partial execution.

To systematically evaluate dynamic planning and execution, we develop a dynamic-environment modeling framework that captures how world-state evolution and persistent commitments jointly constrain planning and replanning. The framework models world-state evolution through two complementary mechanisms: Time-Indexed Dynamics capture systematic world-state changes as world time advances, whereas Event-Driven Dynamics capture contingent events that arise during execution and abruptly alter the world state. In parallel, the framework explicitly models persistent commitments created by executed actions—including time occupation, resource locking, financial costs, and cross-step dependencies—which continue to constrain subsequent execution and plan repair. Together, these mechanisms instantiate dynamic environments in which planning must account for both evolving execution-time conditions and constraints carried forward from prior execution.

Building on this framework, we introduce DynPlanBench, a benchmark for evaluating temporal planning and commitment-aware replanning in dynamic environments. DynPlanBench comprises 153 tasks across three domains: travel, retail, and logistics. Each domain provides a unified interactive environment with explicit world time, persistent domain state, and executable tools shared across tasks. Each task instantiates execution-time-dependent conditions, contingent events that alter the world during execution, and persistent commitments created by completed actions. These dynamics give rise to a continuous plan–execute–replan process, in which agents must plan against conditions at intended execution times, execute actions that modify the environment, and revise disrupted plans while carrying forward commitments from prior execution. To support fine-grained and consistent analysis, we provide task-specific, deterministic and verifiable judge functions that assess execution outcomes, plan feasibility, and plan–execution consistency.

We conduct an extensive evaluation of a diverse set of frontier proprietary and open-source language models. Our results show that even the best-performing model achieves only a 48.7% success rate on DynPlanBench, revealing a substantial gap between current agent capabilities and the requirements of reliable reasoning, planning, and execution in dynamic worlds. Performance further degrades consistently as either capability becomes more demanding: the average success rate drops from 32.6% to 17.5% as temporal difficulty increases, and from 37.5% to 16.9% as replanning difficulty increases. Further analysis reveals two representative failure patterns: local feasibility but global temporal breakdown, where individually feasible temporal decisions fail to remain globally consistent across dependent steps; and commitment-state drift, where agents fail to maintain coherent commitment states across successive changes and plan repairs. Together, these results identify global temporal consistency and commitment-state maintenance as key challenges for reliable long-horizon planning and execution in dynamic environments.

2 Related Work

Temporal Planning Benchmarks. Recent planning benchmarks evaluate natural-language and real-world planning across a range of practical service scenarios (Zheng et al., 2024; Xie et al., 2024; Shao et al., 2026; Zhang et al., 2026). These benchmarks increasingly capture tool-mediated information acquisition, realistic constraints, and long-horizon dependencies. More recent work (Wen et al., 2026; Ma et al., 2026) further studies agents under real-world time and changing temporal conditions. Nevertheless, temporal structure in existing benchmarks is typically represented through dates, scheduling constraints, time windows, and deadlines (Ding et al., 2025; Wen et al., 2026; Ma et al., 2026), treating time primarily as a constraint on the plan rather than as a source of systematic world-state evolution. DynPlanBench complements these efforts by explicitly modeling time-dependent world-state evolution throughout planning and execution.

Dynamic Replanning Benchmarks. Early tool-use benchmarks primarily focused on API-level correctness and tool-call execution (Li et al., 2023; Qin et al., 2024; Guo et al., 2024; Patil et al., 2025; Chen et al., 2025; Liu et al., 2025), while subsequent stateful benchmarks extend evaluation to persistent databases, domain policies, and interdependent tool use (Lu et al., 2025; Li et al., 2026; Barres et al., 2026; He et al., 2026; Chen et al., 2026). Recently, a growing body of work explic-

itly studies agents under changing environments and the resulting need for plan revision (Froger et al., 2026; Hui et al., 2026; Liu et al., 2026b;c;a). STT-Arena (Hui et al., 2026) injects spatio-temporal triggers that can invalidate an ongoing plan, while CostBench (Liu et al., 2026b), Ada-PlanBench (Liu et al., 2026c), and PlanBench-XL (Liu et al., 2026a) study adaptation to newly revealed constraints, cost changes, tool failures, and disrupted execution paths. These works establish dynamic adaptation and plan revision as important evaluation axes. However, persistent commitments created by completed service actions–such as reservations, payments, and resource allocations–are generally not treated as central constraints on subsequent plan repair (Wang et al., 2026; Farn & Shin, 2023; Fan et al., 2025; Wang et al., 2024). DynPlanBench complements these efforts by evaluating commitment-aware replanning after partial execution, where completed service actions remain active constraints on the feasible repair space.

3 DynPlanBench

Figure.1 provides an overview of DynPlanBench. In each task, agents must plan while accounting for world-state evolution as time advances, rather than relying on a static snapshot. When a discrete change disrupts a partially executed plan, the agent must repair the remaining trajectory while respecting persistent commitments created by previously completed actions, making replanning dependent on the execution history. In this section, we first formalize this plan–execute–replan lifecycle, and then describe the dynamic world model, task construction, and evaluation protocol.

3.1 Task Formulation

We model the interaction between an agent and a dynamic environment as a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}_{\Phi,\Psi}, r)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ are the state, action, and observation spaces; $\mathcal{T}_{\Phi,\Psi}$ is a family of transition kernels indexed by the Time-Indexed Dynamics Φ and Event-Driven Dynamics Ψ; and r is the reward function.

The interaction begins from an initial state $s_0 \in \mathcal{S}$, with an initial observation $o_0 \in \mathcal{O}$ that specifies the available tools and the initially observable world information. At interaction step k, let $h_k = (o_0, a_0, o_1, \ldots, a_{k-1}, o_k)$ denote the interaction history. The agent selects an action according to $a_k \sim \pi(\cdot \mid q, h_k)$. The action space comprises both tool-invocation actions and planning actions: $\mathcal{A} = \mathcal{A}_{\text{tool}} \cup \mathcal{A}_{\text{plan}}$. Here, $\mathcal{A}_{\text{tool}}$ includes operations such as querying, booking, modifying, and cancelling, whereas $\mathcal{A}_{\text{plan}}$ comprises the submission of an initial plan and subsequent plan updates.

Unlike static tool-use environments, state transitions in DynPlanBench depend not only on the current state and action, but also on world time and exogenous events. When a_k is executed at world time t_k, the environment transitions according to

$$s_{k+1} \sim \mathcal{T}_{\Phi(t_k),\Psi_k}(\cdot \mid s_k, a_k), \tag{1}$$

and returns an observation $o_{k+1} \in \mathcal{O}$ generated from the updated state. Here, $\Phi(t_k)$ denotes the time-dependent conditions evaluated at t_k, and Ψ_k denotes the event effects triggered or active at interaction step k. Together with the current state s_k and action a_k, they determine the resulting world state and the realized outcome of the action.

Event-driven changes may invalidate future plan segments after partial execution has created persistent commitments. Rather than restarting from the updated state, the agent must revise the remaining trajectory while carrying forward its execution history and active commitments. Each task therefore follows a recurrent *plan–execute–replan* process. The agent first submits an initial plan P^0, then executes actions in the dynamic environment. When a change invalidates the remaining plan, the agent revises it based on the resulting world state, execution history, and active commitments.

Temporal planning. Every task in DynPlanBench is time-dependent: world time is not merely contextual information nor a static scheduling constraint, but a dynamic variable that materially affects world states and action outcomes at execution time. The agent therefore cannot construct a valid plan by assuming that the world state remains unchanged across different time points. Instead, it must schedule actions based on the world conditions expected at their intended execution times.

Commitment-aware replanning. Completed tool actions may create persistent commitments, such as reservations, payments and resource allocations. When subsequent changes invalidate the current

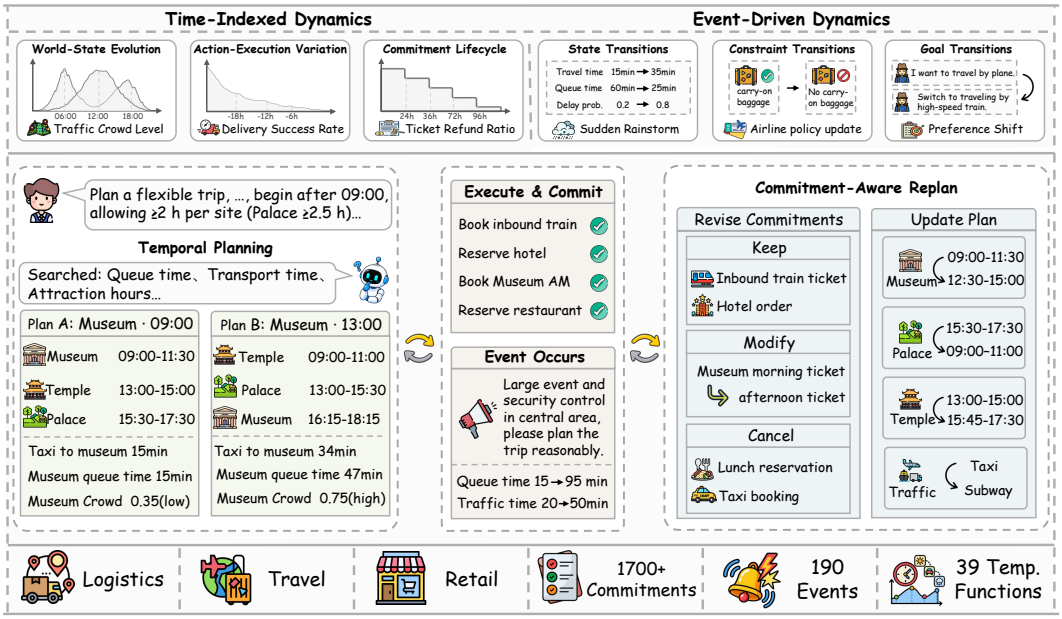

Figure 1: Overview of DynPlanBench. The benchmark models dynamic worlds with Time-Indexed and Event-Driven Dynamics. Agents plan for conditions at the intended execution time and repair disrupted plans by preserving, modifying or cancelling commitments created by prior actions.

plan, these commitments remain in effect, preventing the agent from simply discarding its prior execution and reconstructing a new plan from the updated world state alone. Instead, the agent must determine which existing commitments and plan segments can be preserved, modified, or cancelled, and appropriately repair the remaining trajectory under these persistent constraints.

3.2 Dynamic World Mechanisms

Time-Indexed Dynamics We implement the Time-Indexed Dynamics Φ as a collection of temporal functions over an advancing world-time axis to simulate systematic state changes in real-world environments. Each function takes static base data from the underlying database, world time t, and relevant environmental factors as inputs to compute the effective state at t.

To capture the different ways in which time affects a service world, we organize the Time-Indexed Dynamics into three components: $\Phi = (\Phi_{\text{state}}, \Phi_{\text{action}}, \Phi_{\text{commit}})$.

- **World-State Dynamics.** Φ_{state} determines the effective conditions at world time t from persistent domain data and time-dependent factors. It captures how task-relevant properties, such as availability, capacity, and service load, evolve even when no agent action modifies the database.
- **Action-Execution Dynamics.** Φ_{action} specifies how a tool action is realized when executed at world time t. Given the effective world state and the action arguments, it determines the action's success, duration, cost, and resulting state changes. Consequently, the same action may produce different outcomes when executed under different temporal conditions.
- **Commitment-Lifecycle Dynamics.** Φ_{commit} determines how commitments created by completed actions evolve after their creation. As world time advances, a commitment may become active, expired, consumed, or non-cancellable, while its modification or cancellation costs may change. These lifecycle changes alter the set of feasible repair operations available to the agent.

Event-Driven Dynamics We implement the Event-Driven Dynamics Ψ through task-specific event specifications executed by an event engine. During an episode, the engine monitors world time and the agent's trajectory. Once an event condition holds, it applies effects to the environment state and exposes, or withholds, the change according to the event's perception specification.

The Event-Driven Dynamics Ψ capture contingent incidents arising during task execution. Each event e comprises three components: $e = (\text{trigger}_e, \text{transition}_e, \text{perception}_e)$

- **Trigger.** The trigger specifies the runtime condition under which an event occurs. We distinguish time-, action-, and entity-query triggers, activated by a designated world time, a completed action, or an entity query, respectively. These triggers allow incidents to depend on both exogenous world progression and the agent's interaction trajectory.
- **World transition.** Once triggered, an event applies a discrete transition to the world state or task specification. We distinguish three transition types. *State Transitions* modify factual conditions in the service world; *Constraint Transitions* alter the feasible or permissible space for subsequent planning and execution; and *Goal Transitions* update the task objective to the agent. An event may involve one or more transition types, potentially invalidating part of the current plan, changing its preferred continuation, or requiring the agent to reassess the situation.
- **Perception.** The perception component determines how an event is exposed to the agent. A change may be announced through an environment or user message, or remain hidden until revealed by later tool interactions. In either case, the world transition occurs independently of recognition, and the agent must adapt from its observation history.

3.3 Task Curation

We construct DynPlanBench from real-world scenarios across three domains: travel, retail, and logistics, which are rich in temporal constraints, business commitments, and unexpected events.

