TravelPlanner: A Benchmark for Real-World Planning with Language Agents

Jian Xie [*1] **Kai Zhang** [*2] **Jiangjie Chen** [1] **Tinghui Zhu** [1] **Renze Lou** [3]
Yuandong Tian [4] **Yanghua Xiao** [1] **Yu Su** [2]

Abstract

Planning has been part of the core pursuit for artificial intelligence since its conception, but earlier AI agents mostly focused on constrained settings because many of the cognitive substrates necessary for human-level planning have been lacking. Recently, language agents powered by large language models (LLMs) have shown interesting capabilities such as tool use and reasoning. *Are these language agents capable of planning in more complex settings that are out of the reach of prior AI agents?* To advance this investigation, we propose TravelPlanner, a new planning benchmark that focuses on travel planning, a common real-world planning scenario. It provides a rich sandbox environment, various tools for accessing nearly four million data records, and 1225 meticulously curated planning intents and reference plans. Comprehensive evaluations show that the current language agents are not yet capable of handling such complex planning tasks—even GPT-4 only achieves a success rate of 0.6%. Language agents struggle to stay on task, use the right tools to collect information, or keep track of multiple constraints. However, we note that the mere possibility for language agents to tackle such a complex problem is in itself non-trivial progress. TravelPlanner provides a challenging yet meaningful testbed for future language agents. All the resources are available on the project website.

1. Introduction

Planning is a hallmark of human intelligence. It is an evolutionary feat built upon numerous other capacities: using various tools to iteratively collect information and make decisions, recording intermediate plans (in working memory or on a physical device) for deliberation, and exploring alternative plans by running simulations, which in turn depends on a world model (Mattar & Lengyel, 2022; Ho et al., 2022). For decades, researchers have been attempting to develop AI agents to mimic humans' planning capability (Russell & Norvig, 2010; Georgievski & Aiello, 2015; Karpas & Magazzeni, 2020; Lin et al., 2023; Zhang et al., 2024), but often in constrained settings (Campbell et al., 2002; Silver et al., 2016; 2017) because many of the cognitive substrates necessary for human-level planning have been lacking. AI agents that can work robustly in the largely unconstrained settings in which humans operate remain a distant goal.

The advent of large language models (LLMs; OpenAI (2022; 2023); Touvron et al. (2023a;b); Jiang et al. (2023)) brings new light to this classic problem. A new generation of language agents (Su, 2023; Sumers et al., 2023; Xie et al., 2023) powered by LLMs has emerged, characterized by their capability of using language as a vehicle for thought and communication. These agents have shown interesting capabilities, such as tool use (Schick et al., 2023; Patil et al., 2023; Qin et al., 2024) and various forms of reasoning (Wei et al., 2022; Yao et al., 2022; Lewkowycz et al., 2022), potentially fulfilling the role of some of the cognitive substrates that were lacking in earlier AI agents. Researchers have therefore investigated their potential in an array of planning tasks ranging from classic planning settings like Blocksworld (Valmeekam et al., 2023) to embodied agents (Huang et al., 2022; Ichter et al., 2022; Song et al., 2023; Wang et al., 2023) and web agents (Deng et al., 2023; Zhou et al., 2024). However, the planning settings in existing work still largely follow the conventional setting that focuses on single-objective optimization with fixed ground truths. An agent is tasked with predicting from a pre-defined set of actions, just now made by an LLM-powered agent.

Are language agents capable of planning in more complex yet realistic settings, closer to those in which humans operate? To advance this investigation, we propose TravelPlanner, a new planning benchmark that focuses on a common real-world planning scenario—travel planning. This is a challenging, time-consuming task even for humans (but most people can do it successfully, with the right tools and

[*]Equal contribution. Work done during Jian's internship at OSU NLP Group. [1]School of Computer Science, Fudan University [2]The Ohio State University [3]The Pennsylvania State University [4]Meta AI. Correspondence to: Jian Xie <jianxie22@m.fudan.edu.cn>, Kai Zhang <zhang.13253@osu.edu>, Yu Su <su.809@osu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

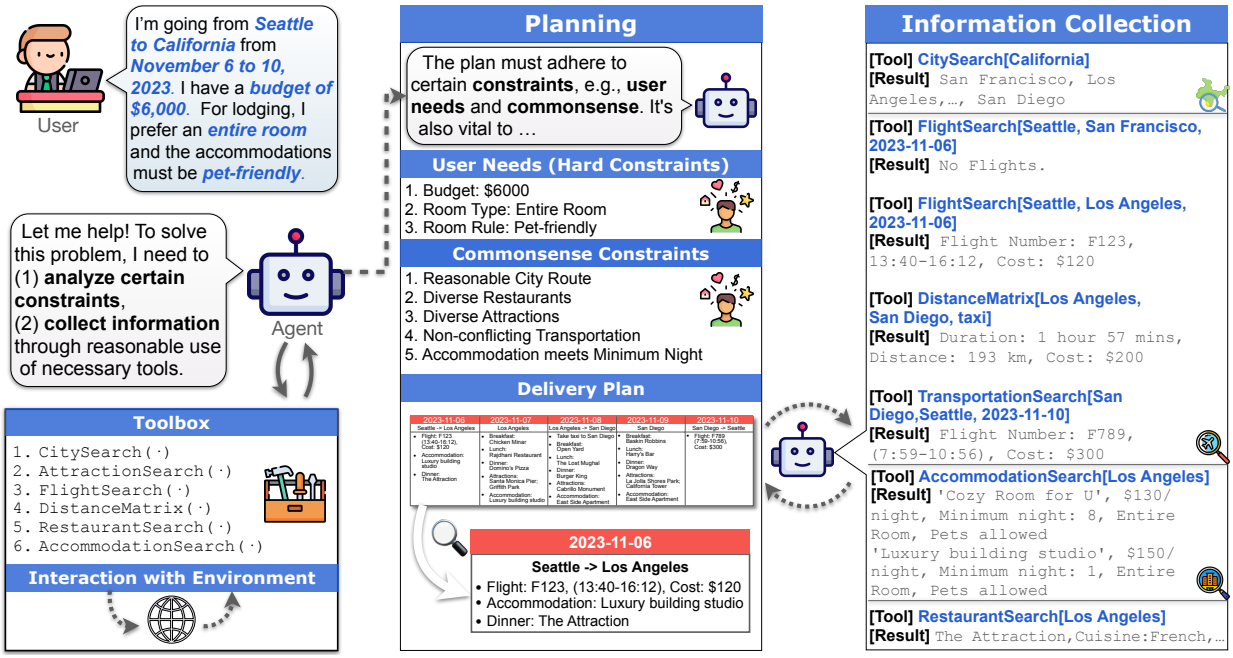

Figure 1. Overview of TravelPlanner. Given a query, language agents are tasked with employing various search tools to gather information. Based on the collected information, language agents are expected to deliver a plan that not only satisfies the user's needs specified in the query but also adheres to commonsense constraints.

enough time): *1)* Planning a multi-day itinerary is inherently *long-horizon*, involving a large number of interdependent decisions on places, lodging, transportation, dining, etc. *2)* Travel planning involves many *constraints*, ranging from explicit constraints such as budget and various user needs to implicit commonsense constraints, e.g., people cannot teletransport to another city without using some means of transportation. *3)* Travel planning requires strong *agency* to proactively acquire necessary information using various tools (e.g., to search flights and restaurants) from the partially observable environment and deliberate over the collected information to further the planning while being mindful of all the explicit and implicit constraints. Planning tasks of such complexity are out of the reach of prior AI agents (Russell & Norvig, 2010).

TravelPlanner provides a rich sandbox environment with around four million data entries crawled from the Internet that can be accessed via six tools. We also meticulously curate 1225 diverse user queries (along with their reference plans), each imposing a different combination of constraints. A representative example is illustrated in Figure 1.

We comprehensively evaluate five LLMs, such as GPT-4 (OpenAI, 2023), Gemini (G Team et al., 2023), and Mixtral (Jiang et al., 2024), and four planning strategies, such as ReAct (Yao et al., 2022) and Reflexion (Shinn et al., 2023), on their capability of delivering complete plans and following constraints. The main findings are as follows:

• State-of-the-art LLMs cannot handle complex planning tasks like those in TravelPlanner. GPT-4 successfully produces a plan that meets all the constraints for a few tasks (0.6%), while all other LLMs fail to complete any tasks.

• Existing planning strategies such as ReAct and Reflexion, which may be effective for simpler planning settings, are insufficient for the multi-constraint tasks in TravelPlanner. They often fail to convert their reasoning into the right actions and to keep track of global or multiple constraints. Language agents need more sophisticated planning strategies to approach human-level planning.

• Further analyses reveal many common failure modes of existing language agents, such as argument errors in tool use, being trapped in dead loops, and hallucinations.

Although most of our findings lean negatively toward the current language agents, we should note that the mere possibility for an artificial agent to tackle such a complex task is non-trivial progress in itself. TravelPlanner provides a challenging yet meaningful testbed for future agents to hill-climb toward human-level planning in complex settings.

Finally, a silver lining: while our well-trained human annotators averagely take 12 minutes to manually annotate a plan, a language agent can produce a plan in just 1–2 minutes automatically. Perhaps one day, language agents will become capable enough to help automate away many of such tedious tasks for us.

