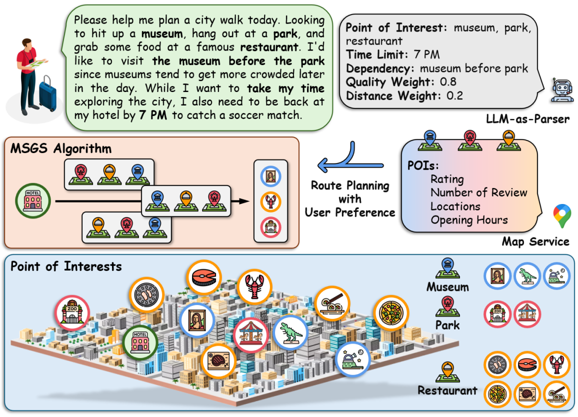
LLMAP: LLM-Assisted Multi-Objective Route Planning with User Preferences

Liangqi Yuan[♠*], **Dong-Jun Han**[♡], **Christopher G. Brinton**[♠], **Sabine Brunswicker**[◇]
[♠] School of Electrical and Computer Engineering, Purdue University, West Lafayette, USA,
[♡] Department of Computer Science and Engineering, Yonsei University, Seoul, South Korea
[◇] Polytechnic Institute, Purdue University, West Lafayette, USA
`{liangqiy,cgb,sbrunswi}@purdue.edu`
`djh@yonsei.ac.kr`

Abstract

The rise of large language models (LLMs) has made natural language-driven route planning an emerging research area that encompasses rich user objectives. Current research exhibits two distinct approaches: direct route planning using LLM-as-Agent and graph-based searching strategies. However, LLMs in the former approach struggle to handle extensive map data, while the latter shows limited capability in understanding natural language preferences. Additionally, a more critical challenge arises from the highly heterogeneous and unpredictable spatio-temporal distribution of users across the globe. In this paper, we introduce a novel LLM-Assisted route Planning (LLMAP) system that employs an LLM-as-Parser to comprehend natural language, identify tasks, and extract user preferences and recognize task dependencies, coupled with a Multi-Step Graph construction with iterative Search (MSGS) algorithm as the underlying solver for optimal route finding. Our multi-objective optimization approach adaptively tunes objective weights to maximize points of interest (POI) quality and task completion rate while minimizing route distance, subject to three key constraints: user time limits, POI opening hours, and task dependencies. We conduct extensive experiments using 1,000 routing prompts sampled with varying complexity across 14 countries and 27 cities worldwide. The results demonstrate that our approach achieves superior performance with guarantees across multiple constraints. [1]



Please help me plan a city walk today. Looking to hit up a **museum**, hang out at a **park**, and grab some food at a famous **restaurant**. I'd like to visit **the museum before the park** since museums tend to get more crowded later in the day. While I want to **take my time** exploring the city, I also need to be back at my hotel by **7 PM** to catch a soccer match.

Point of Interest: museum, park, restaurant
Time Limit: 7 PM
Dependency: museum before park
Quality Weight: 0.8
Distance Weight: 0.2

Figure 1: Overview of our LLMAP system. LLM-as-Parser processes user queries to extract key information and user preferences, which MSGS then leverages to perform route planning.

1 Introduction

The advancement in the natural language understanding capabilities of large language models (LLMs) has transformed many tasks from manual planning to automated understanding, reasoning, and decision making based on human natural language input (Song et al., 2023; Huang et al., 2024; Kambhampati et al., 2024; Yuan et al., 2025). For example, a transformative domain is autonomous driving (Sharan et al., 2023; Zeng et al., 2024), where, for safety considerations, drivers often rely on voice control for tasks such as music playback, message responses, and navigation (Feng et al., 2024). Beyond predetermined itineraries, a more common scenario involves users knowing only their daily tasks, which consist of a series of points of interest (POI) types such as grocery stores, pharmacies, and banks, without specific ordering or POI selection. This situation introduces multiple trade-offs, including the balance between POI quality and distance, while considering various constraints such as user time limits and POI opening hours.

While recent LLM literature has explored spatio-temporal comprehension and route optimization capabilities, most studies conduct only simplified evaluations in toy grid environments (Aghzal et al., 2023; Fatemi et al., 2024). These works typically

---

[*] Corresponding author.

[1] Code and data are available at `https://github.com/liangqiyuan/LLMAP`.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 7866–7894
November 4-9, 2025 ©2025 Association for Computational Linguistics

focus on a single objective, such as minimizing distance (Meng et al., 2024; Xiao and Wang, 2023), while other approaches focus solely on constraint satisfaction without considering distance or POI quality (Xie et al., 2024; Hao et al., 2024; Zhang et al., 2024). A typical use case illustrates the complexity in real-world scenario that such works fails to account for: LLM needs to process qualities (e.g., ratings) for multiple POIs and make weighted trade-offs between qualities and distances based on user preferences, while considering user time limits, POI opening hours, and task sequential dependencies (e.g., purchasing flowers at a florist before visiting a relative in the hospital). Furthermore, the context length grows linearly with the number of POIs, for example, 100 POIs can require up to 20,000 tokens. In practice, these challenges are amplified by the highly heterogeneous nature of user queries, which can originate from anywhere in the world at any time. Consequently, this combination of complex information, inherently multi-objective human expectations, and diverse user preferences poses a critical challenge: *How can LLMs perform route planning based on user preferences, especially when faced with large amounts of heterogeneous and complex POI information?*

In this paper, we introduce LLMAP, a novel system that combines an *LLM-as-Parser* with a novel multi-objective optimization algorithm for user preference-based route planning, as shown in Figure 1. This architecture addresses fundamental limitations in existing approaches that rely solely on *LLM-as-Agent*. LLMAP supports the following capabilities: (i) trade-offs between POI quality and distance according to user preferences, (ii) adaptively tunes objective weights on a per-query basis, (iii) maximization of task completion rate, and (iv) guarantee of constraints, including user time limits, POI opening hours, and task dependencies. The LLMAP system exhibits inherent scalability, extending beyond these specific settings to accommodate various use cases. Recall that the unique challenge in our scenario is that user locations and other information are not known a priori; each user query is highly heterogeneous and unpredictable, potentially originating from anywhere in the world at any time. To address these distinct challenges, we propose a novel solution that employs LLM-as-Parser to interpret human language, coupled with a multi-step graph construction with iterative search (MSGS) algorithm. Our main contributions can be summarized as follows.

- We develop the LLMAP system that performs route planning by interpreting human language and user preferences on a per-query basis. The system enables conversational interaction for real-time preference interpretation and error correction, while trading off multiple user objectives and adhering to various operational constraints throughout the planning process.
- We present the MSGS algorithm for multi-step multi-objective optimization: first ensuring adherence to constraints including user time limits, POI opening hours, and task dependencies, then prioritizing task completion rate, followed by quality-distance trade-off optimization based on user preferences.
- We conduct extensive experiments on 1,000 routing prompts across 14 countries and 27 cities with heterogeneous POI distributions, evaluating 10+ LLMs with both vanilla and chain-of-thought (CoT) prompting strategies.
- We benchmark both our LLMAP system and LLM-as-Agent solutions through comprehensive experiments, demonstrating LLMAP's consistent and substantial advantages in handling multiple routing objectives (e.g., task completion, quality, distance) and constraints (e.g., user time limit, task dependencies, opening hours), while maintaining superior runtime efficiency.

2 Related Work

2.1 LLM for Planning

Recent advances have explored the integration of LLMs into various planning domains (Dagan et al., 2023; Valmeekam et al., 2023; Wu et al., 2024b). These works leverage LLMs' natural language understanding capabilities to interpret user requirements while addressing the spatio-temporal characteristics inherent in planning tasks. In travel planning applications, LLMs are employed to design multi-day itineraries across different cities while considering multiple constraints such as budget limitations, inter-city transportation, and attraction diversity (Xie et al., 2024; Tang et al., 2024; Hao et al., 2024; Singh et al., 2024; Wu et al., 2024a; Ju et al., 2024). For more granular planning tasks, such as path planning in sandbox environments with defined start points, endpoints, and obstacles, LLMs demonstrate capability in environment comprehension and optimal path generation (Xiao and Wang, 2023; Meng et al., 2024). In the domains of

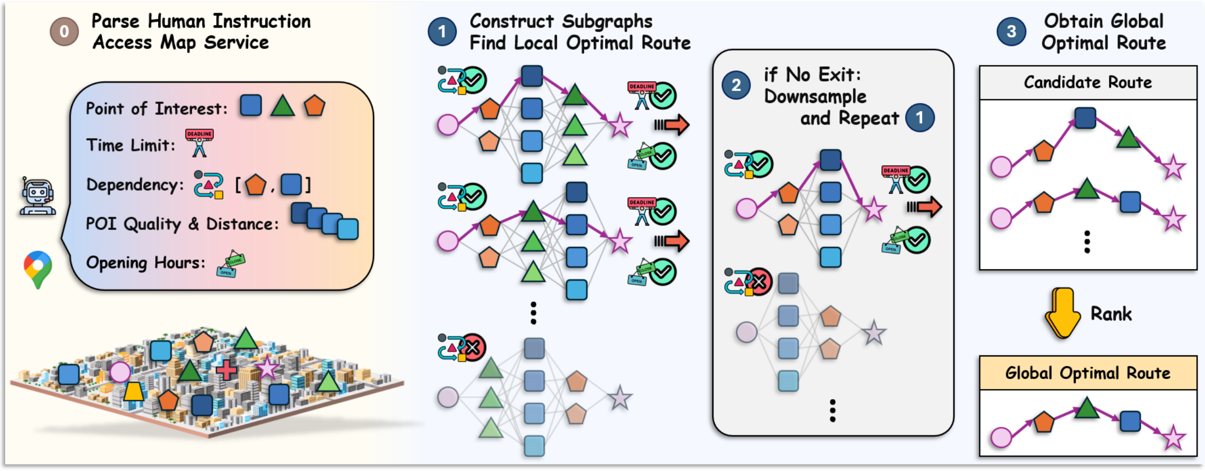


Figure 2: System architecture of LLMAP system and MSGS algorithm. After the LLM-as-Parser interprets the user query and retrieves POI information from the map service, the MSGS algorithm is used to identify the optimal route. Dependency constraints are verified before subgraph construction, while time limits and opening hours are validated after obtaining local optimal paths.

robotics and autonomous driving, planning tasks typically involve multiple obstacles in real-world environments, with visual perception playing a crucial role in the capture of environmental information (Shah et al., 2023; Sinha et al., 2024; Kannan et al., 2024; Han et al., 2024).

2.2 Route Planning Algorithm

Route planning algorithms form the basis of autonomous navigation, with traditional approaches like Dijkstra, A*, D*, and D*-Lite known for optimality and completeness (Koenig and Likhachev, 2002; Bast et al., 2016). These classical methods, especially Dijkstra's algorithm (Haeupler et al., 2024), perform well on simple graphs. However, they may face computational challenges when scaling to large-scale maps. Recent research has explored graph reinforcement learning to address these limitations (Yu and Hu, 2021; Xing et al., 2022). However, existing approaches face limitations in their reliance on large-scale training data and out-of-distribution generalization, while lacking adaptive mechanisms to optimize routing on a per-query basis and balance multiple objectives. Furthermore, with human language as the desired system input, end-to-end learning and inference becomes challenging (Chen et al., 2024), while non-end-to-end approaches suffer from information loss and inability for joint training (Jin et al., 2024; Fan et al., 2024). Given these challenges and the unique characteristics of our research problem, including highly heterogeneous graphs, limited node and edge counts (tens to hundreds), and diverse user preferences, we propose the combination of LLM-as-Parser and the MSGS algorithm as a promising solution.

3 Methodology

3.1 LLMAP System Overview

In the LLMAP system, a user interacts with the LLM to derive target POI types and constraints from a language instruction. Figure 1 shows a classic scenario, which involves a user query:

> Please help me plan a city walk today. Looking to hit up a **museum**, hang out at a **park**, and grab some food at a famous **restaurant**. I'd like to visit **the museum before the park** since museums tend to get more crowded later in the day. While I want to **take my time** exploring the city, I also need to be back at my hotel by **7 PM** to catch a soccer match.