Real-world scenario collection. We collect real user requests and service cases from public platforms, including travel itineraries and trip-change records, retail operation and exception-handling procedures, and logistics scheduling cases. We deduplicate similar materials to improve task diversity and prioritize cases that contain an initial objective, a feasible plan, and an unexpected event arising during execution. We preserve their temporal arrangements, costs, business rules, and post-event adjustment logic to derive an initial query, a reference plan, and a source-grounded event. All queries and events are manually reviewed and refined to ensure clear objectives, sufficient constraints, plausible events, and the existence of a feasible repair path after the change.

Executable environment construction. Given the query and reference plan, we use LLMs to assist in extracting the relevant entities, resources, schedules, and business attributes, while supplementing missing key information through web search. The extracted information is normalized into a unified schema to construct a domain-level static database. In addition to entities on the valid solution path, we include plausible alternatives that are excluded by specific constraints, requiring the agent to actively query and compare candidate solutions. We then instantiate task-specific Time-Indexed Dynamics and Event-Driven Dynamics, calibrated from real-world observations, operational policies, and historical event cases, and abstract API tools from real-world service applications, which operate directly on the resulting dynamic database.

Task validation. Each task undergoes both manual and executable validation. Human reviewers examine scenario realism, instruction quality, temporal dependence, and the semantic validity of unexpected events, while confirming that the task genuinely evaluates temporal planning and commitment-aware replanning. We then jointly test the tools, database, and dynamic mechanisms to verify that temporal changes and event effects meaningfully influence decisions and existing commitments. Finally, with LLM assistance, we construct and manually review a reference action trajectory for each task. Deterministic replay in the executable environment is used to verify task solvability and target-state reachability, together with non-degeneracy checks ensuring that the task cannot be completed by ignoring time-dependent conditions or required commitment handling.

3.4 Task Evaluation

We adopt *rule-guided deterministic evaluation*, assessing each task along three dimensions: execution correctness, plan feasibility, and plan–execution consistency, with strict reproducible programmatic verification inspired by recent planning benchmarks (Shao et al., 2026; Zhang et al., 2026; Lu et al., 2025; Trivedi et al., 2024). For each task, we construct an LLM-assisted and human-reviewed checklist based on the user query, dynamic environment, and required outcomes: $\mathcal{C} = \{c_1, \ldots, c_m\}$. We then compile these criteria into a task-specific set of deterministic checking functions, or-

Table 1: Performance comparison across overall benchmark and three domains. The leaderboard is sorted by the Avg@4 scores. The best and second-best results are shown in **bold** and underline.

Models	Overall			Logistics			Retail			Travel		
	Avg@4	Pass@4	Passˆ4	Avg@4	Pass@4	Passˆ4	Avg@4	Pass@4	Passˆ4	Avg@4	Pass@4	Passˆ4
Closed-Source LLMs												
Claude Opus 5	**48.7**	**68.0**	19.6	**46.7**	**68.9**	21.3	**54.7**	69.8	**20.8**	**43.6**	64.1	15.4
GPT-5.6 Sol	44.6	66.7	**20.9**	45.9	63.9	**26.2**	45.8	**71.7**	17.0	41.0	64.1	**17.9**
GPT-5.5	42.3	59.5	19.6	45.9	60.7	24.6	37.7	58.5	17.0	42.9	59.0	15.4
GPT-5.6 Terra	34.5	52.9	10.5	39.8	50.8	14.8	35.4	56.6	11.3	25.0	51.3	2.6
Qwen3.7-Max	33.2	51.0	13.1	34.0	45.9	16.4	32.1	52.8	9.4	33.3	56.4	12.8
GPT-5.6 Luna	30.9	48.4	10.5	32.8	49.2	9.8	34.4	49.1	15.1	23.1	46.2	5.1
Open-Source LLMs												
DeepSeek-V4-Pro	35.8	60.8	13.1	36.1	59.0	13.1	35.4	58.5	17.0	35.9	**66.7**	7.7
GLM-5.2	33.2	48.4	13.7	34.0	49.2	14.8	34.4	49.1	17.0	30.1	46.2	7.7
DeepSeek-V4-Flash	27.0	46.4	8.5	35.7	55.7	11.5	17.5	37.7	7.5	26.3	43.6	5.1
MiniMax M3	26.8	47.7	7.2	30.7	55.7	11.5	24.1	39.6	5.7	24.4	46.2	2.6
LongCat-2.0	24.7	41.2	5.9	28.3	50.8	9.8	17.0	28.3	1.9	29.5	43.6	5.1
Kimi K2.6	24.0	41.8	5.9	22.1	42.6	4.9	26.4	37.7	9.4	23.7	46.2	2.6
MiMo-V2.5-Pro	18.8	37.9	2.0	28.3	49.2	4.9	14.6	32.1	0.0	9.6	28.2	0.0
Hy3	15.0	28.8	2.6	22.5	44.3	4.9	12.3	20.8	1.9	7.1	15.4	0.0
More Efficient LLMs												
Qwen3.6-35B-A3B	17.2	34.0	3.3	27.5	55.7	6.6	10.4	18.9	0.0	10.3	20.5	2.6
Qwen3.6-27B	15.0	27.5	1.3	26.6	44.3	1.6	9.0	20.8	1.9	5.1	10.3	0.0
Gemma-4-31B-IT	12.6	22.2	3.9	20.5	36.1	4.9	9.4	17.0	3.8	4.5	7.7	2.6
Qwen3.5-27B	11.8	24.8	2.6	19.3	41.0	3.3	5.7	9.4	1.9	8.3	20.5	2.6
Gemma-4-26B-A4B-IT	3.4	9.8	0.0	8.2	23.0	0.0	0.5	1.9	0.0	0.0	0.0	0.0
GLM-4.7-Flash	2.1	6.5	0.0	3.7	13.1	0.0	0.9	1.9	0.0	1.3	2.6	0.0

ganized into three groups: $\mathcal{F} = \mathcal{F}_{\text{exec}} \cup \mathcal{F}_{\text{plan}} \cup \mathcal{F}_{\text{cons}}$, where $\mathcal{F}_x = \left\{f_1^x, \ldots, f_{p_x}^x\right\}, \quad x \in \{\text{exec}, \text{plan}, \text{cons}\}$. Here, $\mathcal{F}_{\text{exec}}$ contains checks of the final environment state and the handling of persistent commitment; $\mathcal{F}_{\text{plan}}$ contains checks of requirement coverage, temporal feasibility, and executability; and $\mathcal{F}_{\text{cons}}$ contains checks of consistency between the submitted plans, executed tool actions, and active commitments.

Each checking function returns a binary result for one atomic criterion, enabling fine-grained error localization and capability diagnosis. The final task reward is $r = \prod_{f_i \in \mathcal{F}} f_i$, such that a task is successful only when all task-specific checking functions pass.

4 EXPERIMENT

4.1 EXPERIMENTAL SETUPS

Models. We evaluate a diverse set of advanced LLMs, including proprietary models, frontier open-source models, and more efficient model families. The complete list of evaluated models is provided in Appendix B.1. For models that support configurable reasoning or thinking modes, we enable them following the corresponding official recommendations.

Implementations. All agents are implemented as function-calling agents based on the OpenAI tool schema. Each task is limited to 150 interaction turns and terminates when the agent outputs `"###STOP###"` or reaches the maximum turn limit. For the main evaluation, each task is run four times with a temperature of 0.0. Full agent prompts are provided in the Appendix B.2.

Metrics. We report Avg@4, Pass@4, and Passˆ4, averaged across tasks. Avg@k measures the average success rate across the k trials. Pass@k measures the probability that at least one trial succeeds (Chen et al., 2021). Passˆk measures the probability that all k trials succeed.

4.2 MAIN RESULTS

Table.1 presents comprehensive evaluation results on DynPlanBench. We can observe that:

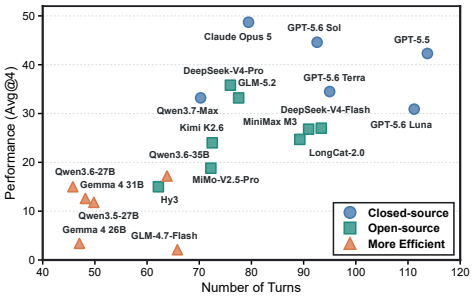

Figure 2: Model performance vs. Turns.

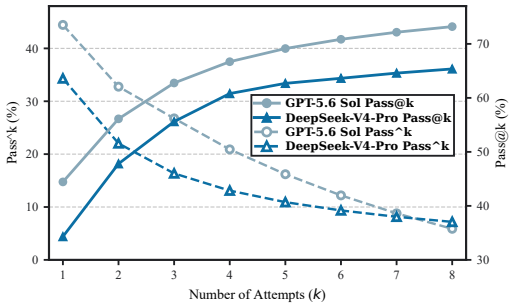

Figure 3: Pass@k vs. Passˆk performance.

Dynamic-world tasks remain challenging for frontier agents. As shown in Figure.2, closed-source models generally occupy the higher-performance region across a broad range of trajectory lengths, while open-source models tend to achieve lower Avg@4. However, the strongest models achieve Avg@4 below 50% (48.7% for Claude Opus 5 and 44.6% for GPT-5.6 Sol), while Passˆ4 further drops to around 20%. Although agents can satisfy most individual requirements, a single critical failure—such as insufficient transfer time or failing to revise an existing commitment during replanning—can invalidate the entire plan. Consequently, strong local correctness does not reliably translate into end-to-end success, indicating that temporal reasoning and commitment-constrained replanning under evolving world states remain significant challenges for current LLM agents.

Occasional success does not translate into reliable service. Pass@4 shows that additional sampling substantially increases the chance of finding at least one successful trajectory, but this gain does not translate into reliable repeated execution. Despite setting temperature to 0.0, cumulative perturbations in multi-turn interactions amplify into divergent trajectories. GPT-5.6 Sol achieves an Overall Pass@4 of 66.7%, while its Passˆ4 is only 20.9%. When extending to $k = 8$, as shown in Figure.3, the performance gains from additional sampling gradually saturate, whereas Passˆk continues to decline as k increases. This indicates that additional attempts improve the chance of occasionally discovering a feasible trajectory, without resolving the underlying instability of dynamic execution. For real-world services involving reservations, payments, and resource allocation, such a gap between occasional success and reliable execution remains a major barrier to deployment.

Domain differences reveal that explicit dynamic structures are increasingly tractable, while implicit and coupled dynamics remain challenging. Most models perform better in Logistics, with an average Avg@4 of 29.4% across models, compared with 22.9% in Retail and 21.3% in Travel. This advantage is pronounced for smaller models, while this cross-domain gap narrows for stronger models. This pattern reflects differences in how dynamic dependencies enter the decision process across domains. In Logistics, relevant dynamics are more directly tied to operational variables such as route times. But in Retail and Travel, dynamic dependencies are often implicit in plan feasibility, such as inventory and staffing states. Agents must therefore identify these dependencies and preserve their implications across both planning and replanning. The more balanced performance of stronger models suggests that they are better able to identify and reason over such implicit dependencies. These results indicate that the remaining challenge lies not only in temporal reasoning itself, but increasingly in identifying latent dynamic dependencies and correctly accounting for their cross-step consequences throughout planning and replanning.

5 Analysis & Discussion

5.1 Task Complexity Analysis

Temporal Planning Difficulty. We investigate how temporal structure affects task difficulty by stratifying tasks into three levels according to temporal constraint density, dependency depth, slack tightness, and time-indexed state evolution, with detailed definitions provided in the Appendix A.5. As shown in Figure.4, performance degrades substantially as temporal difficulty increases, with the average Avg@4 dropping from 32.6% on T1 to 23.6% on T2 and 17.5% on T3.

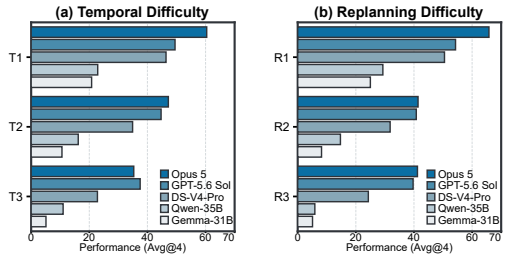

Figure 4: Performance across temporal and replanning difficulty levels.

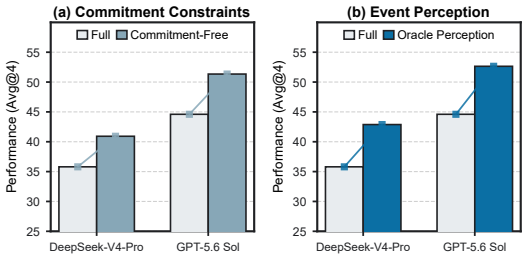

Figure 5: Ablation study of commitment constraints and event perception.

Importantly, the degradation is not uniform: a large portion of the performance loss already occurs between T1 and T2. This suggests that current agents can often handle localized temporal constraints, but struggle once multiple constraints become coupled across steps and must be maintained consistently throughout the plan. To further isolate the contribution of time-dependent world evolution, we replace Time-Indexed Dynamics with fixed baseline values while retaining Event-Driven Dynamics. As shown in Table.2, Avg@4 improves from 44.6% to 59.6% for GPT-5.6 Sol and from 35.8% to 48.7% for DeepSeek-V4-Pro. These results indicate that systematic time-dependent world-state evolution imposes an additional planning burden, particularly when execution-time conditions must remain jointly feasible across multiple plan steps.