2. Related Work

2.1. Large Language Model based Agents

Empowered by large language models (LLMs), language agents have the capability to decompose complex tasks and arrive at solutions through a series of reasoned actions. Notable examples such as AutoGPT (AutoGPT, 2023), BabyAGI (Nakajima, 2023), and HuggingGPT (Shen et al., 2023) have illuminated the community with their impressive abilities. Current LLM-powered language agents, equipped with Memory, Tool-use, and Planning modules, have seen a substantial improvement in their general abilities (Weng, 2023). Memory in language agents refers to their ability to acquire and process information. It is divided into two types: long-term memory, which is the parametric memory inherent in LLMs, and short-term memory, also known as in-context learning (Brown et al., 2020) or working memory. Techniques like memory summarization (Chen et al., 2023; Zhou et al., 2023; Liang et al., 2023) and retrieval (Andreas, 2022; Park et al., 2023; Zhong et al., 2023) are widely employed to enhance the memory capabilities of language agents. Moreover, by interacting with external tools, language agents expand their potential capabilities significantly. This tool-augmentation paradigm has been validated as effective in previous work (Nakano et al., 2021; Lu et al., 2023; Ge et al., 2023; Xie et al., 2023). We further discuss the planning module in Section 2.2.

2.2. Planning

Planning, a hallmark of human intelligence, entails a sequence of actions that involve decomposing tasks, searching for solutions, and making final decisions (Hayes-Roth & Hayes-Roth, 1979; Grafman et al., 2004; Su, 2023). This skill is crucial for achieving human-level intelligence and has been widely studied in areas such as robotics (McDermott, 1992; Alterovitz et al., 2016) and transportation scheduling (Cross & Estrada, 1994; Pinedo, 2005). The emergence of language agents powered by LLMs has further intensified discussions around their planning capabilities (Liu et al., 2023a; Valmeekam et al., 2023). Previous research has demonstrated that language agents can effectively decompose tasks and engage in step-by-step reasoning, leading to significant improvements (Wei et al., 2022; Yuan et al., 2023; Zheng et al., 2024). Furthermore, to optimize solution searches in fewer steps, classical data structures like trees and graphs have been employed in prior studies (Yao et al., 2023; Besta et al., 2023), enhancing the planning capabilities of language agents. In addition, methods involving feedback from the environment (Yao et al., 2022; Shinn et al., 2023) have also been shown to be beneficial. However, while these planning abilities have shown promise in specific tasks, the effectiveness of these planning strategies in scenarios with multiple constraints remains uncertain.

2.3. Evaluation of Language Agents

Previous studies typically assess LLM-powered language agents in focused domains: arithmetic reasoning targeting correct solutions (Roy & Roth, 2015; Cobbe et al., 2021; Patel et al., 2021); tool-use evaluating agents' proficiency in employing tools and reporting results (Li et al., 2023; Xu et al., 2023; Zhuang et al., 2023); and web navigation, testing agents' ability to locate specific websites (Deng et al., 2023; Zhou et al., 2024; Liu et al., 2024). However, the complexity of the real-world implies that previous evaluation methods, which focus on single objective and fixed ground truths, may fall short of capturing the full scope of agents' capabilities. To address this, we introduce TravelPlanner for comprehensive evaluations, assessing whether language agents can generate feasible solutions facing various objectives, referred to as constraints in this paper.

3. TravelPlanner

3.1. Overview

We introduce TravelPlanner, a benchmark crafted for evaluating language agents in tool-use and complex planning within multiple constraints. Grounding to travel planning, a real-world use-case that naturally includes diverse constraints such as user needs and commonsense constraints, TravelPlanner evaluates whether agents can develop flexible travel plans by collecting information via diverse tools and making decisions while satisfying the constraints.

TravelPlanner comprises 1225 queries in total. The queries in TravelPlanner are divided into nine groups. This classification is based on two criteria: the duration of travel and the number of hard constraints. The dataset is divided into the training, validation, and test set. The training set includes 5 queries per group with human-annotated plans (45 pairs in total), the validation set includes 20 queries per group (180 in total), and the test set includes 1000 queries. Detailed distributions are shown in Table A.1.

3.2. Constraint Introduction

In order to assess whether agents can perceive, understand, and satisfy various constraints to formulate a feasible plan, as outlined in Table 1, we include three types of constraints:

• **Environment Constraints**: The real-world is dynamic, necessitating agents to be adaptable. For instance, flights to a particular destination may be unavailable at certain times (e.g., **no flights from Seattle to San Francisco in Figure 1**), possibly because tickets are sold out. In such cases, the agent must dynamically seek an alternative, like changing the destination of the flight or the way of transportation. To simulate this, we introduce environment constraints within TravelPlanner to test the adaptability of agents in planning.

Table 1. Constraint description. The environment constraints are manifested through the feedback received from the environment, assessing whether the language agent can adjust its plan appropriately. The commonsense constraints and hard constraints are evaluated based on how well the language agent's plan aligns with these specific criteria.

Constraint	Description
Environment Constraint	
Unavailable Transportation	There is no available flight or driving information between the two cities.
Unavailable Attractions	There is no available attraction information in the queried city.
Commonsense Constraint	
Within Sandbox	All information in the plan must be within the closed sandbox; otherwise, it will be considered a hallucination.
Complete Information	No key information should be left out of the plan, such as the lack of accommodation during travel.
Within Current City	All scheduled activities for the day must be located within that day's city(s).
Reasonable City Route	Changes in cities during the trip must be reasonable.
Diverse Restaurants	Restaurant choices should not be repeated throughout the trip.
Diverse Attractions	Attraction choices should not be repeated throughout the trip.
Non-conf. Transportation	Transportation choices within the trip must be reasonable. For example, having both "self-driving" and "flight" would be considered a conflict.
Minimum Nights Stay	The number of consecutive days spent in a specific accommodation during the trip must meet the corresponding required minimum number of nights' stay.
Hard Constraint	
Budget	The total budget of the trip.
Room Rule	Room rules include "No parties", "No smoking", "No children under 10", "No pets", and "No visitors".
Room Type	Room types include "Entire Room", "Private Room", "Shared Room", and "No Shared Room".
Cuisine	Cuisines include "Chinese", "American", "Italian", "Mexican", "Indian", "Mediterranean", and "French".
Transportation	Transportation options include "No flight" and "No self-driving".

Table 2. The number of data entries in the database.

Tool	Data Entries (#)
CitySearch	312
FlightSearch	3 827 361
DistanceMatrix	17 603
RestaurantSearch	9552
AttractionSearch	5303
AccommodationSearch	5064

• **Commonsense Constraints**: Agents, functioning in real-world and serving humans, should consider commonsense when designing plans. For instance, repeatedly visiting the same attraction is not typical. To evaluate agents' understanding and utilization of commonsense during planning, we include the commonsense constraint in TravelPlanner.

• **Hard Constraints**: A crucial ability for agents is to effectively satisfy personalized user needs. To evaluate this, TravelPlanner incorporates various user needs, such as budget constraints. These user needs are termed hard constraints. The hard constraint measures the agent's generalization ability with regard to different user needs.

3.3. Benchmark Construction Pipeline

This section outlines the construction pipeline of TravelPlanner, which involves the following steps: *1)* Environment and evaluation setup. *2)* Diverse travel query design. *3)* Reference plan annotation. *4)* Quality check.

Environment Setting. In TravelPlanner, we create a static and closed sandbox environment for consistent and unbiased evaluations. This setup ensures that all agents access the same unchanging information from our static databases, avoiding the variability and potential biases introduced by dynamic data. To offer various travel options that align with the real-world, we ensure the database for each tool in TravelPlanner contains rich information. The database size of each tool is listed in Table 2. For more tool details, please refer to Appendix A.2 and A.3. Additionally, agents are instructed to use the "NotebookWrite" tool to record necessary information for planning. This tool is integrated to evaluate agents' working memory management and prevents maximum token limit caused by context accumulation.

Query Construction. To create diverse queries for TravelPlanner, we begin with several fundamental elements, including departure city, destination, and specific date range, randomly chosen to form the skeleton of each query. Subsequently, we adjust the duration of travel and the number of hard constraints to create different levels of complexity.

The duration of the travel—3, 5, or 7 days—determines the number of cities included in the plan. Specifically, 3-day plans focus on one city, while 5 days and 7 days involve visiting one randomly chosen state, with trips to 2 cities for the 5-day plans and 3 cities for the 7-day plans, respectively. A greater number of days requires more frequent tool usage by language agents, thus increasing the difficulty of managing the long-horizon aspect of planning. The uncertain destination challenges agents to decide on multiple cities, where they must consider factors such as inter-city connectivity.

Furthermore, we introduce diverse user needs as hard constraints to add further complexity and realism. The difficulty levels are categorized as follows:

• **Easy**: Queries at this level are primarily budget-constrained for a single person. The initial budget for each query is determined using a set of crafted heuristic rules.

• **Medium**: In addition to budget constraints, medium queries introduce an additional hard constraint, randomly selected from a constraint pool including cuisine type, room type, and room rule. Furthermore, the number of people varies between 2 and 8, which influences the calculation of costs for both transportation and accommodation.

• **Hard**: Hard queries include additional transportation preference into the constraint pool, along with all the constraints in medium queries. Each hard query contains three hard constraints randomly selected from the constraint pool.

This method ensures the diversity of queries. Minor changes in these elements can lead to significantly different plans. Finally, based on elements, we utilize GPT-4 (OpenAI, 2023) to generate natural language queries.

Human Annotation. To ensure every query has at least one feasible plan, we invite 20 graduate students to meticulously annotate plans for synthesized queries. One plan is deemed eligible only if it meets all the constraints outlined in our evaluation script, which is detailed in Section 3.4. This rigorous process resulted in the creation of 1225 validated query-plan pairs. We pay annotators an average of \$0.80 for each plan they annotate.