This query contains multiple pieces of information: (i) POI types: museum, park, and restaurant, (ii) user preference: prioritize quality, (iii) task dependency: museum before park, and (iv) user time limit: before 7 PM. Based on the POI types, we search existing map services, such as Google Maps, to obtain candidate POI information, including (a) ratings, (b) number of reviews, (c) geographical locations, and (d) opening hours. Subsequently, we implement the MSGS algorithm, as shown in Figure 2, which utilizes a multi-step approach for graph construction, followed by route planning us-

Algorithm 1: POI Graph Construction

```
Input from LLM-as-Parser: POI types ($\mathcal{Y}$) and user preferences on POI quality ($\mathbf{a}$) and distance (b)
Input from Map Service: POI attributes ($\mathcal{X}$)
Output: POI Graph $\mathcal{G}$
Initialize empty node set $\mathcal{V} \leftarrow \emptyset$
for each $y \in \mathcal{Y}$ do
    Interact with map services to retrieve POI data $X_y$
    for each retrieved POI $x \in X_y$ do
        Add node $v$ with attributes $x$ to $\mathcal{V}$
Initialize empty edge set $\mathcal{E} \leftarrow \emptyset$
for each pair of nodes $v_i, v_j \in \mathcal{V}$ where $i \neq j$ do
    Calculate edge weight $\mathrm{w}_{i,j} = \mathbf{a} \cdot \mathcal{X}^{+}(v_j) - \mathrm{b} \cdot \delta(v_i, v_j)$
    Add edge $(v_i, v_j)$ with weight $\mathrm{w}_{i,j}$ to $\mathcal{E}$
Return graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X, \mathcal{Y})$
```

Algorithm 2: Multi-Step Graph Construction with Iterative Search (MSGS)

```
Input from LLM-as-Parser: POI types ($\mathcal{Y}$), user preferences on POI quality ($\mathbf{a}$) and distance (b), user time limit ($T_{\text{user}}$), and a set of task dependency requirements ($\mathcal{D}_{\text{user}}$)
Input from Map Service: POI attributes ($\mathcal{X}$)
Output: Optimal route ($\xi^*$)
Initialize full graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$                ▷ Alg. 1
Initialize optimal route $\xi^* \leftarrow \emptyset$
for each set $\mathbf{y} \subseteq \mathcal{Y}$ do
    for permutation $\vec{\mathbf{y}}$ of $\mathbf{y}$ do
        if $\vec{\mathbf{y}}$ satisfies $\mathcal{D}_{user}$ then
            Construct a subgraph $\mathcal{G}_{\vec{\mathbf{y}}}$
            Search on $\mathcal{G}_{\vec{\mathbf{y}}}$ to find its local optimal route $\widetilde{\xi^*}$
            if $\widetilde{\xi^*}$ satisfies $T_{user}$ and POI opening hours $T_{v_i},\ \forall v_i \in \widetilde{\xi^*}$ then
                $\xi^* \leftarrow \arg\max_{\xi \in \{\xi^*, \widetilde{\xi^*}\}} \mathcal{O}_{(\mathrm{ii})}(\xi)$
    if $\exists\ \xi^*$ then
        Early stop                                             ▷ Route Found
else
    $\xi^* \leftarrow$ [start point, end point]                ▷ No Route Found
```

ing search algorithms (e.g., Dijkstra) to ensure optimal solutions across multiple objectives.

3.2 Graph Construction

After acquiring POI data through map service interactions, we employ Algorithm 1 to construct a POI graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$, wherein $\mathcal{V}$ represents the set of nodes (i.e., discrete POIs), $\mathcal{E}$ denotes the set of edges (delineating potential routes between POIs), $\mathcal{X}$ encompasses the node attribute set (e.g., ratings, number of reviews, geographical locations, and opening hours), and $\mathcal{Y}$ represents node types (e.g., hospital, supermarket). Notably, node types (a.k.a. POI types) are defined based on tasks in the user query and can be further refined beyond the previously mentioned types (e.g., a user's task might specify Walmart and Sam's Club). Additionally, another scenario involves geographical types, such as when users want to visit the Hollywood area without being restricted to a specific POI. Recall the unique challenge in our scenario of constructing graphs from natural language instructions, each query results in a highly heterogeneous POI graph $\mathcal{G}$, varying not only in attributes $\mathcal{X}$ and types $\mathcal{Y}$ but also in the number of nodes $|\mathcal{V}|$ and the number of edges $|\mathcal{E}|$.

3.3 Multi-Objective and User Preferences

Users inherently have multiple objectives, and their preference priorities vary across different queries. For example, during travel, users emphasize ratings of attractions and renowned restaurants, while during busy times, they prioritize shorter routes. Let $\xi = (v_0, v_1, ..., v_N)$ denote a route in graph $\mathcal{G}$, where $v_i \in \mathcal{V}$ and $(v_i, v_{i+1}) \in \mathcal{E}$. Mathematically, we define the multi-step multi-objective optimization problem as follows:

$$\begin{aligned}
\mathcal{O}_{(\mathrm{i})} &:= \max_{\xi\in\Omega}\left(\frac{|\{\mathcal{Y}(v_i)|v_i\in\xi\}|}{|\mathcal{Y}|}\right),\\
\mathcal{O}_{(\mathrm{ii})} &:= \max_{\xi\in\Omega}\left(\sum_{i=0}^{N}\mathbf{a}\cdot\mathcal{X}^{+}(v_{i+1}) - \mathrm{b}\sum_{i=0}^{N-1}\delta(v_i,v_{i+1})\right),\\
\text{s.t.}\ & \sum_{i=0}^{N}\mathcal{X}^{T}(v_i)+\sum_{i=0}^{N-1}\delta(v_i,v_{i+1})\le T_{\text{user}},\\
& \sum_{\tau=0}^{i}\mathcal{X}^{T}(v_\tau)+\sum_{\tau=0}^{i-1}\delta(v_\tau,v_{\tau+1})\le T_{v_i},\ \ \forall v_i\in\xi,\\
& v_i\prec v_j,\quad \forall(v_i,v_j)\in\mathcal{D}_{\text{user}},
\end{aligned}\tag{1}$$

where Ω represents all possible routes in graph $\mathcal{G}$, $\delta(v_i, v_{i+1})$ denotes the travel time between consecutive POIs, and $\mathcal{X}^T(v_i)$ represents the duration spent at each POI. This multi-step multi-objective optimization: $\mathcal{O}_{(\mathrm{i})}$ maximizes task completion rate by optimizing the coverage of desired POI types, and $\mathcal{O}_{(\mathrm{ii})}$ optimizes the trade-off between POI quality and route distance. We cumulatively optimize positive attributes $\mathcal{X}^{+}$ (such as ratings and number of reviews) of each POI visited along the route, while minimizing the total distance traveled. The weight vector $\mathbf{a}$ and the scalar weight b represent the positive importance of POI attributes and the route distance, respectively, which are dynamically adjusted based on user preferences expressed in natural language. The optimization is constrained by user time limit T_{user}, opening hours for each POI T_{v_i}, and task sequential dependencies $\mathcal{D}_{\text{user}}$.

3.4 MSGS Algorithm

Next, we present our proposed MSGS algorithm, which focuses on reconstructing $\mathcal{G}$ to obtain the optimal route ξ^*. In real-world scenarios, $\mathcal{G}$ typically exhibits a fully connected topology, since humans can inherently travel between two locations. However, this would lead to search failure, as standard search algorithms (e.g., Dijkstra) would likely proceed directly to the destination, failing to maintain task completion rates. Therefore, we propose a novel MSGS algorithm to perform multi-step multi-objective optimization. To satisfy $\mathcal{O}_{(\mathrm{i})}$ task completion rate, we reorganize and reconstruct the original graph $\mathcal{G}$ into multiple directed subgraphs, then employ a search strategy (in our paper, the Dijkstra algorithm) to obtain the optimal path. Notably, due to the joint effect of multiple constraints, we traverse all combinations of $\mathcal{Y}$ and its subsets to ensure constraint satisfaction. To address $\mathcal{O}_{(\mathrm{ii})}$ trade-off between POI quality and route distance based on user preferences, we use LLM-as-Parser to interpret the user language and derive the weight vector $\mathbf{a}$ and the scalar weight b. The detailed implementation of our LLMAP system is outlined in Algorithm 2.

4 Experiments

We aim to address two key research questions in our experiments: (RQ1) *Does our LLMAP system outperform pure LLM solutions (i.e., LLM-as-Agent) in route planning tasks?* (RQ2) *How effectively can LLM-as-Parser extract accurate information from human instructions?*

4.1 Dataset Generation

Human Instruction. To evaluate the performance of LLM-as-Agent and LLM-as-Parser, we construct a synthetic dataset called Human Instructions with Preferences for route Planning (HIPP). The dataset comprises 1,000 evaluation samples, each with heterogeneous settings. An example from the HIPP dataset are provided in Appendix B.1. The HIPP dataset generation process consists of three main steps:

1. **Synthetic Label:** We randomly sample one to five POI types from a predefined taxonomy comprising shopping malls, supermarkets, pharmacies, banks, and libraries, utilizing a discrete uniform distribution. Temporal constraints are incorporated with probability 0.3 by generating stochastic time limits between 17:00 and 24:00, sampled from a discrete uniform distribution. Furthermore, task dependencies are established with probability 0.3 between POI categories via independent Bernoulli trials conducted for each consecutive POI pair, thereby modeling sequential task relationships. The user preference weights for the quality and distance of POI are generated using a discrete uniform distribution over the quantized interval {0, 0.1, 0.2, ..., 1.0}, with the constraint that their sum equals unity to ensure proper normalization. POI operational hours are derived from authentic real-world data obtained through the Google Maps API.

2. **Human Instruction (Synthetic Data):** We leverage GPT-4o (OpenAI, 2024) to generate natural instructions based on these synthetic labels through CoT prompting (Wei et al., 2022; Kojima et al., 2022). These instructions are designed to be natural and implicit, deliberately excluding explicit numerical preference weights.

3a. **LLM-as-Agent:** We employ LLMs to comprehend human instructions and process map data, including POI geographical locations, ratings, review counts, and other relevant attributes. To address the token length limitations due to numerous POIs, we sample 10 specific POIs for each POI type as input to LLMs.

3b. **LLM-as-Parser with MSGS:** We implement LLMs to interpret, identify, and estimate parameters from these human instructions, generating estimations in the same JSON format as the synthetic label for evaluation purposes.

Use Case Scenario. We utilize the Google Places API [2] to retrieve detailed POI information, including geographical locations, ratings, number of reviews, opening hours, etc. Our evaluation scenario simulates a common use case: a student traveling from an airport to a university campus while completing intermediate tasks, such as shopping at a supermarket. To ensure generalizability, we conduct experiments across 27 major cities in 14 countries. Detailed information on the selected cities and their information can be found in Appendix B.2.