Table 2: Ablation of dynamic world mechanisms. DS-V4 denotes DeepSeek-V4-Pro and Sol denotes GPT-5.6 Sol. Results are Avg@4 (%).

Setting	Temp.	Event	DS-V4	Sol
Full	✓	✓	35.8	44.6
w/o Temporal	×	✓	48.7	59.6
w/o Event	✓	×	58.8	64.1
Static	×	×	**62.7**	**77.8**

Commitment-Aware Replanning Difficulty. We further stratify the tasks into three replanning difficulty levels according to event complexity, commitment complexity, and perception complexity, with detailed definitions provided in the Appendix A.5. As shown in Figure.4, average Avg@4 across models drops from 37.5% to 21.0% and 16.9% on R1, R2, and R3 respectively. This degradation suggests that the challenge of commitment-aware replanning emerges once agents must coordinate repairs across multiple persistent business objects, while preserving unaffected commitments and maintaining consistency with the remaining plan and the overall goal. Further analysis of event count reveals that this difficulty compounds across successive changes: the pass rate drops from 26.8% for single-event tasks to 12.8% for two-event tasks. In particular, a second event requires the agent to adapt from the state produced by the first repair rather than from the initial state, exposing a *sequential adaptation bottleneck* in which agents may revisit cancelled objects, overwrite still-valid repairs, or miss the accumulated consequences of multiple events. As shown in Figure 5, automatically cancelling superseded orders without cancellation fees improves performance by 6.7 pp for GPT-5.6 Sol and 5.1 pp for DeepSeek-V4-Pro, while immediately revealing all event-induced changes yields gains of 8.0 pp and 7.0 pp, respectively. However, neither intervention closes the gap to the setting without Event-Driven Dynamics, indicating that commitment-aware replanning requires more than recognizing a new change: agents must maintain and revise persistent commitments coherently across successive repairs.

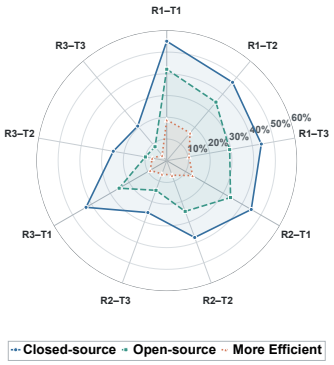

Figure 6: Avg@4 performance across the nine joint difficulty strata.

Compound Effect of Temporal and Replanning Difficulty. As shown in Figure.6, the joint difficulty stratification shows that the effects of temporal and replanning complexity compound when they co-occur. When both dimensions increase, model performance deteriorates most sharply: for closed-source models, the average pass rate drops from 55.0% on R1–T1 to 20.8% on R3–T3. This pattern suggests that the hardest tasks are not those with complex temporal structure or demanding

repairs alone, but those in which agents must repair existing commitments under evolving temporal dependencies while preserving the temporal feasibility of the remaining plan. To further examine this effect at the mechanism level, Table.2 removes Time-Indexed Dynamics and Event-Driven Dynamics separately. Performance improves when either mechanism is disabled, and reaches its highest level in the Static setting where both are removed. Taken together, these results suggest that temporal planning and commitment-aware replanning form a compound source of difficulty rather than two isolated challenges. A local temporal error can change the feasible space for subsequent repair, while repairing persistent commitments can in turn alter the temporal feasibility of downstream actions, allowing errors to propagate through the remainder of the plan.

5.2 Error Pattern Analysis

Overall Error Distribution. We manually inspect the complete failed trajectories of GPT-5.6 Sol and annotate unrecovered errors that materially contribute to task failure. We organize unrecovered errors in failed trajectories into a two-level taxonomy comprising four stages—Observe, Reason, Replan, and Execute—and ten corresponding fine-grained error types, and report their normalized composition in Figure.7. Observe accounts for only 7.9% of errors, whereas Reason and Replan together account for 68.8%, with failures concentrated in temporal consistency, commitment handling, and downstream propagation. Execute errors remain 23.3%, but are overwhelmingly due to argument grounding rather than tool selection or hallucination. Overall, the dominant bottleneck for these agents lies in maintaining consistency across time-dependent states, persistent commitments, and cross-step dependencies, rather than in basic information acquisition or tool selection.

Representative Failure Patterns. Further inspection of failed trajectories reveals two recurring error patterns. First, we observe *global temporal inconsistency*: agents can often handle an individual opening window, travel duration, or temporal constraint, yet fail to propagate its consequences to subsequent dependent steps. A locally plausible decision can therefore cascade into a globally infeasible schedule, revealing a lack of robust global temporal consistency under evolving execution-time conditions. Second, we identify *commitment-state drift*: after one or more repairs, agents fail to preserve the commitment state established by their previous actions and adaptations. They may revisit already-cancelled objects, overwrite still-valid repairs, or respond only to the latest event while overlooking the accumulated effects of earlier changes. Together, these patterns expose a common limitation of current agents: rather than failing primarily at individual local decisions, they struggle to carry the consequences of dynamic changes forward—across future steps for temporal planning and across successive repairs for commitment-aware replanning.

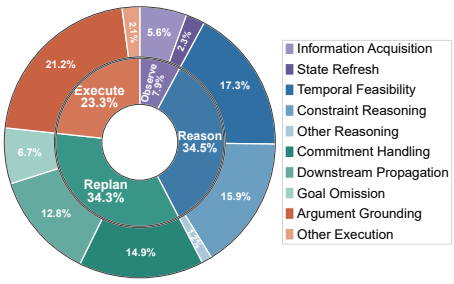

Figure 7: Error distribution of DynPlanBench.

6 Conclusion

In this work, we rethink the evaluation of LLM-based agents in dynamic environments and introduce DynPlanBench to systematically assess temporal planning and commitment-aware replanning. DynPlanBench models world-state evolution through Time-Indexed and Event-Driven Dynamics and explicitly represents persistent commitments created by executed actions, requiring agents to plan against conditions at future execution times and repair disrupted plans under prior commitments. Our broad evaluation of frontier proprietary and open-source models shows that even the strongest models remain far from reliable performance in dynamic worlds. Further analysis reveals persistent limitations in maintaining global temporal consistency across dependent steps and coherent commitment states across successive changes. Overall, our results suggest that the key challenge for real-world agents is shifting beyond local reasoning and tool execution in static settings toward temporal planning over evolving world states and commitment-aware replanning after partial execution. We hope DynPlanBench can provide a foundation for developing and evaluating robust LLM agents under dynamic conditions.

ETHICS STATEMENT

Our research fully complies with the ICLR Code of Ethics. We obtained all datasets in compliance with relevant usage guidelines, ensuring no violation of privacy. DynPlanBench is constructed from publicly available materials and adapted into self-contained benchmark scenarios. During task construction, we removed or anonymized all personally identifiable information (PII) contained in the source materials and avoided procedures that might compromise privacy or security. Throughout our investigation, we carefully avoided any bias or discriminatory effects. The resulting tasks, dynamic mechanisms, and reference trajectories are manually reviewed and validated.

REPRODUCIBILITY STATEMENT

We have prioritized making our findings fully reproducible. The complete codebase are provided in the supplementary materials and will be publicly released to facilitate independent verification and replication. We also describe the experimental settings and implementation details in the paper. We hope these resources will make it easier for the research community to reproduce our results, build upon our work, and further advance research in this area.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. In *ICML*, 2026.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Xinzhi Wang, et al. Acebench: A comprehensive evaluation of llm tool usage. In *EMNLP Findings*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv*, 2021.

Yuxin Chen, Yi Zhang, Zhengzhou Cai, Yaorui Shi, Zhiyuan Yao, Chenhang Cui, Jingnan Zheng, Yaqi Huo, Xi Su, Qi Gu, et al. Vitabench 2.0: Evaluating personalized and proactive agents in long-term user interactions. *arXiv*, 2026.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026.

Zifeng Ding, Sikuan Yan, Moy Yuan, Xianglong Hu, Fangru Lin, and Andreas Vlachos. Tcp: a benchmark for temporal constraint-based planning. In *EMNLP*, 2025.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? In *ICML*, 2024.

Shiqing Fan, Xichen Ding, Liang Zhang, and Linjian Mo. Mcptoolbench++: A large scale ai agent model context protocol mcp tool use benchmark. *arXiv preprint arXiv:2508.07575*, 2025.

Nicholas Farn and Richard Shin. Tooltalk: Evaluating tool-usage in a conversational setting. *arXiv preprint arXiv:2311.10775*, 2023.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, et al. Gaia2: Benchmarking llm agents on dynamic and asynchronous environments. In *ICLR*, 2026.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *ACL Findings*, 2024.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Hui Su, and Xunliang Cai. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. In *ICLR*, 2026.

Tingfeng Hui, Hao Xu, Pengyu Zhu, Hongsheng Xin, Kun Zhan, Sen Su, Chunxiao Liu, and Ning Miao. Stt-arena: A more realistic environment for tool-using with spatio-temporal dynamics. *arXiv*, 2026.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. 2026.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *EMNLP*, 2023.

Jiayu Liu, Qihan Lin, Cheng Qian, Rui Wang, Emre Can Acikgoz, Xiaocheng Yang, Jiateng Liu, Zhenhailong Wang, Xiusi Chen, Heng Ji, et al. Planbench-xl: Evaluating long-horizon planning of llm tool-use agents in large-scale tool ecosystems. *arXiv*, 2026a.

Jiayu Liu, Cheng Qian, Zhaochen Su, Qing Zong, Shijue Huang, Bingxiang He, and Yi R Fung. Costbench: Evaluating multi-turn cost-optimal planning and adaptation in dynamic environments for llm tool-use agents. In *ACL*, 2026b.

Jiayu Liu, Cheng Qian, Zhenhailong Wang, Bingxuan Li, Jiateng Liu, Heng Wang, Jeonghwan Kim, Yumeng Wang, Xiusi Chen, Yi R. Fung, and Heng Ji. Adaplanbench: Evaluating adaptive planning in large language model agents under world and user constraints. *arXiv*, 2026c.

Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, et al. Toolace: Winning the points of llm function calling. In *ICLR*, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *ICLR*, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *NAACL Findings*, 2025.

Yichuan Ma, Linyang Li, Yongkang Chen, Peiji Li, Xiaozhe Li, Qipeng Guo, Dahua Lin, and Kai Chen. Timely machine: Awareness of time makes test-time scaling agentic. In *ACL*, 2026.

OpenAI. GPT-5.6 System Card. OpenAI Deployment Safety Hub, July 2026. URL `https://deploymentsafety.openai.com/gpt-5-6`.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *ICML*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.

Qwen Team. Qwen3.7: The agent frontier. Qwen Technical Blog, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

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, et al. Chinatravel: an open-ended travel planning benchmark with compositional constraint validation for language agents. In *ICLR*, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *ACL*, 2024.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *ICLR*, 2024.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Parag Shah, Yujia Bao, et al. Mcp-bench: Benchmarking tool-using llm agents with complex real-world tasks via mcp servers. In *ICLR*, 2026.

Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-time reasoning agents in evolving environments. In *ICLR*, 2026.

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 *ICML*, 2024.

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

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints. In *ACL*, 2026.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *ICLR*, 2024.

AI USE STATEMENT

In this work, we used generative AI tools for aiding or polishing writing and generating synthetic datasets. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

A BENCHMARK DETAILS

A.1 BENCHMARK OVERVIEW AND STATISTICS

As shown in Table. 3, DynPlanBench comprises 153 tasks spanning three real-world service domains: Logistics, Retail, and Travel. Across these domains, it contains 127 tools and 4,356 structured entities. Tasks within each domain share a unified executable environment comprising domain databases, toolsets, and temporal functions. Individual tasks differ in user objectives, dynamic events, and commitment states. For fine-grained, deterministic evaluation, we define 4,869 task-specific Judge Rules assessing execution outcomes, plan feasibility, and plan–execution consistency.

A.2 TASK CONSTRUCTION AND VALIDATION

Scenario sourcing and task adaptation. We construct DynPlanBench through systematic adaptation of real-world service scenarios across Travel, Retail, and Logistics. Travel scenarios are derived from public itineraries and itinerary-adjustment records, Retail scenarios from industry operating procedures, training materials, and public operational cases, and Logistics scenarios from scheduling cases, cold-chain protocols, and industry reports. For each source scenario, we retain the core user objective, temporal arrangements, operational constraints, and post-disruption adjustment logic, and adapt them into a self-contained task instruction and reference scenario. When execution-critical information is absent from the source material, we supplement prices, opening hours, transport connections, and service restrictions through public information retrieval and manual verification. This process preserves the temporal dependencies and exception-handling structure of the original service cases rather than introducing dynamics solely as artificial sources of difficulty.

Executable environment and dynamic instantiation. For each adapted task, we further instantiate the corresponding real-world scenario as an executable environment. We first use LLM-assisted extraction, guided by the task requirements and reference scenario, to identify the entities, resources, schedules, and business attributes required for execution, and normalize them into a unified structured database for each domain. During this process, personally identifiable and sensitive information is removed or anonymized, while only the task-relevant operational structure is retained. To avoid reducing the environment to a single retrieval path along the reference solution, we include not only entities used by the reference solution but also plausible alternatives and distractors with complete business attributes. These candidates cannot be eliminated simply through missing fields; instead, their feasibility must be determined jointly from budgets, time windows, resource capacities, execution-time conditions, and subsequent dynamic changes. We then abstract domain tools from real service operations, allowing queries, creations, modifications, cancellations, and resource allocations to directly read from and update the underlying database and persistent commitments. Task instructions expose only the user's objectives, role information, and essential constraints, without directly revealing entity identifiers, tool names, or prescribed execution steps, thereby requiring the agent to identify feasible solutions through interaction with the environment.