Quality Control. To ensure the quality of each natural language query and its corresponding annotated plan, the authors performed a detailed review of every query and plan, rectifying any errors found. Additionally, to ensure the challenges, we re-calibrate each query's budget using the costs from corresponding human-annotated plans. This approach replaces the initial heuristic-generated budgets, which might be too high, thus reducing the number of feasible plans. Through multiple stages of human verification, we ensure the high quality of each query in TravelPlanner and the presence of at least one feasible solution.

3.4. Evaluation

To ensure a comprehensive evaluation of the plans offered by agents, we assess them from multiple dimensions. Specifically, we first extract key components[1], including transportation, restaurants, attractions, and accommodations, which are initially presented as natural language. These components are then organized into a formally structured plan, which will be evaluated automatically through pre-defined scripts. The evaluation criteria include the following:

• **Delivery Rate**: This metric assesses whether agents can successfully deliver a final plan within a limited number of steps. Falling into dead loops, experiencing numerous failed attempts, or reaching the maximum number of steps (30 steps in our experimental setting) will result in failure.

• **Commonsense Constraint Pass Rate**: Comprising eight commonsense dimensions, this metric evaluates whether a language agent can incorporate commonsense into their plan without explicit instructions.

• **Hard Constraint Pass Rate**: This metric measures whether a plan satisfies all explicitly given hard constraints in the query, which aims to test the agents' ability to adapt their plans to diverse user needs.

• **Final Pass Rate**: This metric represents the proportion of feasible plans that meet all aforementioned constraints among all tested plans. It serves as an indicator of agents' proficiency in producing plans that meet a practical standard.

We do not separately assess environment constraints since their impact is inherently reflected in the "Within Sandbox" and "Complete Information" metrics. For instance, when cities lack transportation or attractions, agents typically resort to hallucination or opt not to provide an answer, reflecting the impact of environment constraints.

For the Commonsense Constraint Pass Rate and Hard Constraint Pass Rate, we utilize two evaluation strategies: *micro* and *macro*. The *micro* strategy calculates the ratio of passed constraints to the total number of constraints. The **Micro Pass Rate** is defined as:

$$\text{Micro Pass Rate} = \frac{\sum_{p \in P} \sum_{c \in C_p} \mathbb{1}_{\text{passed}(c,p)}}{\sum_{p \in P} |C_p|}, \quad (1)$$

where P represents the set of all plans being evaluated, C_p denotes the set of constraints applicable to a specific plan p in P, and passed(X, Y) is a function determining whether Y meets constraints X.

The *macro* strategy calculates the ratio of plans that pass all commonsense or hard constraints among all tested plans. We define the **Macro Pass Rate** as:

$$\text{Macro Pass Rate} = \frac{\sum_{p \in P} \mathbb{1}_{\text{passed}(C_p,p)}}{|P|}. \quad (2)$$

These two metrics evaluate an agent's capability of following individual constraints vs. all the constraints holistically.

3.5. Sole-Planning Setting

While TravelPlanner is designed to assess the overall abilities of agents in tool-use and planning (two-stage mode),

[1] In our experiments, we use GPT-4-Turbo for this extraction process. Please refer to Appendix B.3.5 for more details.

Table 3. Main results of different LLMs and planning strategies on the TravelPlanner validation and test set. The best results are marked in bold. When the collected information is insufficient, Gemini Pro tends to directly refuse to provide the plan. Interviews with annotators reveal that manually annotating a plan averagely takes around 12 minutes. However, language agents, such as GPT-3.5-Turbo, can accomplish this task in just 1 to 2 minutes, showcasing their efficiency.

	Validation (#180)						**Test** (#1000)					
	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
Two-stage												
Mistral-7B-32K (Jiang et al., 2023)	8.9	5.9	0	0	0	0	7.0	4.8	0	0	0	0
Mixtral-8×7B-MoE (Jiang et al., 2024)	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
Gemini Pro (G Team et al., 2023)	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
GPT-3.5-Turbo (OpenAI, 2022)	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
GPT-4-Turbo (OpenAI, 2023)	**89.4**	**61.1**	**2.8**	**15.2**	**10.6**	**0.6**	**93.1**	**63.3**	**2.0**	**10.5**	**5.5**	**0.6**
Sole-planning												
$\text{Direct}_{\text{GPT-3.5-Turbo}}$	100	60.2	4.4	11.0	2.8	0	100	59.5	2.7	9.5	4.4	0.6
$\text{CoT}_{\text{GPT-3.5-Turbo}}$	100	66.3	3.3	11.9	5.0	0	100	64.4	2.3	9.8	3.8	0.4
$\text{ReAct}_{\text{GPT-3.5-Turbo}}$	82.2	47.6	3.9	11.4	6.7	0.6	81.6	45.9	2.5	10.7	3.1	0.7
$\text{Reflexion}_{\text{GPT-3.5-Turbo}}$	93.9	53.8	2.8	11.0	2.8	0	92.1	52.1	2.2	9.9	3.8	0.6
$\text{Direct}_{\text{Mixtral-8x7B-MoE}}$	100	68.1	5.0	3.3	1.1	0	99.3	67.0	3.7	3.9	1.6	0.7
$\text{Direct}_{\text{Gemini Pro}}$	93.9	65.0	8.3	9.3	4.4	0.6	93.7	64.7	7.9	10.6	4.7	2.1
$\text{Direct}_{\text{GPT-4-Turbo}}$	**100**	**80.4**	**17.2**	**47.1**	**22.2**	**4.4**	**100**	**80.6**	**15.2**	**44.3**	**23.1**	**4.4**

we also setup a simplified mode solely evaluating agents' planning skills (sole-planning mode). In this setting, we utilize human-annotated plans to pre-determine the destination cities, and provide detailed and necessary information directly to agents, such as restaurants in the provided cities. This eliminates the need for tool calling as agents don't need to collect information from scratch via tools anymore.

4. Experiments

We evaluate the performance of various LLMs and planning strategies on TravelPlanner. In the two-stage mode, we use the ReAct (Yao et al., 2022) framework for information collection, which is recognized for its effective iteration with tools (Zhuang et al., 2023) while varying the foundation LLMs. This approach allows us to assess how different LLMs perform under a uniform tool-use framework. The agents are required to give the plan directly based on the information collected by themselves, without employing any other planning strategies. In the sole-planning mode, our evaluation goes beyond varying LLMs to include different planning strategies. This aims to assess if the strategies proven effective in other planning benchmarks maintain their efficacy in TravelPlanner. All experiments are conducted in a zero-shot setting.

4.1. Baselines

Greedy Search. To evaluate the effectiveness of traditional rule-based strategies within TravelPlanner, we include greedy search as a baseline and set cost as the optimization objective. Please refer to Appendix B.1 for more details.

LLMs. Due to the long context window requirement of ReAct and the massive information as text, we limit our consideration to LLMs capable of handling inputs exceeding 8K in length. As a result, our selection includes three closed-source LLMs: **GPT-3.5-Turbo** (OpenAI, 2022), **GPT-4-Turbo** (OpenAI, 2023), and **Gemini Pro** (G Team et al., 2023), as well as two open-source LLMs: **Mistral-7B-32K** (Jiang et al., 2023) and **Mixtral-8x7B-MoE** (Jiang et al., 2024). For all these models, we adopt the official instruction formats whenever available.

Planning Strategies. To explore the effectiveness of current planning strategies, we evaluate four representative ones: **Direct**, **ZS-CoT** (Wei et al., 2022), **ReAct** (Yao et al., 2022), and **Reflexion** (Shinn et al., 2023). For the implementation details, please refer to Appendix B.1. We do not include **ToT** (Yao et al., 2023) and **GoT** (Besta et al., 2023) because they require extensive exploration of the search space, prohibitively costly for problems as complex as TravelPlanner. Also, given their performance close to ReAct in complex tasks (Zhuang et al., 2024), the potential benefits of these methods may be limited.

4.2. Main Results

In this section, we discuss the performance of various LLMs and planning strategies on TravelPlanner (Table 3). We have the following observations:

TravelPlanner poses a significant challenge. In the two-stage mode, GPT-4-Turbo with ReAct achieves only 0.6% in the final pass rate, and none of the other LLMs can pass any

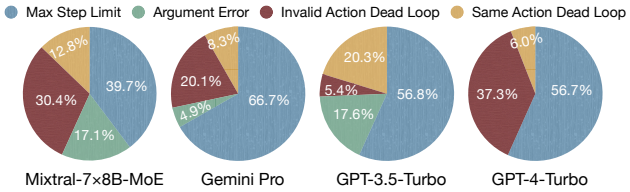

Figure 2. Tool-use error distribution on the test set. An early stop will be triggered if the agent either makes three consecutive failed attempts or repetitive actions, indicating a dead loop.

of the tasks. Even given all the necessary information in the sole-planning mode, existing planning strategies like ReAct and Reflexion still struggle with planning in TravelPlanner, even though they have shown their effectiveness in more conventional planning tasks. It is noteworthy that the best-performing agent still falls short on hard constraints even when compared to greedy search. This poor performance underlines the difficulty of TravelPlanner and shows that current agents still struggle with complex planning.

Agents show a substantial gap between the two modes. The comparison of the two modes reveals the agents' struggles in fiddling with both information collection and planning. Across all metrics, the scores of any model in the two-stage mode are lower than those in the sole-planning mode, with the largest gap reaching over 30%. Similar to humans, language agents also seem to have a limited "cognitive capacity" and their performance deteriorates when multitasking. We provide a further analysis in Section 5.2.

Agents struggle in obtaining a high macro pass rate. While some agents achieve high micro scores, their macro scores remain low. This pattern shows that although agents manage to satisfy some constraints, they often overlook some other constraints in the meantime. Consequently, this indicates the current agents fail to consider multiple constraints holistically, a critical requirement for navigating the complex tasks in TravelPlanner.