4.2 Experimental Setup

Route Evaluation. For route planning, we consider the seven evaluation metrics divided into two

---

[2] `https://developers.google.com/maps/documentation/places/web-service/overview`

Method	**Model**	**Rating** (↑)	**Number of Review** (↑)	**Length** (km) (↓)	**Task Completion Rate** (%) (↑)	**Constraint Violation** (↓) **Time Limit** (hrs)	**Dependency** (%)	**Opening Hours** (%)
LLMAP w/o LLM-as-Parser		4.30	5936	29.78	96.69	0.00	0.00	0.00
LLMAP w/o MSGS		1.66	5674	42.45	17.29	0.00	41.23	1.50
SMT Solver (Hao et al., 2024)		1.22	158	34.36	13.90	0.28	2.51	11.10
SMT Solver v2		1.98	491	36.71	100.00	0.00	60.36	22.40
LLMAP	Phi-3-mini	2.19	3459	29.58	48.44	0.00	0.00	0.00
LLMAP	Phi-3-mini (CoT)	4.40	6910	29.90	96.65	0.00	0.00	0.00
LLM-as-Agent	Phi-3.5-mini	3.85	1514	249.50	92.72	12.14	56.04	96.30
LLM-as-Agent	Phi-3.5-mini (CoT)	3.85	1510	247.33	92.72	11.88	56.04	96.50
LLMAP	Phi-3.5-mini	3.69	5471	29.73	81.60	0.00	0.00	0.00
LLMAP	Phi-3.5-mini (CoT)	4.38	6616	29.59	96.47	0.00	0.00	0.00
LLM-as-Agent	LLaMA-3.2-3B	4.01	2222	59.29	50.10	0.13	9.57	34.40
LLM-as-Agent	LLaMA-3.2-3B (CoT)	4.05	2409	56.50	49.78	0.05	9.11	27.00
LLMAP	LLaMA-3.2-3B	4.00	5860	30.21	91.53	0.09	0.00	0.00
LLMAP	LLaMA-3.2-3B (CoT)	4.06	6788	30.57	88.48	0.14	0.00	0.00
LLM-as-Agent	LLaMA-3.1-8B	3.98	2988	72.98	54.92	0.53	7.29	58.90
LLM-as-Agent	LLaMA-3.1-8B (CoT)	3.89	2878	66.85	51.16	0.40	3.19	46.00
LLMAP	LLaMA-3.1-8B	3.87	5592	30.32	90.00	0.12	0.00	0.00
LLMAP	LLaMA-3.1-8B (CoT)	4.07	6449	30.46	90.24	0.26	0.00	0.00
LLM-as-Agent	Mistral-7B	4.09	3313	53.67	46.46	0.01	3.42	19.90
LLM-as-Agent	Mistral-7B (CoT)	4.12	3623	53.60	46.78	0.00	3.64	21.00
LLMAP	Mistral-7B	4.16	5819	29.85	94.58	0.14	0.00	0.00
LLMAP	Mistral-7B (CoT)	4.30	6071	29.56	96.44	0.00	0.00	0.00
LLM-as-Agent	Gemma-2-2B	3.78	1529	59.45	34.62	0.05	6.83	27.80
LLM-as-Agent	Gemma-2-2B (CoT)	3.77	1543	57.54	33.62	0.04	5.69	23.70
LLMAP	Gemma-2-2B	4.40	6909	30.34	96.03	0.00	0.00	0.00
LLMAP	Gemma-2-2B (CoT)	4.36	6552	29.68	95.46	0.00	0.00	0.00
LLM-as-Agent	Gemma-2-9B	4.09	3585	53.89	53.34	0.00	2.05	21.80
LLM-as-Agent	Gemma-2-9B (CoT)	4.10	3506	54.15	53.60	0.00	1.59	20.80
LLMAP	Gemma-2-9B	4.05	5077	29.53	93.76	0.25	0.00	0.00
LLMAP	Gemma-2-9B (CoT)	3.95	4823	29.52	93.91	0.33	0.00	0.00
LLMAP	GPT-3.5	4.23	6957	30.20	92.48	0.30	0.00	0.00
LLMAP	GPT-3.5 (CoT)	4.20	6660	30.16	91.81	0.34	0.00	0.00
LLMAP	GPT-4o-mini	4.24	6249	29.68	94.03	0.25	0.00	0.00
LLMAP	GPT-4o-mini (CoT)	4.24	6172	29.68	94.03	0.30	0.00	0.00
LLMAP	GPT-4o	4.26	5720	29.70	96.02	0.00	0.00	0.00
LLMAP	GPT-4o (CoT)	4.28	5530	29.73	96.44	0.00	0.00	0.00
LLMAP	OpenAI o1-mini	4.07	5372	29.69	93.78	0.31	0.00	0.00
LLMAP	OpenAI o1-mini (CoT)	4.09	5348	29.60	93.69	0.32	0.00	0.00
LLMAP	OpenAI o1	4.12	5193	29.54	93.53	0.34	0.00	0.00
LLMAP	OpenAI o1 (CoT)	4.11	5161	29.48	93.49	0.34	0.00	0.00

Table 1: Main results for route planning evaluation across different methods. Results use a color scheme where blue indicates superior and red indicates inferior performance, with detailed color threshold specifications in Appendix A.3. We observe that the proposed LLMAP approach offers a significant compared to the LLM-as-Agent baseline across various metrics under different model settings.

categories: soft metrics and hard constraints. Soft metrics include rating, number of reviews, path length, and task completion rate. Hard constraints consist of user time limit, task dependency requirements, and opening hours compliance. The hard constraints represent mandatory requirements that all valid routes must satisfy, while soft metrics reflect varying degrees of route quality.

LLM-as-Parser Evaluation. We evaluate the performance of different LLMs and CoT prompting using four metrics. For POI type identification and ask dependency detection, we employ the F1 score, as they effectively distinguish between missing and superfluous elements. For user time limit extraction we use accuracy, while for user preference estimation we use similarity. Details are provided in Appendix A.3. All four metrics follow a higher-is-better principle, enabling us to compute average scores across metrics.

Implementation. In our experiments, we evaluate various LLMs with their corresponding CoT prompting approaches, including Phi-3-mini (Abdin et al., 2024), Phi-3.5-mini (Abdin et al., 2024), LLaMA-3.2-3B (Dubey et al., 2024), LLaMA-3.1-8B (Dubey et al., 2024), Mistral-7B-v0.3 (Jiang et al., 2023), Gemma-2-2b (Gemma Team et al., 2024), Gemma-2-9b (Gemma Team et al., 2024), GPT-3.5-turbo (OpenAI, 2023), GPT-4o-mini (OpenAI, 2024), GPT-4o (OpenAI, 2024), OpenAI o1-mini (OpenAI, 2024), and OpenAI o1 (OpenAI, 2024). All experiments are conducted on an NVIDIA A100 GPU with 40 GB of memory.

SOTA Baselines. While numerous prompting techniques have been proposed to enhance LLMs' reasoning capabilities, such as CoT (Kojima et al., 2022), ToT (Yao et al., 2023), GoT (Besta et al., 2024), ReAct (Yao et al., 2022), and Reflexion (Shinn et al., 2024), we primarily focus on comparing CoT with our base implementation. This choice is made as our paper focuses on comparing two fundamental paradigms (LLM-as-Agent vs. LLM-as-Parser) rather than evaluating various prompting techniques. Additionally, we observe that the limitations of LLM-as-Agent approach are primarily due to the practical constraints of handling extensive POI information, resulting in increased memory usage and inference time, rather than solely from insufficient reasoning capabilities. Most existing approaches (Xie et al., 2024; Singh et al., 2024) adopt the LLM-as-Agent paradigm, which directly employs LLMs for route planning, hence we do not include specific annotations. We also consider an SMT solver-based method for evaluation (Hao et al., 2024). For simplicity, we utilize synthetic labels and implement the SMT solver to handle three constraints: time limit, dependency, and opening hours. Given that SMT solvers are not designed for multi-step multi-objective optimization, we extend the approach to SMT solver v2, which incorporates task completion rate as an additional constraint to evaluate the solver's performance in our research scenario.

4.3 Main Results

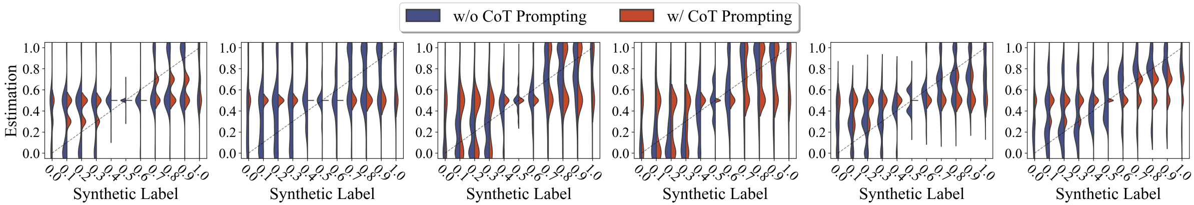
Comparison with LLM-as-Agent. Table 1 presents a comparative analysis between our LLMAP system and LLM-as-Agent approach, demonstrating that LLMAP (LLM-as-Parser + MSGS) consistently outperforms LLM-as-Agent across all metrics. We observe two distinct types of errors in LLM-as-Agent implementations. In the first case, certain LLMs (e.g., Phi-3.5-mini, Phi-3.5-mini (CoT)) fail to perform reasonable route planning, indiscriminately including POIs regardless of their feasibility. While this case appears to achieve high task completion rates, it significantly exceeds time limits and disregards dependency constraints. In the second case, while other LLMs demonstrate the ability to selectively incorporate POIs into routes, they struggle to effectively maximize task completion rates while satisfying multiple constraints. Although these LLMs show potential to maximize ratings and the number of reviews, they exhibit limitations in geographical distance reasoning. LLMAP prioritizes task completion rate and constraint satisfaction, achieving superior performance in these objectives.

LLM	POI F1 (%)	Time Lt. Acc. (%)	Dep. F1 (%)	Preference Sim. (%)
Phi-3-mini	51.20	37.42	26.80	78.58
Phi-3-mini (CoT)	100.00	90.32	99.36	81.53
Phi-3.5-mini	85.70	73.87	74.91	77.81
Phi-3.5-mini (CoT)	100.00	97.74	99.74	76.82
LLaMA-3.2-3B	98.27	79.35	95.29	82.08
LLaMA-3.2-3B (CoT)	96.49	84.52	87.17	80.19
LLaMA-3.1-8B	94.50	86.13	91.48	78.58
LLaMA-3.1-8B (CoT)	96.60	88.71	93.89	82.68
Mistral-7B	99.28	95.48	97.58	82.81
Mistral-7B (CoT)	99.98	99.68	99.22	80.89
Gemma-2-2B	99.90	89.03	98.16	81.54
Gemma-2-2B (CoT)	99.47	93.23	97.76	81.69
Gemma-2-9B	99.92	92.90	99.60	87.91
Gemma-2-9B (CoT)	99.96	90.65	99.48	84.93
GPT-3.5	100.00	86.13	99.46	77.99
GPT-3.5 (CoT)	100.00	85.81	99.53	77.57
GPT-4o-mini	99.80	90.65	99.74	86.14
GPT-4o-mini (CoT)	100.00	90.65	99.74	86.52
GPT-4o	99.90	98.06	99.78	89.35
GPT-4o (CoT)	100.00	99.03	99.74	89.74
OpenAI o1-mini	99.97	91.29	99.66	88.39
OpenAI o1-mini (CoT)	99.87	90.97	99.70	88.16
OpenAI o1	99.90	90.32	99.74	90.33
OpenAI o1 (CoT)	100.00	90.32	99.81	90.76

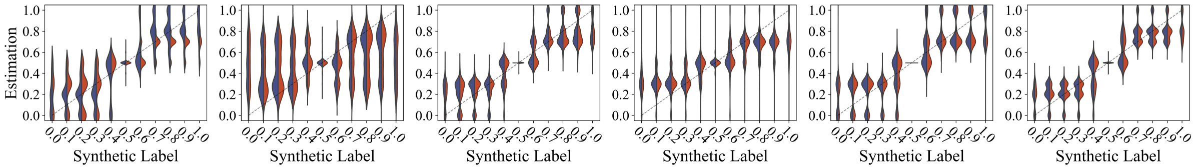
Table 2: Evaluation of various LLM-as-Parser models, with blue highlighting the highest average score.

Comparison with SOTA Baselines. In our scenario, both the LLM-as-Agent solution and the SMT solver approach (Hao et al., 2024) demonstrate significant limitations in generating reasonable routes. The SMT solver's limitations stem not only from its focus on constraints while neglecting the trade-offs between human objectives but also from the substantial complexity and volume of these constraints. The SMT solver tends to generate direct routes from start to end point while bypassing POIs, effectively avoiding constraint violations. However, such solutions clearly fail to meet user expectations. To address this limitation, SMT solver v2 incorporates task completion rate as an additional constraint. However, due to conflicting constraints, while SMT solver v2 satisfies task completion rate and time limit requirements, it inevitably violates dependency constraints to accommodate the former two.