After constructing the base executable environment, we further instantiate task-specific dynamic mechanisms on top of it. Time-Indexed Dynamics systematically vary states such as availability, duration, cost, risk, and quality with the intended execution time. Rather than manually specifying arbitrary perturbations, the dynamic factors and parameter ranges used in the executable environment are calibrated from real-world operational patterns and publicly available information. For each domain, we identify the factors that materially affect service outcomes—such as congestion, queueing, availability, pricing, freshness, capacity, and delay risk—and verify their dependencies and plausible ranges through public information retrieval and manual review. We then instantiate these factors as deterministic or procedural temporal functions so that the resulting dynamics remain executable, controllable, and reproducible across evaluations. Event-Driven Dynamics modify the realized world state during execution when predefined trigger conditions are satisfied. The event

Table 3: Core statistics of DynPlanBench across the three domains. Commit. denotes persistent commitment operations.

Domain	Tasks	Tools	Events	Commit.
Logistics	61	39	69	587
Retail	53	42	76	662
Travel	39	46	45	520
Overall	**153**	**127**	**190**	**1,769**

types, affected variables, and impact magnitudes are grounded in real-world service disruptions and operational cases, rather than synthetic changes introduced solely for increasing difficulty. These events are adapted from disruptions collected from real-world service scenarios, with their affected variables and plausible consequences refined through public information retrieval and manual verification before being instantiated as executable task-specific transitions. For commitment-aware replanning, events are designed to occur after partial execution has already established commitments such as reservations, orders, payments, or resource allocations, forcing the agent to repair the plan under the constraints imposed by its execution history rather than restart from the updated world state. In multi-event tasks, each subsequent event is applied to the world state produced by the preceding execution and repair, yielding a continuous plan–execute–replan process rather than a collection of independent perturbations.

Task validation. Each task undergoes a three-stage validation process. First, human reviewers assess scenario realism, instruction quality, temporal dependence, and the semantic validity of dynamic events, while verifying that the task genuinely requires temporal planning and commitment-aware replanning. Second, we jointly test the database, tools, and dynamic mechanisms to ensure that time-dependent states and event effects are correctly reflected in tool outcomes, persistent commitments, and subsequent decisions. Finally, with LLM assistance, we construct and manually review a reference action trajectory for each task and replay it deterministically in the same executable environment used for evaluation. This verifies target-state reachability and the existence of a feasible repair path after each disruption. We additionally perform non-degeneracy checks to ensure that tasks cannot be solved by ignoring critical changes or bypassing required commitment handling.

Table 4: Benchmark tools grouped by application domain.

Domain	Tool	Function
Logistics	Get Plan	Retrieve the current active dispatch plan.
	Submit Plan	Submit a complete structured dispatch plan before any order actions.
	Update Plan	Replace the current active dispatch plan with a revised version.
	Search Warehouses By Region	Search warehouses by region with optional zone type and minimum capacity filters.
	Search Warehouses By Capacity	Search warehouse zones with available capacity for a given goods type and quantity on a date.
	Search Vehicles By Region	Search vehicles by region with optional type and minimum capacity filters.
	Search Vehicles By Type	Search vehicles by type with optional temperature control filter.
	Search Customers By Region	Search customers by region with optional type filter.
	Search Suppliers By Region	Search suppliers by region with optional goods type filter.
	Search Routes	Search routes by origin with optional destination and vehicle type filters.
	Search Orders	Search orders across all types with optional filters.
	Search Hubs By Region	Search transfer/sorting hubs by region with optional type and capacity filters.
	Get Warehouse Info	Get basic information about a warehouse.

Continued on next page

Table 4 continued

Domain	Tool	Function
	Get Warehouse Capacity	Get real-time remaining capacity and throughput for a warehouse zone on a given date.
	Get Vehicle Info	Get basic information about a vehicle.
	Get Vehicle Availability	Check if a vehicle is available on a given date and optional time window.
	Get Customer Info	Get detailed information about a customer.
	Get Supplier Info	Get detailed information about a supplier.
	Get Hub Info	Get detailed information about a sorting/transfer hub.
	Get Route Details	Get real-time route duration estimate including congestion and weather effects.
	Get Delivery Estimate	Get delivery ETA, window feasibility, and success probability for a customer.
	Get Traffic Conditions	Get current traffic congestion and accident risk for a region.
	Get Weather Forecast	Query weather forecast for a region over a date range.
	Get Order Detail	Get full details of any order by its ID.
	Get Dispatch Policy	Get cancellation, priority, or dispatch scheduling policies.
	Create Delivery Order	Create a delivery order for a customer.
	Create Pickup Order	Create a pickup order to collect items from a supplier.
	Create Vehicle Assignment	Create a vehicle route assignment.
	Create Warehouse Transfer	Create a warehouse-to-warehouse transfer.
	Modify Delivery Order	Modify delivery address, time window, or priority of an existing delivery order.
	Modify Pickup Order	Modify pickup time or vehicle of an existing pickup order.
	Modify Vehicle Assignment	Modify route, time, or order set of an existing vehicle assignment.
	Modify Warehouse Transfer	Modify quantity or vehicle of an existing warehouse transfer.
	Cancel Delivery Order	Cancel a delivery order.
	Cancel Pickup Order	Cancel a pickup order.
	Cancel Vehicle Assignment	Cancel a vehicle assignment.
	Force Terminate Vehicle Assignment	Force-stop an active vehicle assignment after a real disruption event has triggered.
	Cancel Warehouse Transfer	Cancel a warehouse transfer.
Retail	Get Plan	Retrieve the current active operations plan.
	Submit Plan	Submit a complete structured operations plan before any order actions.
	Update Plan	Replace the current active operations plan with a revised version.
	Search Products By Category	Search products by category, sub-category, or storage type.
	Search Suppliers By Type	Search suppliers by supply category or product ID.
	Search Staff By Skill	Search staff members by skill type or availability time.
	Search Inventory By Zone	Search inventory by zone, category, or freshness status.
	Search Orders	Search orders across all types with optional filters.
	Search Equipment By Store	Search store equipment with optional store, type, and zone filters.
	Search Zones By Store	Search store zones with optional store, temperature zone type, and category filters.
	Get Product Info	Get detailed information about a product.
	Get Supplier Info	Get detailed supplier information, optionally with current availability for a delivery date or exact delivery deadline.
	Get Staff Info	Get detailed staff information, optionally with current efficiency if shift start is provided.
	Get Equipment Info	Get detailed equipment information.
	Get Inventory Status	Get detailed inventory status including freshness information.
	Get Zone Info	Get zone details including current inventory summary.

Continued on next page

Table 4 continued

Domain	Tool	Function
	Get Customer Flow Forecast	Get customer flow forecast for upcoming hours.
	Get Freshness Report	Get a freshness report for inventory items, sorted by urgency.
	Get Staff Schedule	Get staff schedule for a specific date.
	Get Staff Efficiency	Query a staff member's current work efficiency considering fatigue, time-of-day rhythm, and skill matching.
	Get Pricing Analysis	Analyze pricing elasticity for a product at a given discount level.
	Get Order Detail	Get detailed information about any order.
	Get Store Policy	Get store operational policies.
	Get Calendar Info	Get calendar, weather, and holiday information.
	Create Purchase Order	Create a purchase order to a supplier.
	Create Stock Placement	Place inventory items onto a store zone shelf.
	Create Price Adjustment	Adjust the price of a product.
	Create Inventory Markdown	Mark down a specific inventory batch without changing the product-wide base price.
	Create Promotion Campaign	Create a promotional campaign for products.
	Create Shift Assignment	Assign a staff member to a shift.
	Create Waste Disposal	Dispose of expired or substandard inventory.
	Record Zone Inspection	Persist completion of a zone inspection after reviewing current zone evidence.
	Activate Pickup Point	Activate an eligible store zone as a temporary pickup point.
	Modify Purchase Order	Modify a purchase order's planned quantity/date, or record the actual received quantity without overwriting the ordered quantity.
	Modify Shift Assignment	Modify a shift assignment's times, role, or zone.
	Modify Promotion Campaign	Modify a promotion campaign's discount or time window.
	Modify Price Adjustment	Modify a price adjustment.
	Cancel Purchase Order	Cancel a purchase order.
	Cancel Shift Assignment	Cancel a shift assignment.
	Cancel Promotion Campaign	Cancel a promotion campaign.
	Cancel Price Adjustment	Cancel a price adjustment and restore original price.
Travel	Get Plan	View the active plan or a specific previously submitted plan.
	Submit Plan	Submit a complete structured itinerary before any booking actions.
	Update Plan	Replace a previously submitted itinerary with a new full itinerary after new information or environment changes require replanning.
	Cancel Attraction Order	Cancel an attraction order and return its refund outcome under the current environment time.
	Cancel Hotel Order	Cancel a hotel order and return its refund outcome under the current environment time.
	Cancel Restaurant Order	Cancel a restaurant reservation and return its refund outcome under the current environment time.
	Cancel Transport Order	Cancel a transport booking and return its refund outcome under the current environment time.
	Create Attraction Order	Book attraction tickets.
	Create Hotel Order	Create a hotel booking.
	Create Restaurant Order	Create a restaurant reservation for reservable table types.
	Create Transport Order	Book a transport ticket.
	Get Attraction Base Info	Get attraction base information, operating rules, and explicit bookable ticket type IDs.
	Get Attraction Crowd	Get real-time crowd index of an attraction considering time, weather, and holidays.
	Get Attraction Inventory	Check real-time ticket availability and inventory status for a specific date.

Continued on next page

Table 4 continued

Domain	Tool	Function
	Get Attraction Price	Calculate real-time prices with holiday, booking time, and simulated inventory pressure adjustments.
	Get Attraction Queue	Calculate estimated queue time for an attraction.
	Get All Transport Duration	Compare taxi, bus, subway, and walk between two locations at a single time or across a time range.
	Get Booking Policy	Check booking, refund, and cancellation policy from either an existing order or a pre-booking preview.
	Get Hotel Base Info	Get room types and general information of a hotel.
	Get Hotel Inventory	Check if a room type is available for the entire stay duration.
	Get Inventory Pressure	Estimate inventory tightness for hotels, attractions, restaurants, or transport products.
	Get Hotel Price	Calculate total stay price considering holidays, proximity heat, and length-of-stay discounts.
	Get Intercity Transport Delay Risk	Get delay probability and risk level for a daily recurring flight or train on a specific travel date.
	Get Intercity Transport Details	Get real-time seat availability, dynamic pricing, and schedule details for a daily recurring flight or train on a specific travel date.
	Get Local Transport Price	Estimate local taxi, bus, subway, or walking cost between two locations at a given time.
	Get Local Transport Reliability	Estimate how stable a local commute is by sampling durations around a target departure time.
	Get Order Detail	Get a full order record, normalized summary, and booking policy by order_id.
	Get Restaurant Base Info	Get general info, opening hours, and table types of a restaurant.
	Get Restaurant Inventory	Check table availability for a restaurant on a given date.
	Get Restaurant Queue	Calculate estimated queue time for a restaurant based on time of day, weather, and holidays.
	Get Transport Duration	Calculate local travel time between two IDs for one mode at one concrete itinerary time.
	Get Transport Hub Info	Get location and suggested arrival buffer for one transport hub.
	Get Transport Price	Get dynamic ticket prices for one daily recurring transport schedule on a specific travel date.
	Get Weather Forecast	Query weather forecast for a city over a date range.
	Search Attractions By City	Find attraction IDs and info in a specific city.
	Search Attractions By Keywords	Search attractions by keywords over attraction name, tags, and description.
	Search Hotels By Attraction	Find hotels near a specific attraction.
	Search Hotels By City	Search hotels by city with optional district and minimum star rating filters.
	Search Hotels Near Location	Search hotels near any location_id with optional radius and star filters.
	Search Orders	Search the current user's existing orders across transport, hotel, attraction, and restaurant bookings.
	Search Restaurants By City	Search restaurants by city with optional district, cuisine keyword, and price range filters.
	Search Restaurants By Location	Search restaurants near any location_id with optional keyword and radius filters.
	Search Transport	Search daily recurring bookable flights and trains between two cities.
	Search Transport By Constraints	Search transport schedules with departure/arrival windows, type filters, and delay-risk constraints.
	Search Transport Hubs	Search transport hubs such as airports and railway stations in a city.

A.3 Dynamic World Implementation

A.3.1 Event-Driven Dynamics

Event-Driven Dynamics captures discrete, task-specific runtime changes that arise during task execution. Each event is jointly defined by a trigger condition, one or more transitions to the world state or task specification, and a perception mechanism. DynPlanBench contains 190 dynamic events that collectively produce 568 transition effects. A single event can affect multiple interdependent state dimensions simultaneously, rather than merely changing the availability of one entity. The covered dynamics differ markedly across domains. Figure.8 summarizes the coverage of dynamic dimensions across the three domains.