In summary, TravelPlanner poses a great challenge to current agents. The SoTA LLMs and planning strategies, which often show equal or superior to human-level performance on many traditional tasks, are still far from sufficient for complex planning tasks that humans are capable of. TravelPlanner provides a challenging yet meaningful benchmark for the development of more capable language agents.

5. In-Depth Analysis

5.1. Tool-Use Error Analysis

As shown in Table 3, even based on GPT-4-Turbo, agents still make mistakes in the process of information collection and thus fail to deliver a plan. This problem is more severe in Gemini Pro and Mixtral. To delve into the underlying causes, we categorize all error types in Figure 2. We find:

Table 4. Constraint pass rate of GPT-4-Turbo on test set. The results of the sole-planning mode are based on the Direct strategy.

Constraint Type	Two-stage			Sole-planning		
	Easy	Medium	Hard	Easy	Medium	Hard
Commonsense Constraint						
Within Sandbox	37.4	31.2	33.9	76.4	71.5	79.3
Complete Information	53.4	52.9	58.0	94.5	96.4	96.2
Within Current City	69.3	67.3	68.3	89.1	80.8	82.4
Reasonable City Route	44.5	45.6	54.9	99.4	99.7	99.1
Diverse Restaurants	85.1	81.4	86.8	91.1	89.8	87.8
Diverse Attractions	94.3	90.4	94.0	100.0	100.0	100.0
Non-conf. Transportation	70.1	73.3	83.1	60.1	56.5	87.5
Minimum Nights Stay	46.8	46.2	51.1	37.4	28.8	30.1
Hard Constraint						
Budget	10.1	8.4	4.4	37.4	35.1	25.1
Room Rule	-	5.6	11.3	-	31.5	43.6
Cuisine	-	10.8	11.4	-	57.5	46.7
Room Type	-	12.4	13.8	-	45.7	56.7
Transportation	-	-	18.6	-	-	77.5
Final						
Final Pass Rate	1.1	0.3	0.3	8.0	2.7	2.2

Table 5. Comparison of the numbers of different tool uses between agent (GPT-4-Turbo) and reference. The results of agent are based on the number of entries written into the "Notebook".

Average	Agent			Reference		
	3-day	5-day	7-day	3-day	5-day	7-day
FlightSearch	0.7	1.2	0.8	2.0	3.0	4.0
DistanceMatrix	0.3	0.6	1.2	2.0	3.0	4.0
RestaurantSearch	0.9	1.5	2.4	1.0	2.0	3.0
AttractionSearch	0.9	1.7	2.4	1.0	2.0	3.0
AccommodationSearch	0.9	1.6	2.5	1.0	2.0	3.0

1) **Agents incorrectly use tools.** Except for GPT-4-Turbo, other LLMs-based agents all have argument error problems to varying degrees. It sheds light that the use of simple tools still poses a significant challenge for agents. *2)* **Agents trap in dead loops.** Even with GPT-4-Turbo, invalid actions and repetitive action loops contribute to 37.3% and 6.0% of errors, respectively. Despite receiving feedback that actions are invalid or yield null results, agents persistently repeat these actions. This suggests that agents fail to dynamically adjust their plans based on environment feedback.

5.2. Planning Error Analysis

We detail each constraint pass rate in Table 4, from which we have the following observations:

The number of hard constraints affects the performance of agents. Agents consistently exhibit pass rates below 10% across all levels of difficulty, and this performance deteriorates further as more constraints are introduced. This trend highlights that current agents struggle with multi-constraint tasks, a key aspect of TravelPlanner.

Comprehensive information collection is essential for agents to plan effectively. In the sole-planning mode, agent performance improves over the two-stage mode. To investi-

Figure 3. Case studies of failures. Agents fail to finalize plans due to repeated errors like incorrect dates, confusion with information details leading to hallucinatory answers, and disconnects between reasoning and actions. All cases are gained from GPT-4-Turbo based agents. For details of GPT-4-Turbo with the Reflexion strategy, please refer to Appendix B.2. We provide more cases in Appendix C.2.

gate this, Table 5 shows agents in two-stage mode use tools less effectively compared to the reference plans. This comparison indicates that agents often fail to finish completed information collection. Consequently, they either generate made-up information or omit specific details. This leads to low pass rates for the "Within Sandbox" and "Complete Information" constraints. Additionally, this discrepancy becomes more pronounced with an increase in the duration of travel. This emphasizes the need for agents to improve their capabilities in long-horizon tasks.

Agents struggle with global planning scenarios. Global constraints "Minimum Nights Stay" and "Budget" demand a holistic approach to planning, necessitating that agents not only assess their current decisions but also anticipate future implications. Current LLMs' auto-regressive nature limits them to independently obtain outcomes from multiple future branches. This highlights the necessity and urgent need of new strategies, such as backtracking for adjusting or employing heuristic methods for forward-looking planning.

5.3. Case Studies

To investigate the drawbacks of current agents in-depth, we provide several failure cases in Figure 3. We conclude with the following features:

Agents fail to complete a plan due to the inability to rectify persistent errors. In tool-use scenarios, agents often fail to deliver a plan even when all preceding steps are executed correctly. Further investigation reveals that this issue often stems from incorrect date inputs. As shown in the left part of Figure 3, despite correct execution, agents repeatedly use incorrect dates. This leads to null results, as the data in the TravelPlanner sandbox is based on 2022. Such repeated failures eventually cause the agents to stop planning. This indicates a significant limitation: current agents cannot self-correct their initial and incorrect assumptions.

Agents produce hallucinatory answers due to information confusion. To understand why agents provide hallucinatory answers even when supplied with sufficient information in the sole-planning mode, we conduct a detailed analysis. We observe a tendency for agents to confuse one piece of information with another. As shown in the middle part of Figure 3, agents mistakenly use the same flight number for both departure and return flights. Such errors result in hallucinations, as the information provided in the plan does not align with the data in the sandbox. This suggests that agents might be lost when faced with mass information, known as "Lost in the Middle" (Liu et al., 2023b).

Agents struggle to align their actions with their reasoning. To understand the reasons behind the lower delivery rate of Reflexion (Shinn et al., 2023), we examine specific examples. As illustrated in the right part of Figure 3, we observe a discrepancy between what agents think and what they do. Despite recognizing the necessity to minimize costs, they tend to randomly select items, some of which may be more expensive. This discrepancy demonstrates that agents struggle to synchronize their actions with their analytical reasoning, severely impeding their delivery rate.

6. Conclusion

We introduce TravelPlanner, a benchmark grounded in real-world scenarios, designed to assess the multi-constraint planning and tool-use abilities of current language agents. Our benchmark presents a significant challenge: even the most advanced language agent frameworks only achieve

a mere 0.6% score in the final pass rate. Further analysis shows that these agents are unable to take all constraints into consideration to deliver feasible plans.

TravelPlanner's intricate logic and general applicability stand as vital components in the progressive development of language agents, thus contributing to the broader quest for AI abilities. We envision TravelPlanner as a catalyst for future research, aiming to enhance agents' performance in increasingly complex scenarios, hill-climbing towards human-level cognitive capabilities.

Acknowledgements

The authors would like to thank annotators from the OKAgent group for their manual annotation. The authors would also like to thank Siyu Yuan, Yikai Zhang, and Aili Chen from Fudan University as well as the anonymous reviewers for their valuable comments. This research was sponsored in part by Cisco and YS's startup funds.

Impact Statement

TravelPlanner aims to provide an effective benchmark for complex planning in future research. Some of the data in the TravelPlanner environment is derived from publicly available data on the Internet, and the content involved does not represent the authors' viewpoints. We realize that everyone's definition of commonsense may be different. Our current evaluation criteria are based on the authors' consensus, and we encourage additional discussions to enrich our commonsense dimension, aiming for a more thorough evaluation. We have released our evaluation scripts to foster innovation and aid the development of new methods. We encourage the use of evaluation feedback in training set, such as implementing reinforcement learning techniques, to enhance learning. However, we strictly prohibit any form of cheating in the validation and test sets to uphold the fairness and reliability of the benchmark's evaluation process.

References

Alterovitz, R., Koenig, S., and Likhachev, M. Robot planning in the real world: Research challenges and opportunities. *Ai Magazine*, 2016.

Andreas, J. Language models as agent models. In *Findings of EMNLP*, 2022.

AutoGPT. Autogpt, 2023. URL https://github.com/Significant-Gravitas/AutoGPT.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Proceedings of NeurIPS*, 2020.

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. Deep blue. *Artificial intelligence*, 2002.

Chen, H., Pasunuru, R., Weston, J., and Celikyilmaz, A. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cross, S. and Estrada, R. Dart: an example of accelerated evolutionary development. In *Proceedings of Workshop on RSP*, 1994.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. In *Proceedings of NeurIPS*, 2023.

G Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Ge, Y., Hua, W., Ji, J., Tan, J., Xu, S., and Zhang, Y. Openagi: When llm meets domain experts. In *Proceedings of NeurIPS*, 2023.

Georgievski, I. and Aiello, M. Htn planning: Overview, comparison, and beyond. *Artificial Intelligence*, 2015.

Grafman, J., Spector, L., and Rattermann, M. J. Planning and the brain. In *The cognitive psychology of planning*. 2004.

Hayes-Roth, B. and Hayes-Roth, F. A cognitive model of planning. *Cognitive science*, 1979.

Ho, M. K., Saxe, R., and Cushman, F. Planning with theory of mind. *Trends in Cognitive Sciences*, 2022.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of ICML*, 2022.