Impact of CoT Prompting. We observe that CoT prompting provides limited benefits for the LLM-as-Agent approach while substantially enhancing LLM-as-Parser performance. This disparity arises because CoT fundamentally aims to help LLMs emulate human step-by-step reasoning processes. However, when applied to LLM-as-Agent scenar-



(a) Phi-3-mini (b) Phi-3.5-mini (c) LLaMA-3.2-3B (d) LLaMA-3.1-8B (e) Mistral-7B (f) Gemma-2-2B



(g) Gemma-2-9B (h) GPT-3.5 (i) GPT-4o (j) GPT-4o-mini (k) OpenAI o1-mini (l) OpenAI o1

Figure 3: Impact of LLM-as-Parser on user preference weight estimation. The gray dashed line represents the ideal scenario where estimations perfectly match the synthetic labels.

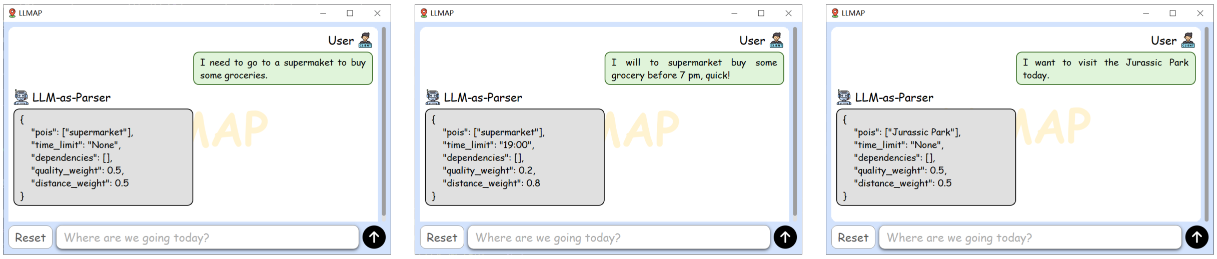
ios, neither CoT-enhanced LLMs nor humans can effectively process hundreds of POIs (in our experiments, limited to 10 POIs per type to reduce input length) along with their associated information and required computations, such as distance calculations and multi-objective trade-offs. In contrast, for LLMAP, CoT prompting effectively facilitates LLMs in emulating human reasoning patterns, leading to a more accurate interpretation of user instructions, particularly for Phi-3-mini and Phi-3.5-mini models. We provide a detailed analysis of CoT performance improvements for these models in Section 4.4.

4.4 LLM-as-Parser Evaluation

Impact of Estimation Accuracy. Table 2 further demonstrates the reliability of LLM-as-Parser. We evaluate the F1 score and accuracy using synthetic samples against LLM estimations to assess their capability in extracting POIs, routes, and preference information from natural language instructions. Our results show that LLMs excel at identifying POIs, time limits, and dependencies, as these elements primarily rely on lexical cues and logical structures that can be directly parsed from the instructions. However, the accuracy in estimating preference weights is comparatively lower, as LLMs must infer numerical values from textual descriptions. Natural language rarely conveys precise numerical information, for example, when a user states “I am in a hurry,” the inferred distance weight could range from 0.7 to 0.9.

Impact of Language-to-Numerical Preference. As illustrated in Figure 3, the distribution of estimates follows similar patterns for both low user preferences (weights ranging from 0 to 0.3) and high user preferences (weights ranging from 0.7 to 1.0). Specifically, LLM estimations remain consistent regardless of the precise synthetic label values within these ranges. This pattern persists across different LLMs and remains consistent with or without CoT prompting. For balanced user preferences (weights ranging from 0.4 to 0.6), LLM estimates cluster around 0.5. While some models demonstrate more robust estimation capabilities (e.g., Gemma-2-9B, GPT-4o, OpenAI o1-mini, OpenAI o1) with shorter distribution tails, all LLMs exhibit a stepped distribution rather than the ideal linear trend (indicated by the gray dashed line). This reveals LLM ability to approximately classify user preferences, or more broadly, sentiments, from natural language, while struggling with fine-grained numerical outputs. This suggests that further fine-tuning might be necessary to better capture individual linguistic nuances. A sample and further analysis are provided in Appendix B.1.

Impact of CoT on Parsing. While CoT prompting significantly improves the performance of LLM-as-Parser for certain models (e.g., Phi-3-mini and Phi-3.5-mini), it adversely affects others (e.g., Gemma-2-2B (CoT)). The improvements in the former case can be attributed to the prevention of format errors that often result in default parameter usage without CoT prompting. However, in the latter case, CoT prompting leads to an overestimation of POI correlations. For example, given the instruction “I want to go to the supermarket and the library today,” there is no explicit dependency (visiting the supermarket before the library is not mandatory). Nevertheless, the sequential ordering in the sentence (“supermarket” before “library”) causes the model to overestimate their interdependence.



User

I need to go to a supermaket to buy some groceries.

LLM-as-Parser

```
{
  "pois": ["supermarket"],
  "time_limit": "None",
  "dependencies": [],
  "quality_weight": 0.5,
  "distance_weight": 0.5
}
```

User

I will to supermarket buy some grocery before 7 pm, quick!

LLM-as-Parser

```
{
  "pois": ["supermarket"],
  "time_limit": "19:00",
  "dependencies": [],
  "quality_weight": 0.2,
  "distance_weight": 0.8
}
```

User

I want to visit the Jurassic Park today.

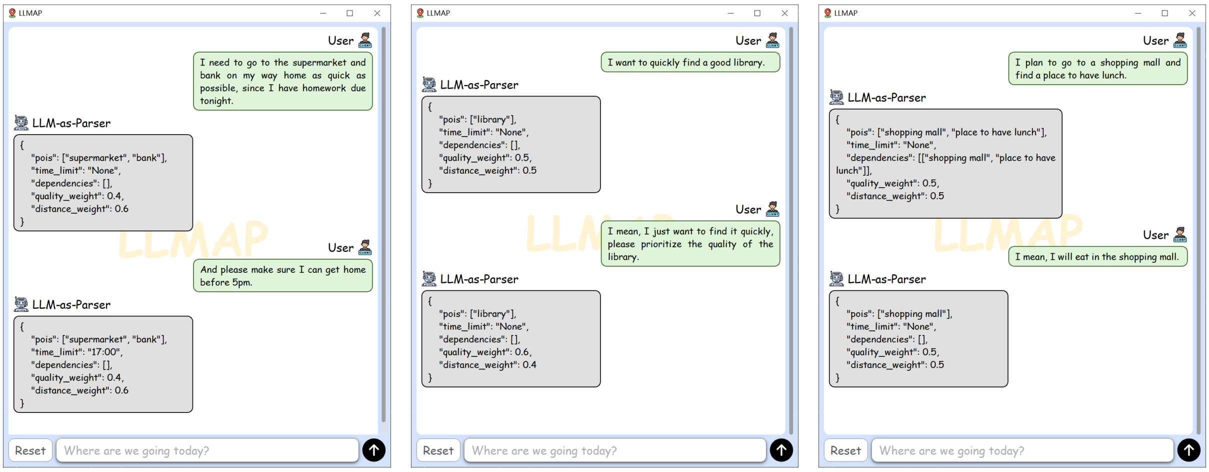
LLM-as-Parser

```
{
  "pois": ["Jurassic Park"],
  "time_limit": "None",
  "dependencies": [],
  "quality_weight": 0.5,
  "distance_weight": 0.5
}
```

(a) Typographical Error

(b) Grammar Error

(c) Non-Existent POI



User

I need to go to the supermarket and bank on my way home as quick as possible, since I have homework due tonight.

LLM-as-Parser

```
{
  "pois": ["supermarket", "bank"],
  "time_limit": "None",
  "dependencies": [],
  "quality_weight": 0.4,
  "distance_weight": 0.6
}
```

User

And please make sure I can get home before 5pm.

LLM-as-Parser

```
{
  "pois": ["supermarket", "bank"],
  "time_limit": "17:00",
  "dependencies": [],
  "quality_weight": 0.4,
  "distance_weight": 0.6
}
```

User

I want to quickly find a good library.

LLM-as-Parser

```
{
  "pois": ["library"],
  "time_limit": "None",
  "dependencies": [],
  "quality_weight": 0.5,
  "distance_weight": 0.5
}
```

User

I mean, I just want to find it quickly, please prioritize the quality of the library.

LLM-as-Parser

```
{
  "pois": ["library"],
  "time_limit": "None",
  "dependencies": [],
  "quality_weight": 0.6,
  "distance_weight": 0.4
}
```

User

I plan to go to a shopping mall and find a place to have lunch.

LLM-as-Parser

```
{
  "pois": ["shopping mall", "place to have lunch"],
  "time_limit": "None",
  "dependencies": [["shopping mall", "place to have
lunch"]],
  "quality_weight": 0.5,
  "distance_weight": 0.5
}
```

User

I mean, I will eat in the shopping mall.

LLM-as-Parser

```
{
  "pois": ["shopping mall"],
  "time_limit": "None",
  "dependencies": [],
  "quality_weight": 0.5,
  "distance_weight": 0.5
}
```

(d) Vague Instruction

(e) Ambiguity

(f) Misinterpretation

Figure 4: Demonstration of errors and conversational correction strategies. The interface facilitates interaction between real users and the LLM while maintaining contextual information to support conversational functionality.

4.5 User-to-LLM Conversational Correction

We present user interaction scenarios in Figure 4, demonstrating user engagement with the LLMAP system to identify and rectify computational reasoning errors through a graphical user interface implemented via the PyQt5 framework. Beyond the linguistic errors in Figures 4(a), 4(b), and 4(c), we present three representative LLM reasoning errors and corrective strategies. Figure 4(d) displays a user's vague instruction that fails to explicitly specify the exact time of the deadline, consequently hindering the LLM's ability to infer the time limit constraint. Figure 4(e) illustrates a case of ambiguous user instruction, where "quickly" modifies the action "find" rather than indicating that the route itself should be fast, resulting in a misinterpretation of user preference. In Figure 4(f), the LLM misconstrues the user's intention of "find a place", where the phrase does not refer to a "place to have lunch" as a POI, but rather represents one purpose for visiting the shopping mall. These scenarios demonstrate that conversational correction effectively addresses LLM comprehension errors through iterative user dialogue.

4.6 Further Analysis

We provide detailed analyses in Appendix C, including the time complexity analysis of MSGS, empirical runtime comparisons, ablation study, transportation modes, departure days and times, the impact of geographic variations, and the integration of essential waypoints.

5 Conclusion

In this paper, we present a novel LLMAP system that integrates LLM-as-Parser with the MSGS algorithm to facilitate multi-objective route planning. Our system decomposes the complex task into two components: using LLMs to understand and extract information from human instructions, and employing MSGS to generate optimal routes that satisfy both constraints and user preferences. Through extensive experiments across 27 major cities in 14 countries, we demonstrate that our approach outperforms LLM-as-Agent solutions across multiple metrics. A future direction is to integrate richer information sources, such as user text reviews, as supplementary input to LLMs to enhance preference matching.

Limitations

The primary limitation of our proposed method lies in the computational overhead of the MSGS algorithm when handling a large number of POI types, as it requires permutation of POI type sets to find the optimal solution. Human instructions typically do not encompass an extensive list of POI types, as users rarely express intentions to visit dozens of locations such as hospitals, supermarkets, banks, bookstores, museums, and theaters in a single request. Nevertheless, the time complexity of the MSGS algorithm does increase significantly with the number of POI types.

Acknowledgments

This work was supported by a doctoral fellowship offered by the Purdue-Windracers Center AIDA[3], in part by the National Science Foundation (NSF) under grants CPS-2313109 and CPS-2146171, and by the OpenAI Researcher Access Program.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2023. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *arXiv preprint arXiv:2310.03249*.

Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. 2016. Route planning in transportation networks. *Algorithm engineering: Selected results and surveys*, pages 19–80.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Gautier Dagan, Frank Keller, and Alex Lascarides. 2023. Dynamic planning with a llm. *arXiv preprint arXiv:2308.06391*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenqi Fan, Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Hui Liu, Xiaorui Liu, Dawei Yin, et al. 2024. Graph machine learning in the era of large language models (llms). *arXiv preprint arXiv:2404.14928*.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2024. Test of time: A benchmark for evaluating llms on temporal reasoning. *arXiv preprint arXiv:2406.09170*.