DynPlanBench manages Event-Driven Dynamics through a unified mechanism that continuously monitors three trigger types. **Time-based triggers** activate when world time satisfies predefined conditions. They simulate time-dependent external disturbances, such as weather changes and resource-state updates. **Action-based triggers** activate after a specified business operation completes successfully. They may additionally match relevant entities or key parameters, allowing events to occur at execution points with explicit business semantics. **Entity-based triggers** activate when the agent queries or accesses a specified entity, modeling localized interactive feedback, such as user rejection of a queried entity or the revelation of entity-specific preferences, rather than introducing additional exogenous disturbances. Additional queries therefore mainly expose task-relevant information rather than systematically increasing event severity. Each trigger fires only once after its condition is satisfied, and the occurrence is recorded in the current execution state. If a scheduled time-based event has not yet fired when the agent attempts to terminate, termination is deferred. The environment advances to the event's scheduled time, applies the event, returns the resulting observation to the agent, and resumes the interaction. This prevents an agent from bypassing scheduled dynamics by terminating the task early. Once triggered, an event applies its specified transitions to the executable world state or task specification. Its effects can include changes to entity attributes, resource or service conditions, temporally scoped environmental states, traffic conditions, feasible or permissible constraints, and task objectives. A single event may apply several related transition effects simultaneously. Subsequent tool queries and business operations therefore operate under the updated world and task conditions rather than receiving only a natural-language notification. Transition effects are decoupled from agent perception. Some changes are exposed directly through event or user messages, whereas others update only the underlying state or provide limited high-level information. The agent must then query the relevant entities again to determine the full effects.

A.3.2 Time-Indexed Dynamics

Time-Indexed Dynamics captures states and action outcomes that vary systematically with time in service environments. Domain-shared temporal functions compute these quantities dynamically for each intended execution time. The three domains model distinct temporal factors. In Logistics, these functions cover regional congestion, route duration and cost, delivery ETAs and time-window feasibility, vehicle availability and failure risk, warehouse occupancy and throughput, and weather. In Retail, they capture hourly customer traffic, product freshness, dynamic price elasticity, employee efficiency, and supplier fulfilment capacity. In Travel, they cover attraction opening status, ticket availability, prices, crowding and queues; hotel check-in availability and dynamic room rates; local transit; restaurant queues and reservation-slot availability; intercity ticket availability and delays; and weather and holiday conditions. Table.5 summarizes representative temporal functions, dynamic outputs, and primary influencing factors in each domain.

At runtime, each temporal function combines static entity attributes with the target time and relevant environmental factors to compute the effective state at that time. Dynamic attributes are therefore not fixed in advance in the database. Multiple functions may also form dependency chains. In Logistics, for example, departure time first determines the applicable regional congestion and weather conditions. These factors then alter route duration, transport cost, and delay risk, which ultimately affect delivery ETAs and the feasibility of customer time windows. Consequently, the same entity or action may differ in availability, duration, cost, and service quality across planned execution times. The agent must therefore plan against the expected execution time rather than the current world-state snapshot.

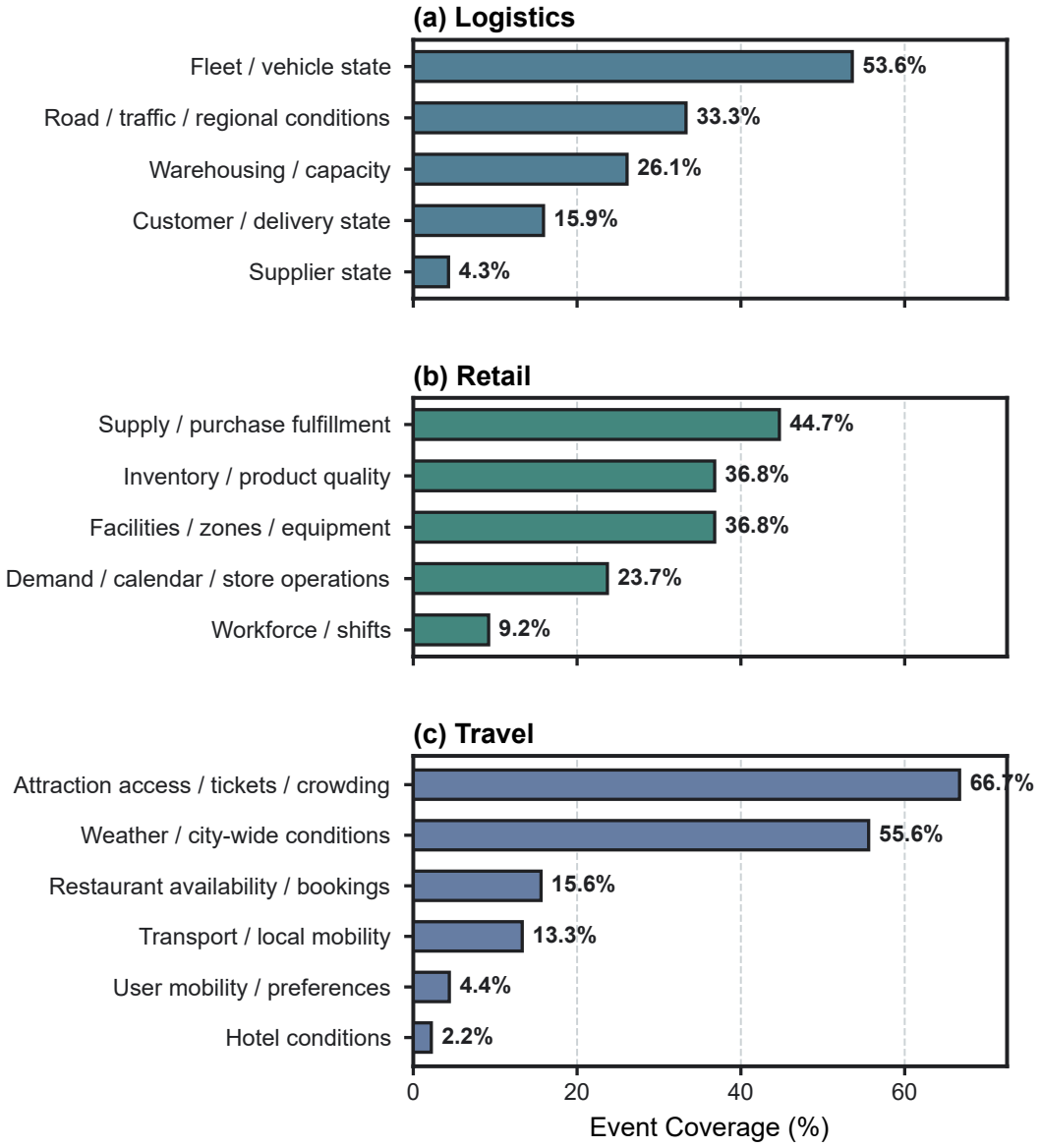

Figure 8: Coverage of event-driven state, constraint, and goal changes across the three application domains.

Table 5: Time-indexed functions grouped by application domain.

Domain	Temporal Function	Dynamic Output	Main Factors
Logistics	Congestion Index	Regional congestion index	Hour of day; weekday/weekend; commuter and logistics peaks; regional baseline and peak coefficients; weather; events.
	Route Duration	Route travel duration	Route configuration; vehicle type; distance; congestion; weather; departure time; event overrides.
	Accident Risk	Accident probability	Regional baseline risk; weather severity; congestion.
	Route Cost	Route transportation cost	Distance; vehicle characteristics; load; fuel consumption; congestion; temporally varying route conditions.
	Route Delay Risk	Route delay risk	Departure time; origin and destination; vehicle type; congestion; weather; active events.
	Delivery ETA	Estimated delivery arrival time	Departure time; route duration; warehouse processing time; supplier preparation time.
	Delivery-Window Feasibility	Delivery-window feasibility	Estimated arrival time; customer weekday/weekend delivery windows; target date.
	Delivery Success	Delivery success probability	Customer reliability; query time; alignment between expected arrival and delivery window.
	Remaining Driving Hours	Remaining daily driving hours	Current date and time; active assignments; daily driving limits; continuous-driving limits.
	Breakdown Risk	Vehicle breakdown probability	Time since maintenance; maintenance cycle; weather; vehicle-specific baseline risk.
	Vehicle Availability	Vehicle availability over requested interval	Requested date and time interval; vehicle status; maintenance schedule; overlap with active assignments.
	Warehouse Occupancy	Simulated warehouse occupancy	Target date; advance days; seasonality; warehouse capacity; baseline occupancy.
	Warehouse Throughput	Warehouse throughput and residual capacity	Zone capacity; simulated occupancy; committed allocations; weather; events; target date.
	Warehouse Queue Time	Loading/unloading queue time	Current hour; active assignments; dock count; vehicle type; peak operating windows.
	Weather State	Date-conditioned weather state	Day of year; seasonal parameters; deterministic variation; special-date overrides.
Retail	Customer Flow	Hourly customer-flow forecast	Time of day; three demand peaks; weekday/weekend; weather and precipitation; holidays; local-event multiplier; store operating hours.
	Product Freshness	Inventory-batch freshness	Elapsed time; product-specific decay parameters; current and storage temperatures; piecewise exposure history; initial freshness.
	Pricing Elasticity	Dynamic pricing elasticity	Product; discount level; current time; freshness; demand and pricing parameters.
	Staff Efficiency	Staff working efficiency	Current time; shift start; hours worked; assigned role; employee skill and current state.
	Supplier Availability	Supplier capacity and delivery feasibility	Ordering window; order time; deadline; seasonal capacity; weather; lead time; urgency; reliability; minimum order quantity.
Travel	Attraction Inventory	Remaining attraction-ticket inventory	Attraction capacity; visit date; booking horizon; weekday/weekend; seasonality; existing reservations; event overrides.
	Attraction Opening	Attraction opening status	Daily opening hours; weekday; fixed closure day; temporary closures; timed event overrides.
	Attraction Price	Dynamic attraction-ticket price	Ticket tier; visit date; demand and seasonality; inventory pressure; pricing policy; event overrides.

Continued on next page

Table 5 continued

Domain	Temporal Function	Dynamic Output	Main Factors
	Crowd Index	Attraction crowd index	Time of day; weekday/weekend; seasonality; weather; attraction popularity; local events.
	Queue Time	Expected attraction queue time	Crowd index; attraction service rate; time slot; active queue overrides.
	Crowd/Queue State	Conditioned crowd and queue state	Attraction identity; date and time interval; world-state overrides; active events.
	Hotel Occupancy	Hotel occupancy and remaining rooms	Room capacity; check-in date; booking horizon; weekday/weekend; seasonality; existing reservations; event overrides.
	Hotel Price	Dynamic hotel-room price	Base room rate; date; occupancy; booking horizon; weekend/holiday effects; seasonality; event multiplier.
	Local Congestion	Local traffic congestion index	Hour of day; weekday/weekend; commuter peaks; region; weather; events.
	Commute Duration	Mode-specific local travel duration	Origin–destination baseline or geospatial fallback; travel mode; distance; detour; waiting and access penalties; congestion; weather; events; temporal and regional overrides.
	Restaurant Opening	Restaurant opening status	Weekday-specific service periods; meal period; closure day; special-date and event overrides.
	Restaurant Queue Time	Expected restaurant queue time	Meal period; current time; weekday/weekend; popularity; reservations; event and crowd multipliers.
	Restaurant Inventory	Remaining table inventory	Table capacity; date; meal slot; existing reservations; demand pattern; event overrides.
	Slot Availability	Reservation-slot availability	Restaurant operating hours; slot definitions; table type; existing reservations; capacity; date-specific closures.
	Seat Inventory	Remaining intercity-transport seats	Schedule capacity; travel date; seat class; booking horizon; reservations; demand and event overrides.
	Transport Price	Dynamic intercity-transport fare	Base fare; seat class; travel date; booking horizon; inventory pressure; seasonal and event multipliers.
	Transport Delay Risk	Intercity-transport delay probability	Transport type; route; travel date and time; weather; seasonal baseline; active disruption events.
	Weather State	Date-conditioned weather state	Day of year; city-specific seasonal parameters; deterministic variation; special-date overrides.
	Holiday Calendar	Holiday status and holiday type	Calendar date; fixed and movable holiday definitions.

A.4 Evaluation Protocol

Evaluation evidence. For each completed rollout, evaluation is performed over the execution evidence recorded by the environment. This evidence includes tool calls and their returned results and arguments, the timing and ordering of actions, triggered environment events, persistent business states produced during execution, submitted plans, and the commitments that remain active at termination. Active commitments include reservations, orders, resource allocations, task assignments, and other persistent business objects created or modified through executable tools. Accordingly, task completion is determined from the agent's observable effects on the environment rather than from self-reported claims in its final natural-language response.

Evaluation dimensions. We organize the evaluation criteria along three complementary dimensions: *execution correctness*, *plan feasibility*, and *plan–execution consistency*. **Execution correctness** measures whether the required state transitions actually occurred, including the creation, mod-

ification, cancellation, or preservation of relevant commitments. **Plan feasibility** measures whether the final active plan remains executable under the realized event history and the time-indexed conditions at each segment's intended execution time. **Plan–execution consistency** measures whether the final plan and realized execution describe the same operational outcome: active commitments must be reflected in the plan, whereas cancelled or replaced objects must no longer be relied upon.