Ichter, B., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C.,

Rao, K., Sermanet, P., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K., Kuang, Y., Jesmonth, S., Joshi, N. J., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of CoRL*, 2022.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Karpas, E. and Magazzeni, D. Automated planning for robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2020.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V. V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., and Misra, V. Solving quantitative reasoning problems with language models. In *Proceedings of NeurIPS*, 2022.

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., and Li, Y. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of EMNLP*, 2023.

Liang, X., Wang, B., Huang, H., Wu, S., Wu, P., Lu, L., Ma, Z., and Li, Z. Unleashing infinite-length input capacity for large-scale language models with self-controlled memory system. *arXiv preprint arXiv:2304.13343*, 2023.

Lin, J., Tomlin, N., Andreas, J., and Eisner, J. Decision-oriented dialogue for human-ai collaboration. *arXiv preprint arXiv:2305.20076*, 2023.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023b.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. Agentbench: Evaluating llms as agents. In *Proceedings of ICLR*, 2024.

Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.-W., Wu, Y. N., Zhu, S.-C., and Gao, J. Chameleon: Plug-and-play compositional reasoning with large language models. In *Proceedings of NeurIPS*, 2023.

Mattar, M. G. and Lengyel, M. Planning in the brain. *Neuron*, 2022.

McDermott, D. Robot planning. *AI magazine*, 1992.

Nakajima, Y. Task-driven autonomous agent utilizing gpt-4, pinecone, and langchain for diverse applications, 2023.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

OpenAI. Chatgpt, 2022. URL https://openai.com/blog/chatgpt.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of UIST*, 2023.

Patel, A., Bhattamishra, S., and Goyal, N. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL*, 2021.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Pinedo, M. *Planning and scheduling in manufacturing and services*. Springer, 2005.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *Proceedings of ICLR*, 2024.

Roy, S. and Roth, D. Solving general arithmetic word problems. In *Proceedings of EMNLP*, 2015.

Russell, S. J. and Norvig, P. *Artificial intelligence a modern approach*. 2010.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Proceedings of NeurIPS*, 2023.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. HuggingGPT: Solving AI tasks with chatGPT and its friends in hugging face. In *Proceedings of NeurIPS*, 2023.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of NeurIPS*, 2023.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

Silver, D., Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., et al. The predictron: End-to-end learning and planning. In *Proceedings of ICML*, 2017.

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., and Su, Y. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of ICCV*, 2023.

Su, Y. Language agents: a critical evolutionary step of artificial intelligence. 2023. URL https://yusu.substack.com/p/language-agents.

Sumers, T. R., Yao, S., Narasimhan, K., and Griffiths, T. L. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Valmeekam, K., Olmo, A., Sreedharan, S., and Kambhampati, S. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Proceedings of NeurIPS*, 2023.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS*, 2022.

Weng, L. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023. URL https://lilianweng.github.io/posts/2023-06-23-agent/.

Xie, T., Zhou, F., Cheng, Z., Shi, P., Weng, L., Liu, Y., Hua, T. J., Zhao, J., Liu, Q., Liu, C., Liu, L. Z., Xu, Y., Su, H., Shin, D., Xiong, C., and Yu, T. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2023.

Xu, Q., Hong, F., Li, B., Hu, C., Chen, Z., and Zhang, J. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*, 2023.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *Proceedings of ICLR*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of NeurIPS*, 2023.

Yuan, S., Chen, J., Fu, Z., Ge, X., Shah, S., Jankowski, C., Xiao, Y., and Yang, D. Distilling script knowledge from large language models for constrained language planning. In *Proceedings of ACL*, 2023.

Zhang, Y., Yuan, S., Hu, C., Richardson, K., Xiao, Y., and Chen, J. Timearena: Shaping efficient multitasking language agents in a time-aware simulation. *arXiv preprint arXiv:2402.05733*, 2024.

Zheng, B., Gou, B., Kil, J., Sun, H., and Su, Y. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Zhong, W., Guo, L., Gao, Q., and Wang, Y. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., et al. Webarena: A realistic web environment for building autonomous agents. In *Proceedings of ICLR*, 2024.

Zhou, W., Jiang, Y. E., Cui, P., Wang, T., Xiao, Z., Hou, Y., Cotterell, R., and Sachan, M. Recurrentgpt: Interactive generation of (arbitrarily) long text. *arXiv preprint arXiv:2305.13304*, 2023.

Zhuang, Y., Yu, Y., Wang, K., Sun, H., and Zhang, C. ToolQA: A dataset for LLM question answering with external tools. In *Proceedings of NeurIPS*, 2023.

Zhuang, Y., Chen, X., Yu, T., Mitra, S., Bursztyn, V., Rossi, R. A., Sarkhel, S., and Zhang, C. Toolchain*: Efficient action space navigation in large language models with a* search. In *Proceedings of ICLR*, 2024.

Appendices

Within this supplementary material, we elaborate on the following aspects:

- Appendix A: Benchmark Details
- Appendix B: Experiment Details
- Appendix C: Case Presentation

A. Benchmark Details

A.1. Dataset Distribution

In Table A.1, we list the detailed group distribution on training, validation and test set.

Table A.1. Dataset distribution.

	Days	Easy	Medium	Hard
Training (#45)	3-day	5	5	5
	5-day	5	5	5
	7-day	5	5	5
Validation (#180)	3-day	20	20	20
	5-day	20	20	20
	7-day	20	20	20
Test (#1000)	3-day	122	104	82
	5-day	116	114	121
	7-day	110	115	116

A.2. Tool Description

In Table A.2, we list the detailed tool description. The original data for each tool is sourced from publicly available Internet data. We then modify this data, which includes adding, deleting, and altering certain keys and values to suit our requirements. In this way, we effectively avoid the problem of data contamination. For more details, please refer to Appendix A.3.

Table A.2. Tool description and the number of data entries in the database.

Tool	Data Entries(#)	Description
CitySearch	312	Search cities in the given state.
FlightSearch	3 827 361	Search flight information for a specific date between two cities.
DistanceMatrix	17 603	Search the driving distance, time, and possible cost between two cities.
RestaurantSearch	9552	Search restaurants in the given city.
AttractionSearch	5303	Search attractions in the given city.
AccommodationSearch	5064	Search accommodations in the given city.
NotebookWrite	-	Write the selected data entry into the Notebook tool with a short description.

A.3. Environment Database Construction

FlightSearch For FlightSearch, we source original data from the Kaggle Flight Status Prediction dataset[2]. From this dataset, we extract data spanning from 2022-03-01 to 2022-04-01. We specifically included fields like "FlightDate", "DepTime", "ArrTime", "ActualElapsedTime", "Distance", "OriginCityName", and "DestCityName" while discarding other values. To incorporate "Price" into our dataset, we generate this value by multiplying the "Distance" by a random factor

[2] https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022?select=Combined_Flights_2022.csv

ranging from 0.2 to 0.5.

DistanceMatrix We utilize the Google Distance Matrix API[3] to calculate the driving distance and estimated travel time between two cities. For the "self-driving" and "taxi" modes of transportation, we calculate the 'Price' by multiplying the 'Distance' by factors of 1 and 0.15, respectively. To ensure consistency and reliability of data, we store the search results in our database, thereby creating a fixed dataset for our evaluations.

RestaurantSearch Our restaurant data is sourced from the Kaggle Zomato Restaurants Dataset[4]. From this dataset, we extract the "Restaurant Name" and "Average Cost" for each establishment. Subsequently, we randomly assign these restaurants to various cities. To align with the constraint requirements of TravelPlanner, we also randomly categorize each restaurant under the following cuisines: "Chinese", "American", "Italian", "Mexican", "Indian","Mediterranean", "Middle Eastern", "Korean", "Asian", "French".

AttractionSearch For AttractionSearch, we employ the Google Places API[5] to gather information about attractions in each city. In TravelPlanner, we retain essential details such as "Name", "Address", "Phone", "Website", "Latitude", and "Longtitue" for each attraction. To maintain data consistency and reliability, we store these search results in our database, creating a standardized dataset for our analyses.

AccommodationSearch Our accommodation data is obtained from the Kaggle Airbnb Open Data Dataset[6]. From this dataset, we extract key details "NAME", "room type", "price", "minimum nights", "review rate number", and "maximum occupancy". Items are then randomly assigned to various cities. To meet the specific constraint requirements of TravelPlanner, we also assign each item random room rules, including "No parties", "No smoking", "No children under 10", "No pets", and "No visitors".

B. Experiment Details

B.1. Baselines

Greedy Search To assess the effectiveness of traditional search algorithms in TravelPlanner, we integrate a greedy search approach, focusing on minimizing costs. For 5 or 7-day travel plans, the first one or two cities in the returned city search result are selected as destinations. The transportation choice is based on the lowest cost option among flights, taxis, and self-driving. The diet component involves selecting the restaurant with the lowest average cost. The cheapest accommodation is chosen for lodging. For attractions, we opt for a random selection for each day of the itinerary.

Planning Strategy Current planning strategies have shown effectiveness in traditional tasks like mathematical problem-solving, but their capability to handle the more complex and constrained scenarios like TravelPlanner remains to be seen. To explore this, we evaluate four distinct planning strategies on TravelPlanner: *1)* **Direct**: In this method, the query is input directly into the model along with instructions detailing the task and relevant information gathered. *2)* **ZS-CoT** (Wei et al., 2022): This strategy enhances the reasoning process by requiring intermediate steps. Building on the Direct method, we add the prompt "Let's think step by step" to elicit reasoning. *3)* **ReAct** (Yao et al., 2022): This strategy incorporates environmental feedback into the reasoning process. Thus, it enhances the language agent's reasoning ability by offering additional information. In TravelPlanner, we provide the cost associated with each entire day's plan as environmental feedback. *4)* **Reflexion** (Shinn et al., 2023): This approach utilizes a reflection model to provide high-level insights on previous erroneous attempts. Such reflective guidance aids language agents in identifying and correcting flawed reasoning. In order to control the cost, we conduct tests on Direct using four different models, while the other strategies are evaluated using GPT-3.5-Turbo. Detailed instructions for each strategy are available in Appendix B.3.