Jie Feng, Jun Zhang, Junbo Yan, Xin Zhang, Tianjian Ouyang, Tianhui Liu, Yuwei Du, Siqi Guo, and Yong Li. 2024. Citybench: Evaluating the capabilities of large language model as world model. *arXiv preprint arXiv:2406.13945*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Bernhard Haeupler, Richard Hladík, Václav Rozhoň, Robert E Tarjan, and Jakub Tětek. 2024. Universal optimality of dijkstra via beyond-worst-case heaps. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2099–2130. IEEE.

Dongge Han, Trevor McInroe, Adam Jelley, Stefano V Albrecht, Peter Bell, and Amos Storkey. 2024. Llm-personalize: Aligning llm planners with human preferences via reinforced self-training for housekeeping robots. *arXiv preprint arXiv:2404.14285*.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. 2024. Large language models can plan your travels rigorously with formal verification tools. *arXiv preprint arXiv:2404.11891*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*.

Da Ju, Song Jiang, Andrew Cohen, Aaron Foss, Sasha Mitts, Arman Zharmagambetov, Brandon Amos, Xian Li, Justine T Kao, Maryam Fazel-Zarandi, et al. 2024. To the globe (ttg): Towards language-driven guaranteed travel planning. *arXiv preprint arXiv:2410.16456*.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*.

Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. 2024. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12140–12147. IEEE.

Sven Koenig and Maxim Likhachev. 2002. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, volume 1, pages 968–975. IEEE.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Silin Meng, Yiwei Wang, Cheng-Fu Yang, Nanyun Peng, and Kai-Wei Chang. 2024. Llm-a*: Large language model enhanced incremental heuristic search on path planning. *arXiv preprint arXiv:2407.02511*.

OpenAI. 2023. Introducing chatgpt. `https://openai.com/blog/chatgpt`.

OpenAI. 2024. Gpt-4o mini: advancing cost-efficient intelligence. `https://openai.com/index/gpt-4o-mini/`.

OpenAI. 2024. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`.

OpenAI. 2024. Openai o1 system card. `https://openai.com/index/openai-o1-system-card/`.

Dhruv Shah, Błażej Osiński, Sergey Levine, et al. 2023. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR.

SP Sharan, Francesco Pittaluga, Manmohan Chandraker, et al. 2023. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Harmanpreet Singh, Nikhil Verma, Yixiao Wang, Manasa Bharadwaj, Homa Fashandi, Kevin Ferreira, and Chul Lee. 2024. Personal large language model agents: A case study on tailored travel planning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 486–514.

Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. 2024. Real-time anomaly detection and reactive planning with large language models. *arXiv preprint arXiv:2407.08735*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Yihong Tang, Zhaokai Wang, Ao Qu, Yihao Yan, Kebing Hou, Dingyi Zhuang, Xiaotong Guo, Jinhua Zhao, Zhan Zhao, and Wei Ma. 2024. Synergizing spatial optimization with large language models for open-domain urban itinerary planning. *arXiv preprint arXiv:2402.07204*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qinzhuo Wu, Wei Liu, Jian Luan, and Bin Wang. 2024a. Toolplanner: A tool augmented llm for multi granularity instructions with path planning and feedback. *arXiv preprint arXiv:2409.14826*.

Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, et al. 2024b. Can graph learning improve planning in llm-based agents? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Hengjia Xiao and Peng Wang. 2023. Llm a*: Human in the loop large language models enabled a* search for robotics. *arXiv preprint arXiv:2312.01797*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Qiang Xing, Yan Xu, Zhong Chen, Ziqi Zhang, and Zhao Shi. 2022. A graph reinforcement learning-based decision-making platform for real-time charging navigation of urban electric vehicles. *IEEE Transactions on Industrial Informatics*, 19(3):3284–3295.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zishun Yu and Mengqi Hu. 2021. Deep reinforcement learning with graph representation for vehicle repositioning. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):13094–13107.

Liangqi Yuan, Dong-Jun Han, Shiqiang Wang, and Christopher G Brinton. 2025. Local-cloud inference offloading for llms in multi-modal, multi-task, multi-dialogue settings. *arXiv preprint arXiv:2502.11007*.

Qingbin Zeng, Qinglong Yang, Shunan Dong, Heming Du, Liang Zheng, Fengli Xu, and Yong Li. 2024. Perceive, reflect, and plan: Designing llm agent for goal-directed city navigation without instructions. *arXiv preprint arXiv:2408.04168*.

Cong Zhang, Derrick Goh Xin Deik, Dexun Li, Hao Zhang, and Yong Liu. 2024. Meta-task planning for language agents. *arXiv preprint arXiv:2405.16510*.

A Details of Experimental Setup

A.1 Default Settings

LLMs exhibit potential reasoning errors, particularly in output formatting, which may precipitate system failures such as improper JSON format generation or missing dictionary keys that trigger subsequent errors. We therefore establish default formatting protocols to rectify any LLM output anomalies. Figure 5 illustrates this phenomenon wherein Phi-3-mini fails to execute parsing behavior in accordance with user instructions during inference, instead generating hallucinated outputs containing inappropriate keys and semantically vacuous information. Under these circumstances, we implement error correction through default parametrization to ensure successful system execution.

Original Output (Erroneous):

```
{
    "pois": ["poi1", "poi2", ...],
    "time_constraint": "HH:00",
    "dependencies": [["poi1", "poi2
       "], ...],
    "rating_weight": float,
    "route_weight": float,
    "weather_constraint": "
       weather_condition",
    "special_events": ["event1", "
       event2", ...],
    "traffic_condition": "light/
       moderate/heavy",
    "vehicle_type": "car/bike/
       public_transport"
}
```

Error Correction (Default Setting):

```
{
    "pois": [],
    "time_limit": "None",
    "dependencies": [],
    "quality_weight": 0.5,
    "distance_weight": 0.5
}
```

Figure 5: An example of Phi-3-mini generating erroneous output. The model fails to adhere to expected schema, producing irrelevant keys and placeholders instead of proper values.

A.2 Time Parameters Setup

We configure the default route departure time as Monday 10:00 AM in all experiments. The visit duration parameters are systematically assigned for different POI categories: 120 minutes for shopping malls, 30 minutes for supermarkets, 15 minutes for pharmacies, 20 minutes for banks, and 60 minutes for libraries. These duration values are incorporated into the total time calculation whenever the corresponding POIs are included in the generated routes.

This experimental configuration is designed to be flexible and extensible for parameter adjustment, serving primarily as a mechanism to evaluate route feasibility with respect to user time limits and POI operating hours. In practical applications, the system can be extended to incorporate real-time data by leveraging smartphone timestamps and geolocation services. Furthermore, the visit duration estimates can be refined by integrating Google Maps' real-time occupancy data for more accurate predictions based on venue congestion levels.

A.3 Evaluation Metrics

Evaluation Metrics Formulation. We evaluate the performance of our LLM-as-Parser approach across multiple dimensions: POI type identification, user time limit extraction, task dependency detection, and user preference weight estimation. For the set-based evaluations, we compute the F1 score by comparing the predictions against ground truth labels. Formally, for each instance i, we denote the ground truth set as $\mathbf{y}_i$ and the predicted set as $\hat{\mathbf{y}}_i$. The precision, recall, and F1 score are defined as:

$$\begin{aligned} \text{Precision} &= \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{\mathbf{y}}_i \cap \mathbf{y}_i|}{|\hat{\mathbf{y}}_i|}, \\ \text{Recall} &= \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{\mathbf{y}}_i \cap \mathbf{y}_i|}{|\mathbf{y}_i|}, \\ F_1 &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \end{aligned} \tag{2}$$

where N denotes the number of samples. Furthermore, to assess the capability of our LLM-as-Parser in extracting user time limits and user preference weights, we employ two different evaluation metrics. For user time limit extraction, we define an accuracy metric. In this context, only two outcomes are possible: either the label and the estimated value are identical, yielding a score of 1, or they differ, yielding a score of 0. For each instance i, let y_i denote the ground truth value and $\hat{y}_i$ the estimated value. This is formalized via the indicator function:

$$\begin{aligned} \text{Accuracy} &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{y_i = \hat{y}_i\}, \\ \mathbb{I}\{y_i = \hat{y}_i\} &= \begin{cases} 1, & \text{if } y_i = \hat{y}_i, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

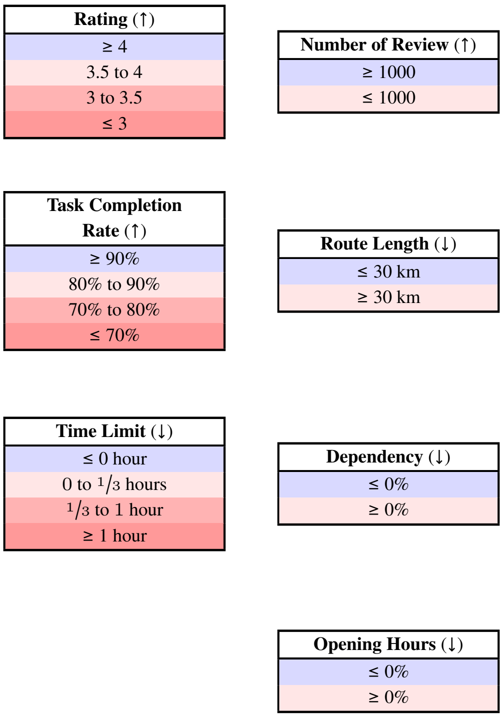
For user preference weights, we define a similarity metric to measure how close the estimated weight distribution is to the ground truth. Let $\mathbf{w}_i = (w_i^1, w_i^2, \ldots, w_i^d)$ denote the ground truth weight vector for instance i, and $\hat{\mathbf{w}}_i$ the estimated weight vector. The similarity is defined as:

$$\text{Similarity} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{1}{d}\sum_{j=1}^{d}\left|w_i^j - \hat{w}_i^j\right|\right), \tag{4}$$

where d is the number of weight components, indexed by j (e.g., rating weight and route weight). This metric yields a value in $[0, 1]$, with 1 indicating identical distributions and 0 indicating maximal difference.

Table 3 Visualization. To facilitate better comparison of different methods across multiple objectives, we establish distinct thresholds for each metric as shown in Table 3, enhancing the visualization of results in Table 1. These thresholds are determined based on empirical observations and result distribution, though they can be adjusted according to user requirements in different scenarios. For ratings, task completion rate, and user time limit, we implement multiple threshold levels to enable more nuanced distinctions. In contrast, we employ binary color coding for number of reviews, route length, dependency, and opening hours metrics. Through this color-coded visualization scheme, we aim to identify methods that satisfy all thresholds, indicated by blue coloring across all metrics.

Table 3: Color threshold settings used in Table 1. This color-coding scheme is purely for visualization purposes and does not affect any numerical results.



Rating (↑)
≥ 4
3.5 to 4
3 to 3.5
≤ 3

Number of Review (↑)
≥ 1000
≤ 1000

Task Completion Rate (↑)
≥ 90%
80% to 90%
70% to 80%
≤ 70%

Route Length (↓)
≤ 30 km
≥ 30 km

Time Limit (↓)
≤ 0 hour
0 to 1/3 hours
1/3 to 1 hour
≥ 1 hour

Dependency (↓)
≤ 0%
≥ 0%

Opening Hours (↓)
≤ 0%
≥ 0%

B Details of Dataset

B.1 HIPP Dataset

The HIPP dataset comprises 1,000 evaluation samples, each containing a unique synthetic label, human instruction, and 24 LLM-as-Parser estimations. Figure 6 illustrates a representative example from the dataset. Our analysis reveals that most LLMs successfully perform their basic functions (i.e., extracting POI types and constraints), with the exception of Phi-3-mini and Phi-3.5-mini. These exceptions stem from their inability to generate information in the correct format (i.e., JSON format). To ensure that route planning does not fail, we implement default fallback settings for these models, as demonstrated in Figure 6. Under these fallback settings, the route is simplified to direct travel between the origin and destination without any intermediate POI visits. We observe that CoT prompting enhances the ability of Phi-3-mini and Phi-3.5-mini to output correctly formatted responses. While most models accurately extract POI types, user time limits, and dependency constraints, they exhibit uncertainty in identifying user preference weights, as illustrated in Figure 3.