Rule-based deterministic verification. DynPlanBench implements these criteria as task-specific rules evaluated over structured execution evidence. In the implementation, rules are organized into two registries, execution rules and plan rules. Plan–execution consistency rules are implemented within plan rules, since they jointly inspect the final active plan and execution evidence. Execution rules verify business-object existence and status, required fields and quantities, budgets and refunds, resource assignments, and event-relative action timing. Plan rules verify required or forbidden segments, time windows, durations, ordering, reachability, overlap, and resource compatibility; they also enforce plan–execution consistency through rules such as commitment coverage. When a task admits multiple valid recovery strategies, the Judge accepts any branch that satisfies the required conditions. DynPlanBench allows diverse plans and action trajectories that are plausible in real-world service settings: an agent is not required to follow a particular sequence of actions as long as its plan and execution satisfy the task requirements, respect the specified constraints, and avoid explicitly prohibited outcomes or actions. A rollout is successful only if all applicable checks pass.

Deterministic Rule-Based Validator

```
from typing import Any, Callable

EXEC_RULES: dict[str, Callable] = {}
PLAN_RULES: dict[str, Callable] = {}


def evaluate(evidence, rule, registry, plan=None):
    """Evaluate one deterministic predicate."""
    kind = rule["rule_type"]

    # Accept any explicitly valid recovery branch.
    if kind == "any_of":
        return any(
            all(evaluate(evidence, r, registry, plan)
                for r in branch)
            for branch in rule["alternatives"]
        )

    # Apply event-specific rules only if the event occurs.
    if kind == "if_event_triggered":
        triggered = any(
            e["event_id"] == rule["event_id"]
            for e in evidence["events"]
        )
        return (not triggered) or all(
            evaluate(evidence, r, registry, plan)
            for r in rule["rules"]
        )

    handler = registry.get(kind)
    if handler is None:
        return False

    try:
        return handler(evidence, rule, plan)
    except Exception:
        return False  # fail closed
```

```python
def commitment_coverage(evidence, rule, plan):
    """Match persistent commitments to final-plan segments."""
    commitments = [
        c for c in evidence["commitments"]
        if c["order_type"] == rule["order_type"]
        and all(
            c.get(k) == v
            for k, v in rule.get("order_filters", {}).items()
        )
    ]

    segments = [
        s for s in plan["itinerary"]["segments"]
        if s.get("segment_type") == rule["segment_type"]
    ]

    def covered(commitment):
        return any(
            all(
                segment.get(k) == commitment.get(k)
                for k in rule.get("match_fields", [])
            )
            for segment in segments
        )

    return (
        len(commitments) >= rule.get("min_count", 1)
        and all(covered(c) for c in commitments)
    )


PLAN_RULES["commitment_coverage"] = commitment_coverage


def run_judge(evidence: dict[str, Any],
              spec: dict[str, Any]) -> float:
    """Return the binary task-level score."""
    execution = [
        evaluate(evidence, rule, EXEC_RULES)
        for rule in spec.get("predicate_rules", [])
    ]

    plan_rules = spec.get("plan_rules", [])
    plan_id = evidence.get("active_plan_id")
    plan = evidence["plans"].get(plan_id) if plan_id else None

    if plan_rules and plan is None:
        return 0.0

    planning = [
        evaluate(evidence, rule, PLAN_RULES, plan)
        for rule in plan_rules
    ]

    passed = (
        bool(execution)
        and all(execution)
        and all(planning)
    )
    return float(passed)
```

This example illustrates abridged evaluator logic used in DynPlanBench. Task-specific rules operate on structured execution evidence; `any_of` permits alternative valid re-

Table 6: Replanning difficulty levels.

Level	Characterization
R1	Localized change requiring limited commitment repair and state refresh.
R2	Coordinated repair involving multiple affected objects, commitments, or state dependencies.
R3	Compound replanning under successive or broad changes, requiring substantial commitment-state maintenance and downstream propagation.

covery paths, `if_event_triggered` activates disruption-specific requirements, and `commitment_coverage` aligns persistent commitments with the final active plan. Unknown rules, missing required plans, and evaluation errors fail closed. Diagnostic messages, field aliases, and bookkeeping logic are omitted for clarity.

A.5 Difficulty Stratification

To characterize the temporal-planning and commitment-aware-replanning complexity of DynPlan-Bench, we stratify tasks along two complementary dimensions: **Temporal Difficulty (T1–T3)** and **Replanning Difficulty (R1–R3)**. Both dimensions are defined from intrinsic task properties and calibrated to the operational structure of each domain. Lower levels represent local, weakly coupled decisions. Higher levels involve broader cross-step dependencies, dynamic state changes, and existing commitments that must be coordinated.

Replanning Difficulty. Replanning Difficulty measures the complexity of restoring a feasible plan after a dynamic change, given the agent's execution history and persistent commitments. We characterize the replanning burden along three factors: Event burden: The number of dynamic changes that must be handled and whether they occur sequentially. Commitment burden: The persistent commitments that must be preserved, modified, cancelled, or re-established after a change. Perception burden: The scope of relevant state that the agent must identify or reacquire to determine the full effects of a change. Based on these three factors, as shown in Table.6, we compute a composite replanning structural score for each task and assign the task to one of three difficulty levels according to predefined score thresholds: R1 for scores ≤ 3, R2 for scores from 4 to 6, and R3 for scores ≥ 7. Among the 153 tasks, 46, 73, and 34 are classified as R1, R2, and R3, respectively. The calibration rules reflect the operational structure of each domain. In Logistics, difficult repairs often require coordinated changes across vehicles, routes, warehouses, and delivery states. Retail emphasizes dependencies among inventory, supply, staffing, and operational decisions. Travel commonly requires cascading repairs across multiple itinerary segments and bookings.

Temporal Difficulty. Temporal Difficulty measures the temporal structure that the agent must maintain when planning future actions and the propagation of temporal decisions across subsequent steps. We characterize this difficulty along four factors: Temporal constraint density: The number and interaction of time windows, durations, ordering relations, deadlines, calendar conditions, and other temporal constraints. Dependency depth: The extent to which a temporal decision affects subsequent planning steps and the depth over which its effects propagate. Slack tightness: The amount of temporal flexibility available for adjusting mutually dependent activities. Time-indexed state evolution: The extent to which traffic, inventory, congestion, service capacity, product quality, and related states change during plan execution. We similarly combine the four factors above to compute a temporal structural score for each task and assign the task to one of three difficulty levels according to predefined score thresholds: T1 for scores ≤ 10, T2 for scores from 11 to 13, and T3 for scores ≥ 14. These factors manifest differently across domains. In Logistics, temporal difficulty arises primarily from interactions among delivery windows, route times, and resource availability. Retail more often involves the joint evolution of demand, freshness, supply, and staffing conditions. Travel combines cross-location transit, opening hours, reservation slots, and itinerary dependencies spanning multiple days.

Table 7: Temporal difficulty levels.

Level	Characterization
T1	Primarily local temporal conditions with loose dependencies and limited time-varying state changes.
T2	Multiple coupled temporal constraints or evolving states that must be coordinated across several plan steps.
T3	Dense temporal dependencies, tight feasible margins, or compound time-indexed evolution requiring global temporal consistency.

B Experimental Details and Additional Results

B.1 Experimental Setup

Models. We evaluate 20 language models spanning closed-source, open-source, and more efficient model families. The closed-source models include Claude Opus 5, GPT-5.6 Sol, GPT-5.6 Terra, GPT-5.6 Luna, GPT-5.5 and Qwen3.7-Max. The open-source models include DeepSeek-V4-Pro, DeepSeek-V4-Flash, GLM-5.2, MiniMax M3, Kimi K2.6, LongCat-2.0, MiMo-V2.5-Pro, and Hy3. The more efficient models include Gemma-4-26B-A4B-IT, Gemma-4-31B-IT, Qwen3.6-35B-A3B, Qwen3.6-27B, Qwen3.5-27B, and GLM-4.7-Flash. For models that support configurable reasoning or thinking modes, we use the reasoning-enabled configuration recommended by the corresponding provider.

Agent Scaffold. To ensure a consistent comparison across models, all agents are implemented under the same function-calling scaffold based on the OpenAI tool schema. For a given task, all models receive the same task instruction, tool definitions, and environment feedback. At each interaction step, the agent may invoke an available domain tool, submit or update a structured plan, or terminate the task. Tool calls are executed directly in the corresponding executable environment, where they may update the database, persistent commitments, world time, and subsequent observations.

Information Access. During evaluation, the agent has access only to the task instruction, the available tools, and information exposed through its interaction with the environment. The agent is not provided with the reference trajectory, task-specific Judge Rules, or future events that have not yet been triggered or revealed. For dynamic changes that are not directly exposed, the agent must query the relevant environment state through subsequent tool interactions.

Interaction Protocol. All trials start from the same task-specific initial world state and world time. The environment transition is deterministic given the initial state and agent actions. As execution proceeds, Time-Indexed Dynamics are evaluated at the corresponding world time, while Event-Driven Dynamics are applied when their trigger conditions are satisfied. Each rollout is limited to 150 interaction turns and terminates when the agent outputs `###STOP###` or reaches the maximum turn limit.If a scheduled time-based event remains pending when the agent outputs `###STOP###`, termination is deferred until the event is triggered and the resulting observation is returned to the agent, after which interaction resumes.

Sampling Settings. For the main evaluation, each task is independently run four times with the temperature set to 0.0. All models are evaluated using the same interaction budget and evaluation protocol. Although we use temperature=0 for reproducibility, agent trajectories can still diverge in multi-step interactions. This is expected in environments because early action choices alter subsequent environment states and available options. We therefore analyze four independent runs for each task and observe that most variance originates from different intermediate planning decisions.
Metrics. Each rollout produces a binary task reward $r \in \{0, 1\}$ according to the deterministic evaluation described in Sec. 3.4. For the main evaluation, we report three complementary metrics over four independent trials: Avg@4, Pass@4, and Pass^4. Avg@4 measures the mean success rate across the four runs. Pass@4 denotes the probability that at least one of the four trials successfully completes the task, whereas Pass^4 measures the probability that all four trials are successful, providing a stricter measure of execution reliability.

More generally, suppose a task is evaluated over n valid trials, of which c are successful. We compute Pass@k and Pass^k as

$$\text{Pass@}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \qquad \text{Passˆk} = \frac{\binom{c}{k}}{\binom{n}{k}}. \tag{2}$$

For the main four-run setting, where $n = k = 4$, Pass@4 indicates whether at least one trial succeeds, while Passˆ4 requires all four trials to succeed. Avg@4 is defined as

$$\text{Avg@4} = \frac{1}{4}\sum_{j=1}^{4} r_j, \tag{3}$$

which measures the average task success across repeated executions.

B.2 Agent Prompts and Interaction Instructions

All models are evaluated using a shared prompt structure, with domain-specific operational instructions instantiated for Travel, Retail, and Logistics. The complete system prompts used in our experiments are provided below.

Travel Planning Agent Prompt

```
[System]

# Travel Planning Agent

## Environment

You are an autonomous travel planning agent. Plan and execute a
    complete,
feasible trip that satisfies the user's requirements. Make routine
    decisions
yourself when the available information is sufficient.

## Tool Usage Guidelines

- Use tools to obtain identifiers, schedules, prices, availability,
    policies,
  travel times, and current conditions. Do not invent operational
    facts.
- Inspect relevant options before making commitments and use exact
    values
  returned by tools.
- Gather enough information to compare realistic options, then
    proceed without
  unrelated exhaustive searches.
- Treat tool errors and new environment or user messages as
    authoritative.
  Verify affected options before changing commitments.

## Planning and Execution

- Call `submit_plan` with a complete structured itinerary before
    making any
  booking.
- Keep dates, times, locations, party details, and booked services
    consistent.
- Allow enough time for local travel between consecutive activities.
  An explicit transport segment applies only when it correctly
    connects the
  surrounding locations and times.
- Use `get_plan` when needed.
- Update the plan before executing commitments that depend on the
    revision.
```

```
- Booking and cancellation tools use the environment's current
    time. Future
  itinerary timestamps do not change booking time. Use `wait_until`
    only when
  intentionally advancing time is necessary.
- Check cancellation terms before replacing an order. Reserve
    restaurants only
  when the selected table or service requires a reservation.

## Completion

Respect all explicit constraints and keep the final active
    itinerary consistent
with the final active orders. Finish only after required bookings
    are complete
and invalid commitments have been resolved.
```