B.2. GPT-4-Turbo with Reflexion strategy in sole-planning mode.

We provide the results of GPT-4-Turbo with Reflexion strategy on validation set in Table B.3.

[3] https://developers.google.com/maps/documentation/distance-matrix/overview?hl=en
[4] https://www.kaggle.com/datasets/shrutimehta/zomato-restaurants-data
[5] https://developers.google.com/maps/documentation/places/web-service/overview?hl=en
[6] https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata

Table B.3. GPT-4-Turbo with Reflexion strategy on validation set.

	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro	
$\text{Reflexion}_{\text{GPT-4-Turbo}}$	80.6	62.9	6.1	52.4	40.0	3.3

B.3. Prompt List

B.3.1. Tool-use Prompt

We tailor the ReAct (Yao et al., 2022) framework to suit our specific requirements in TravelPlanner. An example of the instruction prompt for our needs is as follows:

```
Collect information for a query plan using interleaving 'Thought', 'Action', and '
Observation' steps. Ensure you gather valid information related to transportation, dining,
 attractions, and accommodation. All information should be written in Notebook, which will
 then be input into the Planner tool. Note that the nested use of tools is prohibited. '
Thought' can reason about the current situation, and 'Action' can have 8 different types:
(1) FlightSearch[Departure City, Destination City, Date]:
Description: A flight information retrieval tool.
Parameters:
Departure City: The city you'll be flying out from.
Destination City: The city you aim to reach.
Date: The date of your travel in YYYY-MM-DD format.
Example: FlightSearch[New York, London, 2022-10-01] would fetch flights from New York to
London on October 1, 2022.

(2) GoogleDistanceMatrix[Origin, Destination, Mode]:
Description: Estimate the distance, time and cost between two cities.
Parameters:
Origin: The departure city of your journey.
Destination: The destination city of your journey.
Mode: The method of transportation. Choices include 'self-driving' and 'taxi'.
Example: GoogleDistanceMatrix[Paris, Lyon, self-driving] would provide driving distance,
time and cost between Paris and Lyon.

(3) AccommodationSearch[City]:
Description: Discover accommodations in your desired city.
Parameter: City - The name of the city where you're seeking accommodation.
Example: AccommodationSearch[Rome] would present a list of hotel rooms in Rome.

(4) RestaurantSearch[City]:
Description: Explore dining options in a city of your choice.
Parameter: City - The name of the city where you're seeking accommodation.
Example: RestaurantSearch[Tokyo] would show a curated list of restaurants in Tokyo.

(5) AttractionSearch[City]:
Description: Find attractions in a city of your choice.
Parameter: City - The name of the city where you're seeking attractions.
Example: AttractionSearch[London] would return attractions in London.

(6) CitySearch[State]
Description: Find cities in a state of your choice.
Parameter: State - The name of the state where you're seeking cities.
Example: CitySearch[California] would return cities in California.

(7) NotebookWrite[Short Description]
Description: Writes a new data entry into the Notebook tool with a short description. This
 tool should be used immediately after FlightSearch, AccommodationSearch, AttractionSearch
, RestaurantSearch or GoogleDistanceMatrix. Only the data stored in Notebook can be seen
by Planner. So you should write all the information you need into Notebook.
```

```
Parameters: Short Description - A brief description or label for the stored data. You don't need to write all the information in the description. The data you've searched for will be automatically stored in the Notebook.
Example: NotebookWrite[Flights from Rome to Paris in 2022-02-01] would store the informatrion of flights from Rome to Paris in 2022-02-01 in the Notebook.

(8) Planner[Query]
Description: A smart planning tool that crafts detailed plans based on user input and the information stroed in Notebook.
Parameters:
Query: The query from user.
Example: Planner[Give me a 3-day trip plan from Seattle to New York] would return a detailed 3-day trip plan.
You should use as many as possible steps to collect engough information to input to the Planner tool.

Each action only calls one function once. Do not add any description in the action.

Query: {query}
```

B.3.2. Direct Planning Prompt

We provide the instruction prompt of Direct strategy as follows:

```
You are a proficient planner. Based on the provided information and query, please give me a detailed plan, including specifics such as flight numbers (e.g., F0123456), restaurant names, and accommodation names. Note that all the information in your plan should be derived from the provided data. You must adhere to the format given in the example. Additionally, all details should align with commonsense. The symbol '-' indicates that information is unnecessary. For example, in the provided sample, you do not need to plan after returning to the departure city. When you travel to two cities in one day, you should note it in the 'Current City' section as in the example (i.e., from A to B).

***** Example *****
Query: Could you create a travel plan for 7 people from Ithaca to Charlotte spanning 3 days, from March 8th to March 14th, 2022, with a budget of $30,200?
Travel Plan:
Day 1:
Current City: from Ithaca to Charlotte
Transportation: Flight Number: F3633413, from Ithaca to Charlotte, Departure Time: 05:38, Arrival Time: 07:46
Breakfast: Nagaland's Kitchen, Charlotte
Attraction: The Charlotte Museum of History, Charlotte
Lunch: Cafe Maple Street, Charlotte
Dinner: Bombay Vada Pav, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 2:
Current City: Charlotte
Transportation: -
Breakfast: Olive Tree Cafe, Charlotte
Attraction: The Mint Museum, Charlotte;Romare Bearden Park, Charlotte.
Lunch: Birbal Ji Dhaba, Charlotte
Dinner: Pind Balluchi, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 3:
Current City: from Charlotte to Ithaca
Transportation: Flight Number: F3786167, from Charlotte to Ithaca, Departure Time: 21:42, Arrival Time: 23:26
Breakfast: Subway, Charlotte
Attraction: Books Monument, Charlotte.
Lunch: Olive Tree Cafe, Charlotte
Dinner: Kylin Skybar, Charlotte
```

```
Accommodation: -

***** Example Ends *****

Given information: {text}
Query: {query}
Travel Plan:
```

B.3.3. REACT & REFLEXION PLANNING PROMPT

The instruction prompts for the React and Reflexion planning strategies are provided as follows:

```
You are a proficient planner. Based on the provided information and query, please give me
a detailed plan, including specifics such as flight numbers (e.g., F0123456), restaurant
names, and hotel names. Note that all the information in your plan should be derived from
the provided data. You must adhere to the format given in the example. Additionally, all
details should align with common sense. Attraction visits and meals are expected to be
diverse. The symbol '-' indicates that information is unnecessary. For example, in the
provided sample, you do not need to plan after returning to the departure city. When you
travel to two cities in one day, you should note it in the 'Current City' section as in
the example (i.e., from A to B). Solve this task by alternating between Thought, Action,
and Observation steps. The 'Thought' phase involves reasoning about the current situation.
The 'Action' phase can be of two types:
(1) CostEnquiry[Sub Plan]: This function calculates the cost of a detailed sub plan, which
 you need to input the people number and plan in JSON format. The sub plan should
encompass a complete one-day plan. An example will be provided for reference.
(2) Finish[Final Plan]: Use this function to indicate the completion of the task.
You must submit a final, complete plan as an argument.

***** Example *****
Query: Could you create a travel plan for 7 people from Ithaca to Charlotte spanning 3
days, from March 8th to March 14th, 2022, with a budget of $30,200? The preference is for
an entire room, and we would not be taking any flights. In terms of cuisine, we are
interested in sampling some Italian and Chinese food.You can call CostEuquiry like
CostEuquiry[{{"people_number": 7,"day": 1,"current_city": "from Ithaca to Charlotte","
transportation": "Flight Number: F3633413, from Ithaca to Charlotte, Departure Time:
05:38, Arrival Time: 07:46","breakfast": "Nagaland's Kitchen, Charlotte","attraction": "
The Charlotte Museum of History, Charlotte","lunch": "Cafe Maple Street, Charlotte","
dinner": "Bombay Vada Pav, Charlotte","accommodation": "Affordable Spacious Refurbished
Room in Bushwick!, Charlotte"}}]
You can call Finish like Finish[
Day: 1
Current City: from Ithaca to Charlotte
Transportation: Flight Number: F3633413, from Ithaca to Charlotte, Departure Time: 05:38,
Arrival Time: 07:46
Breakfast: Nagaland's Kitchen, Charlotte
Attraction: The Charlotte Museum of History, Charlotte
Lunch: Cafe Maple Street, Charlotte
Dinner: Bombay Vada Pav, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 2:
Current City: Charlotte
Transportation: -
Breakfast: Olive Tree Cafe, Charlotte
Attraction: The Mint Museum, Charlotte;Romare Bearden Park, Charlotte.
Lunch: Birbal Ji Dhaba, Charlotte
Dinner: Pind Balluchi, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 3:
Current City: Charlotte
Transportation: Flight Number: F3786167, from Charlotte to Ithaca, Departure Time: 21:42,
Arrival Time: 23:26
```

```
Breakfast: Subway, Charlotte
Attraction: Books Monument, Charlotte.
Lunch: Olive Tree Cafe, Charlotte
Dinner: Kylin Skybar, Charlotte
Accommodation: -]
***** Example Ends *****

You must use Finish to indict you have finished the task. And each action only calls one
function once.
Given information: {text}
Query: {query}
```