Synthetic Label:

```
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.9,
    "distance_weight": 0.1
}
```

Human Instruction (Synthetic Data):
Today, let's plan to visit the bank, library, supermarket, and shopping mall. Please be home by 19:00. Prioritize visiting POIs with high ratings as they are more important today. Start at the bank before heading to the library, and continue to the supermarket right after the library.

LLM-as-Parser (Estimation):

```
"Phi-3-mini":
{
    "pois": [],
    "time_limit": "None",
    "dependencies": [],
    "quality_weight": 0.5,
    "distance_weight": 0.5
}
```

```
"Phi-3.5-mini":
{
    "pois": [],
    "time_limit": "None",
    "dependencies": [],
    "quality_weight": 0.5,
    "distance_weight": 0.5
}
```

```
"LLaMA-3.2-3B":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"LLaMA-3.1-8B":
{
   "pois": ["bank", "library", "supermarket", "shopping mall"],
   "time_limit": "19:00",
   "dependencies": [["bank", "library"], ["library", "supermarket"]],
   "quality_weight": 0.7,
   "distance_weight": 0.3
}
```

```
"Mistral-7B-v0.3":
{
   "pois": ["bank", "library", "supermarket", "shopping mall"],
   "time_limit": "19:00",
   "dependencies": [["bank", "library"], ["library", "supermarket"]],
   "quality_weight": 0.7,
   "distance_weight": 0.3
}
```

```
"Gemma-2-2B":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.8,
    "distance_weight": 0.2
}
```

```
"Gemma-2-9B":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.8,
    "distance_weight": 0.2
}
```

```
"GPT-3.5":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"GPT-4o-mini":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.75,
    "distance_weight": 0.25
}
```

```
"GPT-4o":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"OpenAI o1-mini":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"OpenAI o1":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"Phi-3-mini (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"Phi-3.5-mini (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.5,
    "distance_weight": 0.5
}
```

```
"LLaMA-3.2-3B (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 1.0,
    "distance_weight": 1.0
}
```

```
"LLaMA-3.1-8B (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 1.0,
    "distance_weight": 0.0
}
```

```
"Mistral-7B-v0.3 (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"Gemma-2-2B (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"Gemma-2-9B (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"GPT-3.5 (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"GPT-4o (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"GPT-4o-mini (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"OpenAI o1-mini (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