Retail Operations Agent Prompt

```
[System]

# Retail Operations Agent

## Environment

You are an autonomous retail operations agent. Plan and execute the
    requested
procurement, stocking, pricing, promotion, staffing, inspection,
    and disposal
work. Make routine decisions yourself when the available
    information is
sufficient.

## Tool Usage Guidelines

- Use tools to obtain product, inventory batch, supplier, storage
    zone, staff,
  demand, freshness, and order information. Do not invent
    operational facts.
- Inspect the current state before creating or modifying an
    operation and use
  exact identifiers returned by tools.
- Gather enough information to establish feasibility, then proceed
    without
  unrelated exhaustive searches.
- Treat tool errors and new environment or user messages as
    authoritative.
  Recheck resources affected by a disruption.

## Planning and Execution

- Call `submit_plan` with a complete structured operations plan
    before creating
  orders, assignments, adjustments, campaigns, or disposals.
- Keep plan segments consistent with their products, batches,
    suppliers, zones,
  staff, quantities, dates, and times.
- Use `get_plan` when needed.
- Update the plan before executing operations that depend on
    changed inventory,
```

```
  staffing, pricing, promotion, storage, or supplier conditions.
- Verify the correct inventory batch and respect freshness, expiry,
    temperature,
  and storage compatibility. Do not stock unsafe goods or use
    incompatible
  zones.
- Check supplier capacity and lead time before procurement. Check
    staff
  availability, skills, coverage, and working limits before
    assignments.
- Limit pricing, promotion, and disposal actions to the requested
    scope.
- Time-sensitive tools use the environment's current time. Use
    `wait_until`
  only when intentionally advancing time is necessary. Check
    current state and
  applicable terms before modifying or cancelling a commitment.

## Completion

Finish only after every required operation is represented in the
    active plan,
all necessary actions have been executed, and the final operational
    state is
consistent.
```

Logistics Dispatch Agent Prompt

```
[System]

# Logistics Dispatch Agent

## Environment

You are an autonomous logistics dispatch agent. Plan and execute
    all requested
pickups, deliveries, transfers, and vehicle assignments. Make
    routine decisions
yourself when the available information is sufficient.

## Tool Usage Guidelines

- Use tools to obtain warehouse, supplier, customer, route,
    vehicle, capacity,
  traffic, weather, and order information. Do not invent
    operational facts.
- Inspect relevant resources before creating or modifying
    commitments and use
  exact identifiers returned by tools.
- Gather enough information to establish feasibility, then proceed
    without
  unrelated exhaustive searches.
- Treat tool errors and new environment or user messages as
    authoritative.
  Recheck resources affected by a disruption.

## Planning and Execution

- Call `submit_plan` with a complete structured dispatch plan
    before creating
```

```
  orders, assignments, or transfers.
- Keep segment times, locations, routes, vehicles, and related
    orders
  consistent. Include travel, loading, unloading, and transfer time.
- Use `get_plan` when needed.
- Update the plan before executing operations that depend on
    changed routing,
  timing, resources, or destinations.
- Check vehicle availability, capacity, route compatibility, driver
    limits,
  service windows, and cargo handling requirements. Restricted
    goods must use
  compatible vehicles and storage zones throughout the operation.
- Time-sensitive tools use the environment's current time. Use
    `wait_until`
  only when intentionally advancing time is necessary. Check order
    state and
  cancellation terms before modifying or cancelling a commitment.

## Completion

Finish only after every required operation is represented in the
    active plan,
all necessary commitments have been executed, and the final state
    is feasible
and consistent.
```

B.3 Additional Results and Ablations

Sensitivity to Agent Interaction Scaffolds. We compare three interaction scaffolds under the same backbone and task setting: Function Calling (FC), ReAct (Yao et al., 2023), and Act-only. FC uses the model's native structured function-calling interface, whereas ReAct and Act-only use text-formatted actions; ReAct additionally interleaves explicit reasoning traces with actions, while Act-only directly predicts actions without an explicit reasoning trace. Under the same backbone, FC achieves the best overall performance for both models, whereas ReAct performs the worst: GPT-5.6 Sol drops from 44.6% with FC to 32.8% with ReAct, while DeepSeek-V4-Pro drops from 35.8% to 20.1%. Further inspection of the trajectories shows that ReAct tends to produce longer execution sequences and is more prone to losing global plan context during local adaptation, causing still-valid commitments and downstream plan segments to be inconsistently preserved. These results suggest that, in long-horizon dynamic tasks, the choice of interaction scaffold affects not only how reasoning and actions are organized, but also how reliably agents maintain plan state and execution history across successive rounds of execution and repair.

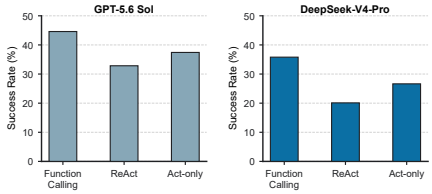

Figure 9: Effect of interaction scaffolds on agent performance.

Effect of Reasoning. We further isolate the effect of model-side reasoning by comparing thinking and no-thinking configurations under the same function-calling scaffold, while keeping the task setting, tools, and interaction budget unchanged. Table.8 reports results for GPT-5.6 Sol and DeepSeek-V4-Pro under the two configurations. Enabling thinking improves the overall performance by 5.9 pp and 5.2 pp for the two models, respectively, indicating that additional reasoning generally benefits performance on dynamic planning tasks. However, the gains

Table 8: Effect of reasoning on DynPlanBench.

Model	Mode	Overall	Logistics	Retail	Travel
Sol	Thinking	44.6	45.9	45.8	41.0
	No-Think	38.7	45.9	34.4	33.3
	Δ	+5.9	0.0	+11.4	+7.7
DS-V4	Thinking	35.8	36.1	35.4	35.9
	No-Think	30.6	39.3	20.3	30.8
	Δ	+5.2	-3.2	+15.1	+5.1

Table 10: Results on DynPlanBench stratified by replanning difficulty. All values are percentages. The best and second-best results in each column are shown in **bold** and underline, respectively.

Models	R1			R2			R3		
	Avg@4	Pass@4	Pass^4	Avg@4	Pass@4	Pass^4	Avg@4	Pass@4	Pass^4
Closed-Source LLMs									
Claude Opus 5	**65.8**	**80.4**	**32.6**	**41.4**	**63.0**	13.7	**41.2**	61.8	14.7
GPT-5.6 Sol	54.3	78.3	26.1	40.8	60.3	**19.2**	39.7	**64.7**	**17.6**
GPT-5.5	52.7	69.6	28.3	40.8	60.3	**19.2**	31.6	44.1	8.8
GPT-5.6 Terra	50.0	67.4	23.9	28.1	47.9	4.1	27.2	44.1	5.9
Qwen3.7-Max	47.3	63.0	26.1	27.4	42.5	8.2	26.5	52.9	5.9
GPT-5.6 Luna	46.7	65.2	15.2	26.0	42.5	8.2	19.9	38.2	8.8
Open-Source LLMs									
DeepSeek-V4-Pro	50.5	78.3	21.7	31.8	54.8	11.0	24.3	50.0	5.9
GLM-5.2	51.1	69.6	26.1	26.7	41.1	8.2	22.8	35.3	8.8
DeepSeek-V4-Flash	39.7	67.4	8.7	24.7	38.4	9.6	14.7	35.3	5.9
MiniMax M3	38.6	63.0	17.4	23.3	42.5	2.7	18.4	38.2	2.9
LongCat-2.0	44.0	63.0	17.4	17.8	34.2	1.4	13.2	26.5	0.0
Kimi K2.6	38.6	58.7	13.0	20.5	35.6	4.1	11.8	32.4	0.0
MiMo-V2.5-Pro	33.2	56.5	4.3	12.7	31.5	1.4	12.5	26.5	0.0
Hy3	25.5	52.2	4.3	12.0	21.9	2.7	7.4	11.8	0.0
More Efficient LLMs									
Qwen3.6-35B-A3B	29.3	50.0	8.7	14.7	32.9	1.4	5.9	14.7	0.0
Qwen3.6-27B	23.9	43.5	0.0	11.3	17.8	2.7	11.0	26.5	0.0
Gemma-4-31B-IT	25.0	39.1	10.9	8.2	17.8	1.4	5.1	8.8	0.0
Qwen3.5-27B	20.7	41.3	4.3	9.9	21.9	2.7	3.7	8.8	0.0
Gemma-4-26B-A4B-IT	8.2	21.7	0.0	2.1	6.8	0.0	0.0	0.0	0.0
GLM-4.7-Flash	5.4	17.4	0.0	1.0	2.7	0.0	0.0	0.0	0.0

vary substantially across domains: in Retail, GPT-5.6 Sol and DeepSeek-V4-Pro improve by 11.3 pp and 15.1 pp, respectively, whereas in Logistics, Sol shows no improvement and DeepSeek-V4-Pro decreases by 3.3 pp. Retail tasks more often require agents to jointly reason over evolving inventory conditions, freshness, supplier constraints, staffing, and operational deadlines, whereas many temporal and resource dependencies in Logistics are exposed more explicitly through operational variables. This domain-dependent pattern suggests that additional reasoning is particularly helpful when agents must integrate implicit or coupled dependencies across multiple steps, but does not uniformly improve performance when the relevant constraints are already exposed through explicit environment states.

Evaluator Reliability We further validate the reliability of our deterministic Judge through independent semantic review of 612 trajectories. For each of four models—Claude Opus 5, GPT-5.6 Sol, DeepSeek-V4-Pro, and GLM-5.2—we randomly select one complete evaluation run covering all 153 tasks. Two reviewers independently assess each trajectory without access to the Judge's Overall labels. They examine task requirements, interaction histories, tool responses, submitted plans, and final environment states across four dimensions: Overall, Execution, Plan, and Consistency. Their agreed assessment serves as the reference judgment; disagreements are adjudicated by a third reviewer. We further investigate discrepancies between the reference judgment and the Judge by inspecting the relevant rules, action–response chains, state transitions, event traces, and final commitments. Across these 612 trajectories, the Judge achieves an Overall Accuracy of 96.57%.

Table 9: Reliability of the deterministic evaluator.

Metric	Score
Overall Accuracy	96.57%
Overall Cohen's κ	0.928
Rubric Accuracy	93.22%

Difficulty-Stratified Results. As shown in Tables.10 and 11, we report the detailed results corresponding to the difficulty analysis in Figure.4.

Table 11: Results on DynPlanBench stratified by temporal difficulty. All values are percentages. The best and second-best results in each column are shown in **bold** and underline, respectively.

Models	T1			T2			T3		
	Avg@4	Pass@4	Passˆ4	Avg@4	Pass@4	Passˆ4	Avg@4	Pass@4	Passˆ4
Closed-Source LLMs									
Claude Opus 5	**60.4**	**77.1**	22.9	**47.2**	**70.4**	**21.1**	35.3	50.0	11.8
GPT-5.6 Sol	49.5	75.0	20.8	44.7	66.2	**21.1**	**37.5**	**55.9**	**20.6**
GPT-5.5	53.1	72.9	**27.1**	41.2	60.6	18.3	29.4	38.2	11.8
GPT-5.6 Terra	46.4	66.7	16.7	31.3	47.9	8.5	24.3	44.1	5.9
Qwen3.7-Max	41.7	58.3	20.8	30.6	52.1	7.0	26.5	38.2	14.7
GPT-5.6 Luna	41.7	62.5	10.4	28.2	46.5	11.3	21.3	32.4	8.8
Open-Source LLMs									
DeepSeek-V4-Pro	46.4	72.9	20.8	34.9	57.7	12.7	22.8	50.0	2.9
GLM-5.2	43.2	58.3	18.8	32.7	49.3	12.7	19.9	32.4	8.8
DeepSeek-V4-Flash	35.4	58.3	8.3	26.1	45.1	11.3	16.9	32.4	2.9
MiniMax M3	33.9	56.3	12.5	26.8	49.3	4.2	16.9	32.4	5.9
LongCat-2.0	33.3	52.1	12.5	22.9	40.8	2.8	16.2	26.5	2.9
Kimi K2.6	34.4	56.3	4.2	20.8	33.8	8.5	16.2	38.2	2.9
MiMo-V2.5-Pro	23.4	50.0	0.0	16.9	36.6	2.8	16.2	23.5	2.9
Hy3	20.8	35.4	6.3	12.7	28.2	0.0	11.8	20.6	2.9
More Efficient LLMs									
Qwen3.6-35B-A3B	22.9	45.8	2.1	16.2	31.0	2.8	11.0	23.5	5.9
Qwen3.6-27B	19.3	35.4	0.0	14.1	25.4	2.8	11.0	20.6	0.0
Gemma-4-31B-IT	20.8	35.4	8.3	10.6	18.3	2.8	5.1	11.8	0.0
Qwen3.5-27B	14.6	33.3	4.2	10.9	21.1	1.4	9.6	20.6	2.9
Gemma-4-26B-A4B-IT	7.3	16.7	0.0	1.8	7.0	0.0	1.5	5.9	0.0
GLM-4.7-Flash	2.6	8.3	0.0	2.1	5.6	0.0	1.5	5.9	0.0

Error Pattern Analysis

To further analyze the sources of failure in dynamic planning tasks, we categorize unrecovered errors in failed trajectories into four stages: **Observe**, **Reason**, **Replan**, and **Execute**. These stages respectively correspond to acquiring dynamic information, assessing the current state and plan feasibility, repairing an existing plan after changes, and translating decisions into concrete tool operations. We further define fine-grained error types within each stage to distinguish failures in information acquisition, temporal and constraint reasoning, commitment-aware repair, and execution. **Observe.** Observe errors occur when the agent fails to obtain, refresh, or correctly identify the environmental information required for the current decision. This includes missing necessary information or continuing to rely on stale state after temporal progression or event-driven changes. **Reason.** Reason errors occur when the agent has obtained the relevant information but fails to correctly assess temporal feasibility, constraint relations, or cross-step dependencies, resulting in decisions that may be locally plausible but globally infeasible. **Replan.** Replan errors occur after a dynamic change has been recognized but the agent fails to correctly repair the plan under its execution history and persistent commitments. These errors include incorrect commitment handling, incomplete propagation of local repairs, and omission of still-required goals or plan segments during revision. **Execute.** Execute errors occur when the agent's high-level decision is largely appropriate but is incorrectly realized through tool operations, such as using the wrong entity or arguments or failing to carry out a required business action.