B.3.4. Query Generation Prompt

The instruction prompt for query generation is provided as follows:

```
Given a JSON, please help me generate a natural language query. In the JSON, 'org' denotes
 the departure city. When 'days' exceeds 3, 'visiting_city_number' specifies the number of
 cities to be covered in the destination state. Here are three examples.

-----EXAMPLE 1-----
JSON:
{"org": "Gulfport", "dest": "Charlotte", "days": 3, "visiting_city_number": 1, "date":
["2022-03-05", "2022-03-06", "2022-03-07"], "people_number": 1, "constraint": {"room rule
": null, "cuisine": null, "room type": null}, "budget": 1800}
QUERY:
Please design a travel plan departing Gulfport and heading to Charlotte for 3 days,
spanning March 5th to March 7th, 2022, with a budget of $1800.
-----EXAMPLE 2-----
JSON:
{"org": "Omaha", "dest": "Colorado", "days": 5, "visiting_city_number": 2, "date":
["2022-03-14", "2022-03-15", "2022-03-16", "2022-03-17", "2022-03-18"], "people_number":
7, "constraint": {"room rule": "pets", "cuisine": null, "room type": null}, "budget":
35300}
QUERY:
Could you provide a  5-day travel itinerary for a group of 7, starting in Omaha and
exploring 2 cities in Colorado between March 14th and March 18th, 2022? Our budget is set
at $35,300, and it's essential that our accommodations be pet-friendly since we're
bringing our pets.
-----EXAMPLE 3-----
JSON:
{"org": "Indianapolis", "dest": "Georgia", "days": 7, "visiting_city_number": 3, "date":
["2022-03-01", "2022-03-02", "2022-03-03", "2022-03-04", "2022-03-05", "2022-03-06",
"2022-03-07"], "people_number": 2, "constraint": {"room rule": null, "cuisine": ["Bakery",
 "Indian"], "room type": "entire room", "transportation": "self driving"}, "budget": 6200}
QUERY:
I'm looking for a week-long travel itinerary for 2 individuals. Our journey starts in
Indianapolis, and we intend to explore 3 distinct cities in Georgia from March 1st to
March 7th, 2022. Our budget is capped at $6,200. For our accommodations, we'd prefer an
entire room. We plan to navigate our journey via self-driving. In terms of food, we're
enthusiasts of bakery items, and we'd also appreciate indulging in genuine Indian cuisine.
-----EXAMPLES END-----

JSON: {json}
QUERY:
```

B.3.5. Key Components Extraction Prompt

The instruction prompt for plan key components extraction is provided as follows:

```
Please assist me in extracting valid information from a given natural language text and
reconstructing it in JSON format, as demonstrated in the following example. Use a ';' to
separate different attractions, with each attraction formatted as 'Name, City'. If there's
```

```
 information about transportation, ensure that the 'current_city' aligns with the
destination mentioned in the transportation details (i.e., the current city should follow
the format 'from A to B'). Also, ensure that all flight numbers and costs are followed by
a colon (i.e., 'Flight Number:' and 'Cost:'), consistent with the provided example. Each
item should include ['day', 'current_city', 'transportation', 'breakfast', 'attraction', '
lunch', 'dinner', 'accommodation']. Replace non-specific information like 'eat at home/on
the road' with '-'. Additionally, delete any '$' symbols.
-----EXAMPLE-----
 [{{
        "days": 1,
        "current_city": "from Dallas to Peoria",
        "transportation": "Flight Number: 4044830, from Dallas to Peoria, Departure Time:
        13:10, Arrival Time: 15:01",
        "breakfast": "-",
        "attraction": "Peoria Historical Society, Peoria;Peoria Holocaust Memorial, Peoria
        ;",
        "lunch": "-",
        "dinner": "Tandoor Ka Zaika, Peoria",
        "accommodation": "Bushwick Music Mansion, Peoria"
    }},
    {{
        "days": 2,
        "current_city": "Peoria",
        "transportation": "-",
        "breakfast": "Tandoor Ka Zaika, Peoria",
        "attraction": "Peoria Riverfront Park, Peoria;The Peoria PlayHouse, Peoria;Glen
        Oak Park, Peoria;",
        "lunch": "Cafe Hashtag LoL, Peoria",
        "dinner": "The Curzon Room - Maidens Hotel, Peoria",
        "accommodation": "Bushwick Music Mansion, Peoria"
    }},
    {{
        "days": 3,
        "current_city": "from Peoria to Dallas",
        "transportation": "Flight Number: 4045904, from Peoria to Dallas, Departure Time:
        07:09, Arrival Time: 09:20",
        "breakfast": "-",
        "attraction": "-",
        "lunch": "-",
        "dinner": "-",
        "accommodation": "-"
    }}]
-----EXAMPLE ENDS-----
Text: {text}
JSON:
```

C. Case Presentation

C.1. Example of Query and Reference Plan

we present an example of a query and its corresponding reference plan in our train set as follows:

```
{
    "org": "Indianapolis",
    "dest": "Colorado",
    "days": 7,
    "visiting_city_number": 3,
    "date": [
        "2022-03-11",
        "2022-03-12",
        "2022-03-13",
        "2022-03-14",
        "2022-03-15",
        "2022-03-16",
```

```
        "2022-03-17"
    ],
    "people_number": 5,
    "room rule": "pets",
    "cuisine": [
        "Mexican",
        "Italian",
        "Mediterranean",
        "Indian"
    ],
    "room type": "entire room",
    "transportation": null,
    "budget": 15100,
    "query": "Can you help with generating a 7-day travel plan for a party of 5? We're
    setting off from Indianapolis and planning to explore 3 cities in Colorado from March
    11th to March 17th, 2022. We have a budget of $15,100 for this trip. We'll be bringing
     our pets, so pet-friendly accommodations are a must. We're also hoping to find places
     that offer Mexican, Italian, Mediterranean, and Indian cuisines. Entire rooms for
    accommodations would be ideal.",
    "level": "hard",
    "annotated plan": [
        {
            "days": 1,
            "current_city": "from Indianapolis to Grand Junction(Colorado)",
            "transportation": "Self-driving, from Indianapolis to Grand Junction(Colorado)
            , duration: 19 hours 21 mins, distance: 2,132 km, cost: 106",
            "breakfast": "-",
            "attraction": "-",
            "lunch": "-",
            "dinner": "Nukkadwala, Grand Junction(Colorado)",
            "accommodation": "Lovely 1 BD on the Upper West Side, Grand Junction(Colorado)
            "
        },
        {
            "days": 2,
            "current_city": "Grand Junction(Colorado)",
            "transportation": "-",
            "breakfast": "Om Ji Bhature Wale, Grand Junction(Colorado)",
            "attraction": "Museum of the West, Museums of Western Colorado, Grand Junction
            (Colorado);Eureka! McConnell Science Museum, Grand Junction(Colorado);",
            "lunch": "Penta Cafe, Grand Junction(Colorado)",
            "dinner": "Kings Kulfi, Grand Junction(Colorado)",
            "accommodation": "Lovely 1 BD on the Upper West Side, Grand Junction(Colorado)
            "
        },
        {
            "days": 3,
            "current_city": "from Grand Junction(Colorado) to Alamosa(Colorado)",
            "transportation": "Self-driving, from Grand Junction(Colorado) to Alamosa(
            Colorado), duration: 4 hours 37 mins, distance: 397 km, cost: 19",
            "breakfast": "Punjab Da Pind, Grand Junction(Colorado)",
            "attraction": "Alamosa Colorado Welcome Center, Alamosa(Colorado);Toivo Malm
            Trail System, Alamosa(Colorado);",
            "lunch": "Emperor's Lounge - The Taj Mahal Hotel, Alamosa(Colorado)",
            "dinner": "Cafe Dalal Street, Alamosa(Colorado)",
            "accommodation": "Sunny Chelsea Studio, Alamosa(Colorado)"
        },
        {
            "days": 4,
            "current_city": "Alamosa(Colorado)",
            "transportation": "-",
            "breakfast": "Good Luck Cafe, Alamosa(Colorado)",
            "attraction": "Alamosa Archery Range, Alamosa(Colorado);Alamosa Riparian Park,
             Alamosa(Colorado);Alamosa Sub, Alamosa(Colorado);",
            "lunch": "Shri Durga Dosa Corner, Alamosa(Colorado)",
```

```
            "dinner": "Lahore, Alamosa(Colorado)",
            "accommodation": "Sunny Chelsea Studio, Alamosa(Colorado)"
        },
        {
            "days": 5,
            "current_city": "from Alamosa(Colorado) to Denver(Colorado)",
            "transportation": "Self-driving, from Alamosa(Colorado) to Denver(Colorado),
            duration: 3 hours 38 mins, distance: 377 km, cost: 18",
            "breakfast": "Hamburg To Hyderabad, Alamosa(Colorado)",
            "attraction": "Denver Zoo, Denver(Colorado);",
            "lunch": "The Fatty Bao - Asian Gastro Bar, Denver(Colorado)",
            "dinner": "Woods Spice, Denver(Colorado)",
            "accommodation": "Peaceful, beautiful home away , Denver(Colorado)"
        },
        {
            "days": 6,
            "current_city": "Denver(Colorado)",
            "transportation": "-",
            "breakfast": "The Urban Socialite, Denver(Colorado)",
            "attraction": "Denver Art Museum, Denver(Colorado);Denver Museum of Nature &
            Science, Denver(Colorado);",
            "lunch": "Breaktym, Denver(Colorado)",
            "dinner": "Chawla's\u5b8a, Denver(Colorado)",
            "accommodation": "Peaceful, beautiful home away , Denver(Colorado)"
        },
        {
            "days": 7,
            "current_city": "from Denver(Colorado) to Indianapolis",
            "transportation": "Self-driving, from Denver(Colorado) to Indianapolis,
            duration: 15 hours 37 mins, distance: 1,747 km, cost: 87",
            "breakfast": "Starve Stalkers, Denver(Colorado)",
            "attraction": "-",
            "lunch": "-",
            "dinner": "-",
            "accommodation": "-"
        }
    ]
}
```


C.2. Additional Case Study

Query: Please draw up a 3-day travel itinerary for one person, beginning in Oakland and heading to Tucson from March 15th to March 17th, 2022, with a budget of $1,400.