```
"OpenAI o1 (CoT)":
{
    "pois": ["bank", "library", "supermarket", "shopping mall"],
    "time_limit": "19:00",
    "dependencies": [["bank", "library"], ["library", "supermarket"]],
    "quality_weight": 0.7,
    "distance_weight": 0.3
}
```

Figure 6: A representative example illustrating the relationship between synthetic labels, human instructions, and LLM-as-Parser estimations.

B.2 Map Service and Scenario Setting

Map Service. For POI data acquisition, we utilize the Google Places API to retrieve place (i.e., POI) information. The API accepts coordinates as input and returns comprehensive POI data within a specified search radius for given place types (i.e., POI types). The returned information includes POI IDs, names, ratings, numbers of ratings, geographical coordinates, etc. We set the end point as the search center with a radius of 5000 meters. Since the Google Places API limits each query (referred to as a "page") to 20 POI entries, we iterate through all available pages to collect complete POI information for the specified area. Notably, our system is not limited to any specific map service API, it can be readily adapted to work with various mapping services (e.g., aviation maps, hiking maps) to support diverse route planning tasks.

Incomplete POI Data. The real-world data used in this paper, sourced from Google Places API, while not producing formatically inaccurate data (e.g., ratings outside the 1-5 range) and typically possessing geographical location data, presents instances of incomplete information, particularly in regions where Google Maps experiences lower popularity. Consequently, we implement default configuration settings to mitigate system operational failures attributable to incomplete data. Specifically, we configure any incomplete data as follows: ratings as 1.0 (i.e., the minimum value), number of reviews as 1 (i.e., the minimum value), and opening hours as standard business hours (9:00 AM to 5:00 PM). An extreme case occurs when all POIs within a region lack any data, resulting in route planning that relies entirely on distance without considering other user preferences. These configurations aim to minimize potential risks in route planning, and researchers can readily modify these default settings based on empirical data.

Use Case Scenario. To demonstrate our LLMAP system's location-independent applicability and global generalizability, we conduct extensive testing across 27 major cities in 14 countries worldwide, as shown in Table 4. We focus on a common use scenario: students returning to university from airports with intermediate POI stops for various activities. This scenario serves as a representative example that can be readily extended to other situations, such as airport-to-hotel transfers, commutes from office to home, or travel between home and school. Furthermore, our proposed algorithm can be effectively deployed in scenarios where the start and end points share the same location, such as round trips from home to grocery stores. Such cases are handled by treating identical locations as distinct nodes with the same coordinates.

Extended Transportation Scenarios. Our system's versatility extends beyond specific transportation modes, encompassing various mobility scenarios including walking (e.g., city walk, hiking trails), public transit (e.g., urban transit systems, aviation networks), and logistics delivery. A key advantage of our approach is its training-free nature, requiring only parameter adjustments to adapt to different use cases. For example, consider travel planning where users need to coordinate flights between multiple destination cities. In this context, the system can optimize multiple objectives such as minimizing flight duration and costs while maximizing flight quality (e.g., avoiding budget airlines), subject to constraints including total trip duration, city visit sequence, and flight availability. Given access to flight service data, our LLMAP system and MSGS algorithm can reliably generate optimal routes while satisfying all specified constraints.

Country	City	Start Point			End Point		
		Name	Latitude	Longitude	Name	Latitude	Longitude
USA	New York	John F. Kennedy International Airport	40.644624	-73.779703	Columbia University	40.807536	-73.962573
USA	New York	John F. Kennedy International Airport	40.644624	-73.779703	New York University	40.712775	-74.005973
USA	Los Angeles	Los Angeles International Airport	33.942153	-118.403605	University of California, Los Angeles	34.069918	-118.443849
USA	Chicago	O'Hare International Airport	41.980259	-87.908986	University of Chicago	41.790448	-87.600395
USA	Chicago	O'Hare International Airport	41.980259	-87.908986	Northwestern University	42.056459	-87.675267
USA	San Francisco	San Francisco International Airport	37.619115	-122.381627	University of California, Berkeley	37.871214	-122.255463
USA	San Jose	San Jose International Airport	37.363529	-121.928593	Stanford University	37.427660	-122.170060
USA	Seattle	Seattle-Tacoma International Airport	47.448365	-122.308593	University of Washington	47.656717	-122.306618
USA	Boston	Logan International Airport	42.365602	-71.009614	Harvard University	42.374437	-71.118249
USA	Boston	Logan International Airport	42.365602	-71.009614	Massachusetts Institute of Technology	42.360091	-71.094160
Canada	Toronto	Toronto Pearson International Airport	43.679834	-79.628383	University of Toronto	43.661541	-79.400875
Canada	Vancouver	Vancouver International Airport	49.193374	-123.175128	University of British Columbia	49.260605	-123.245994
China	Beijing	Beijing Capital International Airport	40.079857	116.603112	Peking University	39.986913	116.305874
China	Beijing	Beijing Capital International Airport	40.079857	116.603112	Tsinghua University	40.006158	116.318407
China	Shanghai	Shanghai Pudong International Airport	31.144344	121.808273	Fudan University	31.297420	121.503618
China	Shanghai	Shanghai Pudong International Airport	31.144344	121.808273	Shanghai Jiao Tong University	31.025220	121.433778
China	Hong Kong	Hong Kong International Airport	22.313474	113.913728	University of Hong Kong	22.283089	114.136562
China	Hong Kong	Hong Kong International Airport	22.313474	113.913728	Chinese University of Hong Kong	22.419625	114.206761
Taiwan	Taipei	Taiwan Taoyuan International Airport	25.080490	121.231159	National Taiwan University	25.017340	121.539752
Japan	Tokyo	Tokyo Narita International Airport	35.770178	140.384321	University of Tokyo	35.713816	139.762734
South Korea	Seoul	Incheon International Airport	37.458666	126.441968	Seoul National University	37.464827	126.957199
South Korea	Seoul	Incheon International Airport	37.458666	126.441968	Yonsei University	37.566394	126.938707
Singapore	Singapore	Changi Airport	1.358604	103.989944	National University of Singapore	1.296643	103.776394
Singapore	Singapore	Changi Airport	1.358604	103.989944	Nanyang Technological University	1.348310	103.683135
India	Mumbai	Chhatrapati Shivaji Maharaj International Airport	19.090218	72.862812	Indian Institute of Technology Bombay	19.133060	72.915106
India	Delhi	Indira Gandhi International Airport	28.556144	77.099962	Indian Institute of Technology Delhi	28.545718	77.192768
Australia	Sydney	Sydney Kingsford Smith Airport	-33.950031	151.181694	University of Sydney	-33.888404	151.186765
Australia	Melbourne	Melbourne Tullamarine Airport	-37.669919	144.840345	University of Melbourne	-37.798346	144.960974
UK	London	London Heathrow Airport	51.467991	-0.455051	Imperial College London	51.498822	-0.174873
UK	London	London Heathrow Airport	51.467991	-0.455051	University College London	51.524559	-0.134040
UK	Oxford	London Heathrow Airport	51.467991	-0.455051	University of Oxford	51.757043	-1.254518
UK	Cambridge	London Heathrow Airport	51.467991	-0.455051	University of Cambridge	52.205356	0.113168
France	Paris	Charles de Gaulle Airport	49.007883	2.550785	Sorbonne University	48.846950	2.355570
France	Paris	Charles de Gaulle Airport	49.007883	2.550785	Ecole Normale Superieure	48.842024	2.344430
Germany	Munich	Munich International Airport	48.353987	11.788362	Technical University of Munich	48.148765	11.568176
Italy	Milan	Milan Malpensa Airport	45.622714	8.728234	Politecnico di Milano	45.468503	9.182403
Russia	Moscow	Sheremetyevo International Airport	55.973648	37.412503	Moscow State University	55.703935	37.528669

Table 4: Start points and end points information across 27 major cities in 14 countries.

C Further Analysis

C.1 Time Complexity Analysis

Following Algorithm 2, we analyze the time complexity of our MSGS algorithm in detail. Let $k = |\mathcal{Y}|$ represent the number of node types, with each node type containing the average $m = \frac{|\mathcal{V}|}{|\mathcal{Y}|}$ nodes. First, enumerating all possible node type orders introduces up to $k!$ permutations. In addition, exploring subsets of these k node types involves up to 2^k subsets, each potentially requiring $(k-r)!$ permutations for further exploration. This results in an overall exponential complexity driven by k. Second, for each node type order, the graph construction has a polynomial time complexity $\mathcal{O}(km^2)$, while the Dijkstra search operates with a complexity of $\mathcal{O}\left((km)^2 \log(km)\right)$. Since $(km)^2 \log(km)$ dominates km^2, the worst-case complexity can be simplified to $\mathcal{O}\left(2^k \cdot k! \cdot (km)^2 \log(km)\right)$.

Considering the early stop mechanism (Algorithm 2, Line 10), the algorithm progressively explores combinations from k node types to 1 node type until finding a valid solution. The best-case time complexity becomes $\mathcal{O}\left(k! \cdot (km)^2 \log(km)\right)$ when a valid route containing all POI types is found immediately. The worst-case time complexity remains $\mathcal{O}\left(2^k \cdot k! \cdot (km)^2 \log(km)\right)$ if the algorithm needs to explore all possible combinations. In our routing scenario, the best case occurs when the planned route satisfies all constraints while including all POI types, while the worst case happens when including any POI type violates the constraints. According to Table 1, more than 95% of the cases successfully include all requested POI types, indicating that our algorithm typically achieves near best-case performance in practice.

C.2 Runtime Analysis

Beyond the theoretical time complexity analysis, our empirical runtime measurements in Table 5 demonstrate that the LLMAP system significantly outperforms baseline approaches in computational efficiency. In contrast to LLM-as-Agent, our method utilizes LLMs solely for parsing human instructions, effectively limiting input length and reducing execution time. Although the computational cost of LLM-as-Agent scales linearly with the number of POIs, our approach remains independent of POI counts. Solver-based methods struggle with our optimization problem due to its multi-step, multi-objective, and multi-constraint nature, particularly given the conflicting objectives. To address these challenges, LLMAP employs a modular approach where different components serve distinct purposes: LLMs parse human instructions, while MSGS constructs subgraphs incrementally to satisfy multiple constraints and objectives. Moreover, we optimize the time complexity of MSGS through several key tricks, including early dependency constraint checking, omitting edges between nodes of the same type, and early stop criteria.

Method	Parsing Time (s)	Planning Time (s)	Total Time (s)
LLM-as-Agent	-	18.89	18.89
SMT Solver	1.49	9.32	10.81
SMT Solver v2	1.49	28.38	29.87
Ours	1.49	0.29	1.78

Table 5: Runtime analysis of LLMAP and baseline approaches on LLaMA-3.2-3B (CoT). Parsing time refers to the runtime of LLM-as-Parser, while planning time denotes the runtime of our MSGS algorithm or SMT solver execution.

C.3 Ablation Study

We present an ablation study in Table 1 to evaluate the performance impact of removing two key components from our LLMAP system: LLM-as-Parser and MSGS. For the former, we replace LLM-as-Parser outputs with synthetic labels, while for the latter, we directly apply the Dijkstra algorithm on the complete graph without considering the multi-step subgraph construction process. The synthetic labels can be viewed as an approximate ideal parsing scenario, though they do not represent a performance upper bound since user preference weights influence route evaluation metrics. We observe that LLMAP achieves comparable performance to this ideal scenario across multiple LLMs, as highlighted in Table 1 (e.g., GPT-4o, Phi-3-mini (CoT), Phi-3.5-mini (CoT), Mistral-7B (CoT), Gemma-2-2B (CoT), and GPT-4o (CoT)), validating the effectiveness of our LLM-as-Parser approach.

Regarding the w/o MSGS ablation, this variant fails to generate meaningful routes, instead producing direct paths from the start point to the end point. This occurs because applying the Dijkstra algorithm directly on the graph focuses solely on minimizing weighted distances (based on ratings, number of reviews, and length) without maximizing task completion rates or considering constraints.

Average Speed (km/h)	Rating (↑)	Number of Review (↑)	Length (km) (↓)	Task Completion Rate (%) (↑)	Constraint Violation (↓) Time Limit (hrs)	Dependency (%)	Opening Hours (%)
5 (walking)	3.29	5178	28.97	61.11	0.00	0.00	0.00
10 (bicycle)	4.06	5653	29.54	88.09	0.00	0.00	0.00
20 (bus)	4.28	5676	29.72	95.83	0.00	0.00	0.00
30 (car)	4.28	5530	29.73	96.44	0.00	0.00	0.00
40	4.26	5133	29.71	95.99	0.00	0.00	0.00
50 (UAV)	4.25	5072	29.71	95.81	0.00	0.00	0.00
60	4.25	5052	29.69	95.77	0.00	0.00	0.00

Table 6: Impact of different transportation modes.

Departure Day	Rating (↑)	Number of Review (↑)	Length (km) (↓)	Task Completion Rate (%) (↑)	Constraint Violation (↓) Time Limit (hrs)	Dependency (%)	Opening Hours (%)
Monday	4.28	5530	29.73	96.44	0.00	0.00	0.00
Tuesday	4.28	4719	29.72	96.51	0.00	0.00	0.00
Saturday	4.00	5711	29.31	75.84	0.00	0.00	0.00
Sunday	3.41	7273	28.93	50.59	0.00	0.00	0.00

Table 7: Impact of different departure days.

Departure Time (HH:MM)	Rating (↑)	Number of Review (↑)	Length (km) (↓)	Task Completion Rate (%) (↑)	Constraint Violation (↓) Time Limit (hrs)	Dependency (%)	Opening Hours (%)
08:00	3.72	2039	29.29	75.44	0.00	0.00	0.00
10:00	4.28	5530	29.73	96.44	0.00	0.00	0.00
12:00	4.28	5852	29.70	95.53	0.00	0.00	0.00
14:00	4.18	5502	29.50	88.76	0.00	0.00	0.00
16:00	3.68	5521	29.11	64.17	0.00	0.00	0.00
18:00	3.06	5987	28.74	43.11	0.00	0.00	0.00

Table 8: Impact of different departure times.

This demonstrates the necessity of MSGS's multi-step multi-objective optimization approach, which first maximizes task completion rates under constraints before trade-off multiple objectives.

C.4 Impact of Transportation Mode, Departure Day, and Departure Time

Our LLMAP system demonstrates substantial flexibility in accommodating diverse transportation configurations, analogous to Google Maps functionality, thereby enabling users to dynamically calibrate parameters in accordance with their specific requirements. We conduct comprehensive experiments to examine the influence of transportation modalities, departure scheduling, and temporal constraints on system performance metrics.

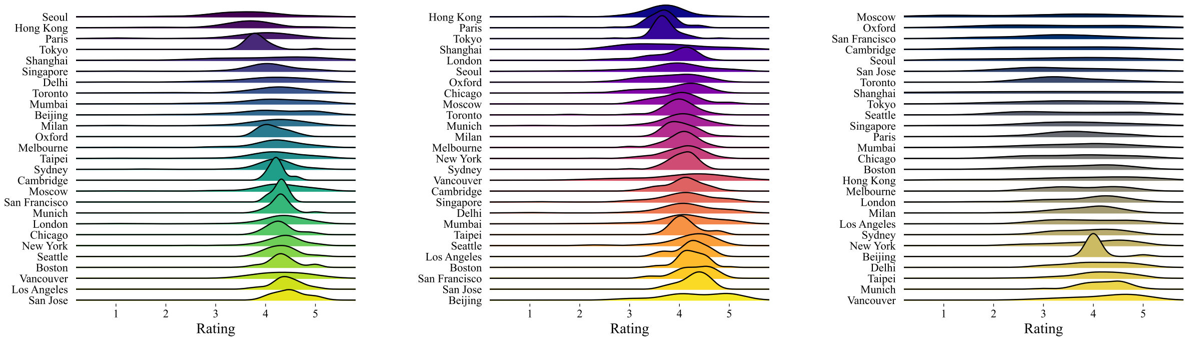
First, transportation modalities characterized by varying average velocities exhibit a pronounced correlation between mobility speed and task completion efficacy, as demonstrated in Table 6. Slower modalities such as pedestrian movement (5 km/h) yield diminished completion rates (61.11%), whereas expedited modalities including public transit (20 km/h) and private vehicular transport (30 km/h) achieve completion rates exceeding 95%. These findings indicate that users should prioritize transportation modality selection based on task-specific requirements and performance objectives. Second, departure day analysis reveals substantial performance variations between weekdays and weekends, as illustrated in Table 7. The system attains optimal completion rates exceeding 96% during weekdays, while weekend performance exhibits considerable degradation (Saturday: 75.84%; Sunday: 50.59%) attributable to the closure of numerous POIs during rest periods. Finally, temporal sensitivity analysis presented in Table 8 demonstrates optimal system performance during mid-morning intervals (10:00-12:00) with completion rates surpassing 95%, while early morning (08:00) and evening (18:00) departures manifest reduced performance metrics (75.44% and 43.11%, respectively). Our LLMAP system enables manual departure time modification and provides intelligent recommendations upon detecting POI closures, maintaining consistency with Google Maps functionality. Adaptive departure time optimization presents challenges, requiring balance between departure timing and dwell duration.

C.5 Impact of User Location and POI Types

Figure 7 illustrates the distribution of POI ratings and review counts across different cities. The data reveals significant heterogeneity across cities and POI types, which presents a fundamental challenge in serving users worldwide. The frequency of Google Maps usage varies substantially across cities. For example, in Chinese metropolises like Beijing and Shanghai, despite their high population density, the number of reviews is notably lower due to Google Maps' limited popularity among Chinese users. However, international tourists visiting China may be unfamiliar with or have restricted access to local mapping services. A key advantage of our system is its ability to perform route planning on such unpredictable data distributions without requiring any training. Across different POI types, shopping malls and supermarkets exhibit review counts approximately an order of magnitude higher than pharmacies, banks, and libraries, primarily due to their significantly higher foot traffic. Benefit by the unique multi-step optimization design of our MSGS algorithm, which prioritizes task completion rate before optimizing other metrics, these order-of-magnitude differences in POI types do not impede our ability to make appropriate POI selections.

C.6 Integration of Essential Waypoints

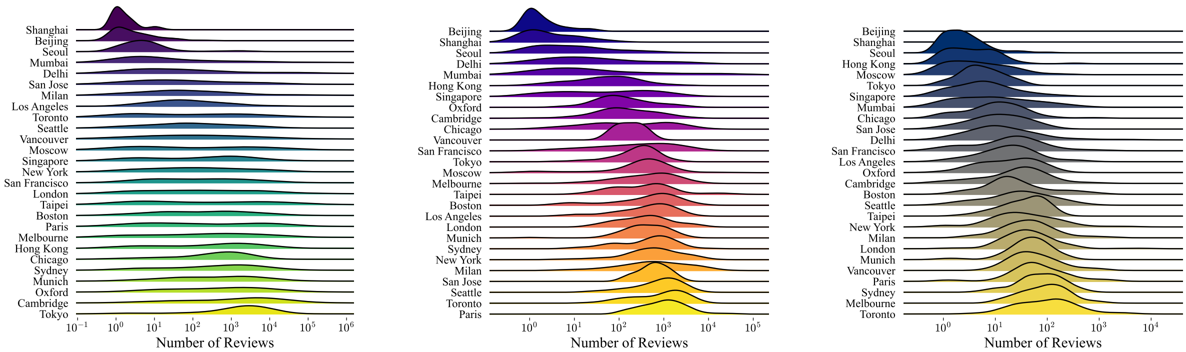
Our LLMAP system currently focuses on effectively removing POIs based on various constraints, while also possessing the potential to implement integration of essential waypoints functionality into route planning. This functionality requires domain-specific knowledge to formulate constraint rules for implementation. Similar to the three constraints established in the current LLMAP system, adding waypoints functionality can be categorized as originating from either user instructions or environmental information. For example, the vehicle's fuel level constitutes one of the constraints within the aforementioned environmental information. To develop an automatic gas station addition feature, specifically a fuel constraint, we need to determine that the current fuel level proves insufficient to complete the route based on vehicle fuel capacity, route length, and fuel consumption rate, thereby identifying the necessity to incorporate a gas station. Although intuitively one might anticipate that removing POIs and adding essential waypoints could generate conflicts, as they represent two distinct approaches to satisfying constraint conditions, our designed multi-step optimization strategy readily resolves these potential conflicts. Since, after fulfilling all constraint conditions, the subsequent step maximizes $\mathcal{O}_{(\mathrm{i})}$ in Equation (1), which represents the task completion rate. Evidently, these newly incorporated essential waypoints, such as gas stations, do not constitute components of the task completion rate but have already been addressed within the constraints. Consequently, this strategic design establishes a sequential order for removing POIs and adding waypoints, ensuring that we do not eliminate task-relevant POIs to satisfy constraints requiring essential waypoint addition.



(a) Shopping Mall - Rating

(b) Supermarket - Rating

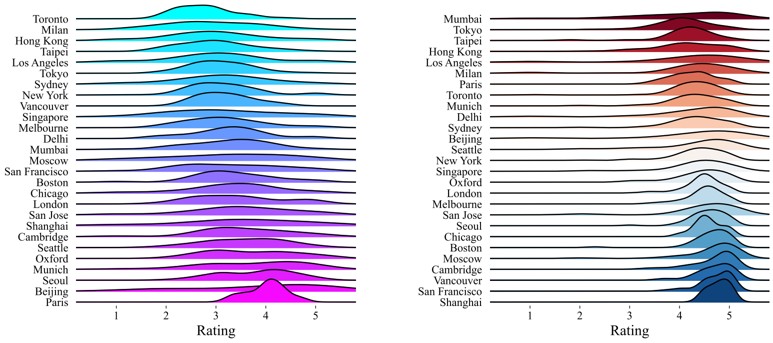
(c) Pharmacy - Rating



(d) Shopping Mall - Number of Reviews

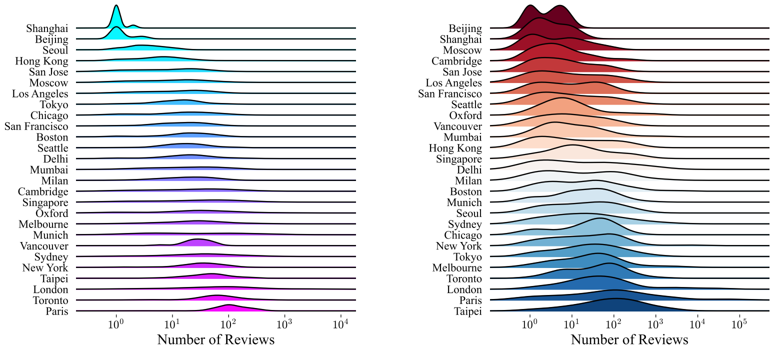
(e) Supermarket - Number of Reviews

(f) Pharmacy - Number of Reviews



(g) Bank - Rating

(h) Library - Rating



(i) Bank - Number of Reviews

(j) Library - Number of Reviews

Figure 7: Distribution of ratings and number of ratings for different POI types across different cities.

D Prompt Template

D.1 Human Instruction Generation

Synthetic Label → Human Instruction (default with CoT prompting):