We inspect the complete failed trajectories of GPT-5.6 Sol and annotate unrecovered errors that materially contribute to task failure. Two annotators independently identify and categorize errors, with disagreements adjudicated by a third annotator. They examine the full interaction history, tool responses, submitted plans, event traces, and final world state to identify errors in observation, reasoning, replanning, or execution. A trajectory may contain multiple distinct unrecovered errors, each assigned to one fine-grained category. The distribution in Figure 7 is normalized over all annotated

Table 12: Fine-grained error taxonomy used for trajectory analysis.

Stage	Error Type	Characterization
Observe	Information Acquisition	Missing information required for the current decision.
	State Refresh	Reliance on stale state after temporal or event-driven changes.
Reason	Temporal Feasibility	Incorrect assessment of timing or execution-time feasibility.
	Constraint / Dependency Reasoning	Failure to account for coupled constraints or propagate dependencies.
	Other Reasoning	Other incorrect decisions or constraint interpretations.
Replan	Commitment Handling	Incorrect maintenance or revision of persistent commitments.
	Downstream Propagation	Failure to propagate a local repair to affected downstream steps.
	Goal / Plan Omission	Missing required goals or valid plan segments after replanning.
Execute	Argument / Object Grounding	Correct operation type with incorrect objects or arguments.
	Other Execution	Other failures in carrying out the intended operation.

error instances and therefore represents error composition rather than the fraction of trajectories exhibiting each error type.

C CASE STUDIES

We present two representative failed trajectories corresponding to the two recurring failure patterns discussed in Sec.5.2. To save space we also present only the failure-relevant portions of the original task instructions, omitting requirements that do not materially affect the analyzed failure. We preserve the chronological order of interactions, omit unrelated tool calls, and abridge tool arguments and responses while retaining the values necessary to diagnose each failure. Each retained tool call is shown individually.

C.1 CASE I: LOCAL FEASIBILITY, GLOBAL TEMPORAL BREAKDOWN

Instruction

It is currently 20:00 on June 16, 2026. My partner and I are traveling from Kunming to Lijiang and Dali from June 17–22. In Lijiang, the one thing we absolutely do not want to miss is Jade Dragon Snow Mountain: we want to take the *large* cableway to Glacier Park, and because of the altitude, please schedule the mountain only after a day of acclimatization. We will be staying at a Naxi-style courtyard inn in Lijiang Old Town.

Initial interactions omitted

The agent first searches candidate hotels, intercity transport, attractions, weather, hotel inventory/prices, attraction metadata, and several candidate dates.

Assistant – Tool Request [Turn 21]

```
Tool: get_attraction_inventory
Arguments:
{
  "attraction_id": "ATTR_LJ_002",
  "visit_day": "2026-06-18"
}
```

Tool Call/Response [Turn 21]


```
{
  "attraction_id": "ATTR_LJ_002",
  "visit_date": "2026-06-18",
  "query_time": "2026-06-16 20:00:00",
  "inventory": {
    "Mountain Admission": {
      "remaining": 1000,
      "total": 20000,
      "status": "Very Tight",
      "percentage": "5.0%"
    },
    "Large Cableway (Glacier Park)": {
      "remaining": 500,
      "total": 10000,
      "status": "Very Tight",
      "percentage": "5.0%"
    },
    "Small Cableway (Spruce Meadow)": {
      "remaining": 3258,
      "total": 5000,
      "status": "Sufficient",
      "percentage": "65.2%"
    },
    "Scenic Shuttle": {
      "remaining": 1000,
      "total": 20000,
      "status": "Very Tight",
      "percentage": "5.0%"
    }
  }
}
```

Assistant – Tool Request [Turn 28]

```
Tool: get_attraction_queue
Arguments:
{
  "attraction_id": "ATTR_LJ_002",
  "t": "2026-06-18 08:30:00"
}
```

Tool Call/Response [Turn 28]


```
{
  "attraction_id": "ATTR_LJ_002",
  "time": "2026-06-18 08:30:00",
  "is_open": true,
  "crowd_index": 0.29,
  "estimated_queue_minutes": 47.0,
  "description": "Estimated waiting time in minutes."
}
```


Assistant – Tool Request [Turn 29]

```
Tool: get_attraction_queue
Arguments:
{
  "attraction_id": "ATTR_LJ_002",
  "t": "2026-06-18 13:00:00"
}
```

Tool Call/Response [Turn 29]


```
{
  "attraction_id": "ATTR_LJ_002",
  "time": "2026-06-18 13:00:00",
  "is_open": true,
  "crowd_index": 0.19,
  "estimated_queue_minutes": 42.5,
  "description": "Estimated waiting time in minutes."
}
```


Intermediate interactions omitted

Additional attraction, price, crowd, hotel, and transport checks are omitted. The next retained call is the local travel-time query used to connect the selected Lijiang hotel to Jade Dragon Snow Mountain.

Assistant – Tool Request [Turn 65]

```
Tool: get_transport_duration
Arguments:
{
  "destination_id": "ATTR_LJ_002",
  "mode": "taxi",
  "origin_id": "HTL_LJ_004",
  "t": "2026-06-18 11:00:00"
}
```

Tool Call/Response [Turn 65]


```
{
  "origin_id": "HTL_LJ_004",
  "destination_id": "ATTR_LJ_002",
  "mode": "taxi",
  "duration_minutes": 44.8,
  "distance_km": 21.45,
  "is_from_config": false,
  "base_duration_min": 34.8,
  "congestion_index": 0.15,
  "traffic_status": "Normal",
  "factors": {
    "is_weekend": false,
    "weather_impact": 1.04,
    "event_impact": 1.0,
```

```
    "override_impact": 1.0
  },
  "query_mode": "point",
  "query_time": "2026-06-18 11:00:00"
}
```


Other local-transfer queries omitted

The agent queries several other local transfers, but it does not *query the reverse transfer from* `ATTR_LJ_002` *to* `HTL_LJ_004` *at the planned 17:30 departure.*

Assistant – Tool Request [Turn 83]


```
Tool: submit_plan
Arguments (abridged; failure-relevant segments only):
{
  "itinerary": {
    "segments": [
      {
        "segment_id": "seg7",
        "date": "2026-06-18",
        "segment_type": "transport",
        "origin_id": "HTL_LJ_004",
        "destination_id": "ATTR_LJ_002",
        "start_time": "10:50",
        "end_time": "11:35",
        "transport_mode": "taxi"
      },
      {
        "segment_id": "seg8",
        "date": "2026-06-18",
        "segment_type": "attraction",
        "location_id": "ATTR_LJ_002",
        "start_time": "11:40",
        "end_time": "17:30"
      },
      {
        "segment_id": "seg9",
        "date": "2026-06-18",
        "segment_type": "hotel",
        "location_id": "HTL_LJ_004",
        "start_time": "18:30",
        "end_time": "23:00"
      }
    ]
  }
}
```


Tool Call/Response [Turn 83]


```
{
  "status": "success",
  "message": "Plan submitted successfully.",
  "plan_id": "plan_001",
  "version": 1,
  "segments": 27
}
```


Routine bookings omitted

Hotel, rail, and unrelated attraction bookings are omitted. The following retained booking creates the core Glacier Park cableway commitment associated with the June 18 mountain visit.

Assistant – Tool Request [Turn 88]


```
Tool: create_attraction_order
Arguments:
{
  "attraction_id": "ATTR_LJ_002",
  "date": "2026-06-18",
  "quantity": 2,
  "ticket_type_id": "T_BIG_CABLE",
  "time_slot_id": "S_AFTERNOON",
```

```
  "user_id": ""
}
```


Tool Call/Response [Turn 88]


```
{
  "status": "success",
  "message": "Order created successfully",
  "order_id": "ORD-ATT-0005",
  "commit_time": "2026-06-16 20:00:00",
  "ticket_type_id": "T_BIG_CABLE",
  "ticket_type_name": "Large Cableway (Glacier Park)"
}
```


Highlighted Plan-Feasibility Checks

`no_overlap`: **PASS** — "No time overlaps detected."

`itinerary_chain_reachable`: **FAIL** — "On 2026-06-18, cannot reach HTL_LJ_004 by 18:30: ATTR_LJ_002-¿HTL_LJ_004 needs ∼ 77 min but only 60 min available."

Failure Analysis. The agent explicitly checks the outbound hotel-to-mountain travel time and selects a lower-crowd afternoon visit, but it incorrectly assumes that travel time remains unchanged across departure times, and therefore never verifies the reverse transfer needed after the mountain segment. The submitted plan ends `ATTR_LJ_002` at 17:30 and starts the hotel segment at 18:30, leaving a positive 60-minute gap. The deterministic validator re-evaluates the missing reverse connection at its planned 17:30 execution time, yielding approximately 77 minutes. The plan is therefore locally non-overlapping yet globally unreachable, illustrating the *local feasibility, global temporal breakdown* pattern.

C.2 Case II: Commitment-State Drift after Replanning

Instruction

It is currently 20:00 on June 14, 2026. My girlfriend and I are taking a slow, romantic two-day trip from Shanghai to Hangzhou on June 15–16. The most important part of the trip is a *sunset* boat cruise on West Lake—we specifically want to be on the water for sunset, so please do not replace it with an ordinary daytime cruise. We would not want to spend the evening in long queues..

Pre-booking exploration omitted

The agent queries the relevant attraction, queue, inventory, transport, and booking information, submits an initial itinerary, and books the June 15 West Lake sunset cruise.

Assistant – Tool Request [Turn 69]


```
Tool: create_attraction_order
Arguments:
{
  "attraction_id": "ATTR_HZ_001",
  "ticket_type_id": "T_BOAT_SUNSET",
  "date": "2026-06-15",
  "quantity": 2,
  "time_slot_id": "S_AFTERNOON"
}
```


Tool Call/Response [Turn 69]


```
{
  "status": "success",
  "order_id": "ORD-ATT-0001",
  "ticket_type_id": "T_BOAT_SUNSET",
  "ticket_type_name": "Sunset Boat Ticket"
}
```

Runtime Event

At 2026-06-15 06:00, a temporary West Lake night event raises the crowd index for `ATTR_HZ_001` to 0.95 and the expected queue to 90 minutes from 16:00 to 19:30, requiring the June 15 sunset-cruise schedule to be reconsidered.

Assistant – Tool Request [Turn 93]

```
Tool: get_attraction_queue
Arguments:
{
  "attraction_id": "ATTR_HZ_001",
  "t": "2026-06-15 16:30:00"
}
```

Tool Call/Response [Turn 93]

```
{
  "time": "2026-06-15 16:30:00",
  "crowd_index": 0.95,
  "estimated_queue_minutes": 90.0
}
```

Additional queue probes omitted

The agent performs additional June 15 boundary checks around the 16:00 crowd surge.

Assistant – Tool Request [Turn 107]

```
Tool: get_attraction_queue
Arguments:
{
  "attraction_id": "ATTR_HZ_001",
  "t": "2026-06-16 14:30:00"
}
```

Tool Call/Response [Turn 107]

```
{
  "attraction_id": "ATTR_HZ_001",
  "time": "2026-06-16 14:30:00",
  "is_open": true,
  "crowd_index": 0.09,
  "estimated_queue_minutes": 5.6
}
```

Assistant

The June 15 sunset cruise is no longer feasible under the event conditions. I will move the sunset cruise to June 16 and keep the rest of the feasible itinerary unchanged.

Assistant – Tool Request [Turn 114]

```
Tool: update_plan
Arguments (abridged; commitment-relevant change only):
plan_id: plan_001

change_summary:
- Move the West Lake sunset cruise from June 15 to June 16 because the June 15 evening
    queue surge requires the original schedule to be reconsidered.

Selected itinerary change:
- sunset_boat:
    date: 2026-06-15 -> 2026-06-16
    service: T_BOAT_SUNSET (unchanged)
```

Tool Call/Response [Turn 114]


```
{
  "status": "success",
  "message": "Plan updated successfully.",
  "plan_id": "plan_001"
}
```


Commitment state is not repaired

After moving the cruise to June 16 in the active plan, the agent does not cancel the paid June 15 sunset-boat order `ORD-ATT-0001` and does not create a replacement June 16 `T_BOAT_SUNSET` order.

Highlighted Commitment Checks

`not_exists_active_order` for the June 15 cruise: **FAIL** — the old June 15 attraction order remains paid and active.

`exists_cancelled_order` for the June 15 cruise: **FAIL** — no cancelled June 15 West Lake attraction order exists.

`exists_order` for the repaired June 16 sunset cruise: **FAIL** — no matching `T_BOAT_SUNSET` order exists for June 16.

Failure Analysis. The agent moves the sunset cruise from June 15 to June 16 in the revised *plan*, but the persistent commitment state still encodes the superseded itinerary: the June 15 paid order remains active, and no June 16 replacement order exists. The resulting failure is a *plan–commitment inconsistency*: the active commitments neither release the abandoned booking nor realize the revised plan. This divergence between two persistent representations of the same intended activity constitutes *commitment-state drift*.