Plan:
{
"day": 3,
"current_city": "Tucson",
"transportation": "-",
"breakfast": "Bakers Oven, Tucson",
"attraction": "San Xavier del Bac Mission, Tucson; Tucson Museum Of Art, Tucson",
"lunch": "Pizza Street, Tucson",
"dinner": "-",
"accommodation": "-"
}

Analysis:
In this instance, the planned trip did not form a closed-loop trip, ending in Tucson on the third day. Additionally, despite the trip including a stay in Tucson, the language agent failed to schedule dinner or accommodation for that day.

Figure C.1. Case 4 (Two-stage Mode).

Query: Can you create a 5-day travel plan for 2 people departing from Syracuse to visit 2 cities in Georgia? We are planning to travel from March 16th to March 20th, 2022. Our budget is approximately $2,000. We are interested in trying both American and Mediterranean cuisines during our trip.

Trajectory:
...
Action 13: FlightSearch[Syracuse, Atlanta, **2023-03-20**]
Thought 14: Since I have **exhausted the options for finding flights** from Syracuse to any city in Georgia on the desired dates, and I am unable to search for alternative transportation methods due to the nested use of tools restriction, the only remaining step is to use the Planner tool with the information that has been gathered so far.
Action 14: Planner[Create a 5-day travel plan from Syracuse to Georgia]

Plan:
Day 1:
transportation: **Flight Number: F1234567, from Syracuse toAtlanta**
...

Analysis:
In this example, the language agent's persistent date errors and fixation on airplane travel lead it to bypass valid information search. Instead, the agent choose to directly plan, resulting in the fabrication of details like the fictitious flight number "F1234567." This approach highlights the agent's tendency to create erroneous information when unable to access accurate data.

Figure C.2. Case 5 (Two-stage mode).

Query: Please curate a 3-day travel plan for a solo traveler from Tulsa to Houston from March 23rd to March 25th, 2022, with a total travel budget of $1,000.

Plan:
{
"day": 3,
"current_city": "from Houston to Tulsa",
"transportation": "Flight Number: F4013298, from Houston to Tulsa, **Departure Time: 08:20, Arrival Time: 09:43**",
"breakfast": "Earthen Spices, Houston",
"attraction": "The Museum of Fine Arts, Houston; Hermann Park, Houston;",
"lunch": "Chawla, Houston",
"dinner": "-",
"accommodation": "-"
}

Analysis:
In this example, the language agent scheduled an 8:20 AM flight departure from Houston but still planned a subsequent trip and lunch in Houston. This plan is illogical, as it overlooks the fact that the agent would no longer be in the city post-flight.

Figure C.3. Case 6 (Two-stage mode).

Query: Could you help develop a week-long travel itinerary suitable for a group of 6 people, departing from Baton Rouge and planning to visit 3 different cities in Texas? The travel dates are set from March 17th to March 23rd, 2022. Our travel budget has been adjusted to $14,600. Bearing in mind that we have children under ten years old, our accommodations need to allow young children and we prefer to occupy entire rooms. We also prefer not to self-drive during this trip.

Plan:
{
"days": 1,"
current_city": "from Baton Rouge to Texarkana",
"breakfast": "-",
"lunch": "Big City Bread Cafe, Texarkana",
"dinner": "Poets Cafe, Texarkana",
},
{
days": 2,
"current_city": "Texarkana",
"transportation": "-",
"breakfast": "Big City Bread Cafe, Texarkana",
...}

Analysis:
In this scenario, the language agents select the same restaurant for both lunch on the first day and breakfast on the second day, a choice that seems to defy commonsense.

Figure C.4. Case 7 (Direct strategy in sole-planning mode).

Query: Can you help put together a 3-day travel plan for a group of 3, leaving from Daytona Beach and heading to Atlanta from March 2nd to March 4th, 2022? We have a budget of $2,100. We require accommodations that **allow children under 10 years of age,** and we prefer having entire rooms to ourselves. Please note, we cannot utilize flights for transportation on this trip.

Plan:

```
{
"day": 1,
"current_city": "from Daytona Beach to Atlanta",
...
"accommodation": "Bright, Modern, Clean, Spacious, Brooklyn Home, Atlanta"
},
{
"day": 2,
...
"accommodation": "Bright, Modern, Clean, Spacious, Brooklyn Home, Atlanta"
}
```

Analysis:
In this example, the language agent makes an error in their accommodation planning. Agent selects a property that prohibits children under 10 and requires a minimum stay of three nights. However, the agents only plan a two-day stay, leading to a violation of the property's minimum stay requirement.

Figure C.5. Case 8 (Direct strategy in sole-planning mode).

Query: Please create a travel plan for a 3-day trip from Missoula to Dallas scheduled from March 23rd to March 25th, 2022. The budget for this trip is set at $1,900.

Plan:

```
[
        {
           "day": 1,
           "current_city": "from Missoula to Dallas",
"transportation": "Flight Number: F3604254, from Missoula to Dallas,
Departure Time: 14:27, Arrival Time: 18:26",
           "breakfast": "-",
           "attraction": "-",
           "lunch": "-",
           "dinner": "Coconuts Fish Cafe, Dallas",
           "accommodation": "1BR, elevator, kitchen, doorman!, Dallas"
        },
        {
           "day": 2,
           "current_city": "Dallas",
           "transportation": "-",
           "breakfast": "Cafe Gatherings, Dallas",
           "attraction": "The Dallas World Aquarium, Dallas;
The Sixth Floor Museum at Dealey Plaza, Dallas;",
           "lunch": "1918 Bistro & Grill, Dallas",
           "dinner": "Yanki Sizzlers, Dallas",
           "accommodation": "1BR, elevator, kitchen, doorman!, Dallas"
        },
        {
           "day": 3,
           "current_city": "from Dallas to Missoula",
           "transportation": "Flight Number: F3604227,
from Dallas to Missoula, Departure Time: 11:28, Arrival Time: 13:48",
           "breakfast": "MONKS, Dallas",
           "attraction": "Reunion Tower, Dallas",
           "lunch": "-",
           "dinner": "-",
           "accommodation": "-"
        }
     ]
```

Analysis:
A feasible plan that meets all constraints.

Figure C.6. Case 9 (Direct strategy in sole-planning mode).

Query: Please assist in crafting a travel plan for a solo traveller, journeying from Detroit to San Diego for 3 days, from March 5th to March 7th, 2022. The travel plan should accommodate a total budget of **$3,000**.

Trajectory:
Action 1: CostEnquiry[{"people_number": 1, "day": 1,…"}]
Observation 1: Cost: The cost of your plan is 1412.0 dollars.

Thought 2: The cost for the first day is $1412, which includes the flight and activities in San Diego. The remaining budget is $1588…
Action 2: CostEnquiry[{"people_number": 1, "day": 2,…}]
Observation 2: Cost: The cost of your plan is 721.0 dollars.

Thought 3: The cost for the second day is $721, which includes meals, attractions, and the same accommodation as the first day. The total spent so far is $2133, leaving a remaining budget of $867.
Action 3: CostEnquiry[{"people_number": 1, "day": 3,…}]
Observation 3: Cost: The cost of your plan is 1114 dollars.

Thought 4: The cost for the third day is $1114, which includes the return flight and activities in San Diego before departure. **The total cost for all three days is now $3247, which exceeds the budget of $3000**. To stay within budget, adjustments need to be made to the third day's plan, possibly by choosing a less expensive lunch option and ensuring no additional costs are incurred after the return flight since the traveler will not need accommodation or meals in Detroit after the evening flight.
Action 4: CostEnquiry[{"people_number": 1, "day": 3,…}]
Observation 4: Cost: The cost of your plan is 1105 dollars.

Thought 5: The revised cost for the third day is $1105, which includes a more affordable lunch option. **The total cost for all three days is now $3238, which is still slightly over the budget of $3000.** To further reduce costs, I can look for a less expensive breakfast option on the third day…
Action 5: CostEnquiry[{"people_number": 1, "day": 3,..}]
Observation 5: Cost: The cost of your plan is 1083 dollars.

Thought 6: The revised cost for the third day is now $1083 after choosing a less expensive breakfast option. **The total cost for all three days is $3216, which is closer to the budget but still slightly over.** Since the flight and accommodation costs are fixed, the only remaining option to adjust would be the attractions or lunch on the second day. However, given the small amount by which the budget is exceeded, **it may be acceptable to slightly exceed the budget due to the variability of daily expenses and the possibility of finding deals on-site.** The traveler should be prepared for a small buffer in their budget.
Action 6: Finish

Analysis:
In this example, the language agent acknowledges that the total cost exceeds 3,000 but, after several unsuccessful modifications, settles for the existing solution. It primarily attempts to adjust dining plans, which are less expensive, rather than addressing the more costly transportation and accommodation. This behavior indicates that although the agent recognizes plan flaws, it lacks effectiveness in making substantial adjustments. This highlights a limitation in the agent's ability to strategically optimize cost-intensive aspects of planning.

Figure C.7. Case 10 (Reflexion strategy in sole-planning mode).