```
"system_prompt" =
```
You need to generate a natural human instruction by following these thinking
    steps:

1. First sentence: State all the POIs that need to be visited today:
   - Look at the list of POIs
   - Think about natural way to express visiting multiple POIs

2. Second sentence: State the return time if specified:
   - If specific time given, express as deadline
   - If no time limit, omit this part

3. Third sentence: Express POI rating vs route length preference indirectly:
   - If POI rating weight > 0.5: Express strong desire for high rating POIs
   - If route length weight > 0.5: Express urgency or need for efficiency
   - If balanced: Express desire for both reasonable rating and efficiency

4. Forth sentence: Express each dependency separately, one by one:
   - Convert each [A,B] pair into natural sequence requirement
   - Think about natural ways to express "must visit A before B"

Keep the language natural but always follow this structure.
```
```

```
"user_prompt" =
```
Based on the following information, generate a natural human instruction:

1. POIs to visit: {', '.join(synthetic_label['pois'])}

2. Return by: {synthetic_label['time_limit']}

3. Preference analysis:
- POI rating weight: {synthetic_label['quality_weight']}
- Route length weight: {synthetic_label['distance_weight']}

4. Dependencies to express: {synthetic_label['dependencies']}

Generate ONE natural instruction that includes all this information. No prefix
    , additional text, or explanation.
```
```

Figure 8: Prompt template for generating natural human instruction from synthetic label using GPT-4o with CoT prompting.

D.2 LLM-as-Parser Estimation

LLM-as-Parser – Human Instruction → Estimation (without CoT prompting):

```
"system_prompt" =
```
Extract POIs, constraints, and weights from a human instruction. POIs should
    be a simple string array. Dependencies should be an empty list if no
    sequence requirements are mentioned. Weights should be between 0 and 1, and
     sum to 1. Extract them from language about importance of POI rating vs
    route length.

Output must be valid JSON with structure:
{
```
```

~~~
    "pois": ["poi1", "poi2", ...],
    "time_limit": "HH:00" or "None",
    "dependencies": [["poi1", "poi2"], ...],
    "quality_weight": float,
    "distance_weight": float
}
```
~~~

~~~
"user_prompt" =
```
Extract POIs, constraints, and weights from this instruction as JSON:
{instruction}

Only output the JSON object. No prefix, additional text, or explanation.
```
~~~

Figure 9: Prompt template for POI extraction, constraint identification, and user preference estimation using LLM-as-Parser without CoT prompting.

LLM-as-Parser – Human Instruction → Estimation (with CoT prompting):

~~~
"system_prompt" =
```
Extract POIs, constraints, and weights from human instruction through step-by-
    step reasoning:

1. POIs Analysis:
    - Look for POIs mentioned that need to be visited
    - Create list of unique POIs

2. Time Limit Analysis:
    - Search for any specific return time
    - Format as HH:00 or "None" if not specified

3. Dependencies Analysis:
    - Look for words indicating sequence (before, after, then, etc.)
    - Create pairs of [POI1, POI2] for each sequence requirement

4. Preference Analysis:
    - Look for language about POI quality/rating importance vs route
      efficiency
    - High POI rating emphasis (quality, best places, etc.) -> quality_weight
      should be large than 0.5
    - High route efficiency emphasis (quick, shortest, save time) ->
      distance_weight should be large than 0.5
    - Balanced language -> both weights around 0.5

Output must be valid JSON with structure:
{
    "pois": ["poi1", "poi2", ...],
    "time_limit": "HH:00" or "None",
    "dependencies": [["poi1", "poi2"], ...],
    "quality_weight": float,
    "distance_weight": float
}
```
~~~

~~~
"user_prompt" =
```
{instruction}

Only output the list object. No prefix, additional text, or explanation.
```
~~~

Figure 10: Prompt template for POI extraction, constraint identification, and user preference estimation using LLM-as-Parser with CoT prompting.

D.3 LLM-as-Agent Estimation

LLM-as-Agent – Human Instruction → Estimation (without CoT prompting):

~~~
”system_prompt” =
```
You are a route planning assistant. Your goal is to plan an optimal route
    based on the following objectives:

Primary Objectives:
1. Minimize the total route length/distance
2. Maximize coverage of different POI types (select exactly one POI per
    required type)
3. Maximize the quality of visited POIs (based on ratings and number of
    ratings)
4. Balance between route efficiency and POI quality
5. Ensure compliance with time limits from instructions
6. Account for dependencies between POIs
7. Respect opening hours of recommended POIs

Visit Duration: shopping mall: 120 mins, supermarket: 30 mins, pharmacy: 15
    mins, bank: 20 mins, library: 60 mins
Travel Speed: 30 km/h
Departure Time: 10:00 AM

Always output POI IDs as provided in the input data. Your output must strictly
     follow this format:
[POI ID, POI ID, POI ID]

Available POIs:
    POI ID: ChIJQ0eRlQ1644kR11stZbdBvM0:
    * Type: shopping mall
    * Rating: 4.4 (4110 reviews)
    * Coordinates: (42.3471832, -71.0778024)
    * Opening Hours: Monday: 11:00 AM - 7:00 PM

    ...... # Omitting the remaining 9 shopping malls

    POI ID: ChIJ296XtDV344kRXTFzOH4n2Vg:
    * Type: supermarket
    * Rating: 4.4 (4547 reviews)
    * Coordinates: (42.38078309999999, -71.10163109999999)
    * Opening Hours: Monday: 7:00 AM - 9:00 PM

    ...... # Omitting the remaining 9 supermarkets

    POI ID: ChIJf52pw8lw44kRzVIBthOOKgM:
    * Type: pharmacy
    * Rating: 4.1 (38 reviews)
    * Coordinates: (42.37640409999999, -71.0901677)
    * Opening Hours: Monday: 9:00 AM - 1:30 PM, 2:00 - 7:00 PM

    ...... # Omitting the remaining 9 pharmacies

    POI ID: ChIJXcQ5xYNw44kRvDwvuZw29wI:
    * Type: bank
    * Rating: 4.5 (114 reviews)
    * Coordinates: (42.35561349999999, -71.0612641)
    * Opening Hours: Monday: 9:00 AM - 5:00 PM

    ...... # Omitting the remaining 9 banks

    POI ID: ChIJU_ibQgx644kRc6zBW_5d1Lk:
    * Type: library
    * Rating: 4.8 (2770 reviews)
    * Coordinates: (42.3493136, -71.0781875)
~~~

```
    * Opening Hours: Monday: 9:00 AM - 8:00 PM

    ...... # Omitting the remaining 9 libraries
```
```

```
"user_prompt" =
```
```
{instruction}

Only output the list object. No prefix, additional text, or explanation.
```
```

Figure 11: Prompt template for POI extraction, constraint identification, and user preference estimation using LLM-as-Agent without CoT prompting. Due to input token length limitations, we sample 10 specific POIs for each POI type as input for LLM-as-Agent.

LLM-as-Agent – Human Instruction → Estimation (with CoT prompting):

```
"system_prompt" =
```
```
You are a route planning assistant. Your goal is to plan an optimal route
    based on the following objectives:

Primary Objectives:
1. Minimize the total route length/distance
2. Maximize coverage of different POI types (select exactly one POI per
    required type)
3. Maximize the quality of visited POIs (based on ratings and number of
    ratings)
4. Balance between route efficiency and POI quality
5. Ensure compliance with time limits from instructions
6. Account for dependencies between POIs
7. Respect opening hours of recommended POIs

Visit Duration: shopping mall: 120 mins, supermarket: 30 mins, pharmacy: 15
    mins, bank: 20 mins, library: 60 mins
Travel Speed: 30 km/h
Departure Time: 10:00 AM

Planning Process:
1. Analyze User Requirements:
  - Identify required POI types from human instruction
  - Note any specified preferences for particular types

2. Prioritize POI Types:
  - Order POI types based on:
    * User specified preferences/requirements
    * Dependencies between types
    * Opening hours

3. Select Specific POIs:
  - For each POI type in order:
    * Consider only POIs of that specific type
    * Choose exactly one POI based on:
      - Rating and number of reviews
      - Location efficiency (distance to previous/next points)
      - Opening hours compatibility
  - Ensure only one POI is selected per type

First analyze the constraints and requirements, then plan accordingly. After
    planning, validate that your route satisfies all constraints.

Always output POI IDs as provided in the input data. Your output must strictly
     follow this format:
```

~~~
[POI ID, POI ID, POI ID]

Available POIs:
    POI ID: ChIJQ0eRlQ1644kR11stZbdBvM0:
    * Type: shopping mall
    * Rating: 4.4 (4110 reviews)
    * Coordinates: (42.3471832, -71.0778024)
    * Opening Hours: Monday: 11:00 AM - 7:00 PM

    ...... # Omitting the remaining 9 shopping malls

    POI ID: ChIJ296XtDV344kRXTFzOH4n2Vg:
    * Type: supermarket
    * Rating: 4.4 (4547 reviews)
    * Coordinates: (42.38078309999999, -71.10163109999999)
    * Opening Hours: Monday: 7:00 AM - 9:00 PM

    ...... # Omitting the remaining 9 supermarkets

    POI ID: ChIJf52pw8lw44kRzVIBthOOKgM:
    * Type: pharmacy
    * Rating: 4.1 (38 reviews)
    * Coordinates: (42.37640409999999, -71.0901677)
    * Opening Hours: Monday: 9:00 AM - 1:30 PM, 2:00 - 7:00 PM

    ...... # Omitting the remaining 9 pharmacies

    POI ID: ChIJXcQ5xYNw44kRvDwvuZw29wI:
    * Type: bank
    * Rating: 4.5 (114 reviews)
    * Coordinates: (42.35561349999999, -71.0612641)
    * Opening Hours: Monday: 9:00 AM - 5:00 PM

    ...... # Omitting the remaining 9 banks

    POI ID: ChIJU_ibQgx644kRc6zBW_5d1Lk:
    * Type: library
    * Rating: 4.8 (2770 reviews)
    * Coordinates: (42.3493136, -71.0781875)
    * Opening Hours: Monday: 9:00 AM - 8:00 PM

    ...... # Omitting the remaining 9 libraries
```
~~~

~~~
”user_prompt” =
```
{instruction}

Only output the list object. No prefix, additional text, or explanation.
```
~~~

Figure 12: Prompt template for POI extraction, constraint identification, and user preference estimation using LLM-as-Agent with CoT prompting. Due to input token length limitations, we sample 10 specific POIs for each POI type as input for LLM-as-Agent.