LLM-Augmented Optimization for Singapore Travel Itinerary Planning

Daniel James[a,*], **Leonardo**[a,**], **Jing Shen Tai**[a,***], **Valerian Yap**[a,****] **and Hoong Chuin Lau**[a,*****]

[a]School of Computing and Information Systems, Singapore Management University
ORCID (Daniel James): https://orcid.org/....-....-....-...., ORCID (Leonardo): https://orcid.org/....-....-....-...., ORCID (Jing Shen Tai): https://orcid.org/....-....-....-...., ORCID (Valerian Yap): https://orcid.org/....-....-....-...., ORCID (Hoong Chuin Lau): https://orcid.org/....-....-....-....

Abstract. This paper presents a combined approach of Large Language Model (LLM) and Operation Research (OR) to develop an optimized travel planner for tourists visiting Singapore. Recognizing that traveler preferences vary significantly across demographics, we aim to generate customized itineraries that balance cost, travel time and personal satisfaction. We apply LLM agents to convert traveler's preferences in text format into a structured format that can be used for optimization. In optimization's case, we apply Adaptive Large Neighborhood Search (ALNS) with data enrichment techniques such as route matrix from Google Maps API to find out duration and price from point A to point B. Our contribution is the ability to make more-realistic itineraries, putting distance and cost into perspective, while focusing on traveler's best interest to minimize expenses and maximize satisfaction. We benchmarked against ... . Our approach ..., demonstrating the power of leveraging strengths of both LLM and ALNS in solving this problem.

1 Introduction

The objective of this project is to develop a personalized travel itinerary planner for tourists visiting Singapore, capable of allowing natural language inputs and input fields into the demo product and generating a feasible yet personalized itinerary by minimizing travel cost, minimizing total transit time between locations, and maximizing satisfaction that aligns with Persona-specific preferences.

Our commercial inspiration was primarily drawn from Pelago by Singapore Airlines, an AI-powered trip planner platform that covers over 2,000 destinations. While Pelago appears to be employing an LLM-based recommendation engine, our work diverges by introducing a multi-agent LLM system that is combined with Operation Research (OR) optimization techniques.

Our goal in this paper includes answering the following questions:

1. Recognising the hallucination in LLM, to what extent can LLM alone generate realistic and feasible travel itineraries? Is Agentic AI needed to be used in our project?
2. Assuming we have an LLM agent (e.g. a domain-expert in Singapore Attractions, equipped with memory, knowledge base and tools), can it handle reasoning consistency and personalisation without hallucinating? What is the trade-off for having multiple domain experts in our system?
3. What are the quantitative and qualitative trade-offs between optimization-only, LLM-only, multi-agent, and hybrid optimization itinerary planning pipelines?

This study is geographically and thematically scoped to the context of Singapore considering our familiarity of our local culture. Specifically, we focus on two categories of points of interest (POIs): (i) attractions, and hawker centres. In total, our curated dataset includes over 85 unique POIs.

2 Related Work

Recent works have shown promise in understanding natural language preferences in the same domain research that we are working on:

2.1 TRIP-PAL

!!! CITE REFERENCE. Planning with Guarantees by Combining LLMs and Automated Planners (JP Morgan AI Research, 2024), which explores the combination of LLMs and formal planners (like A-star heuristic) to provide reliable trip plans.

2.2 Optimizing Travel Itineraries with AI Algorithms in a Microservices Architecture: Balancing Cost, Time, Preferences, and Sustainability

!!! CITE REFERENCE. Demonstrates how various optimization objectives can be orchestrated via modular components with the use of Machine Learning (ML) Model for personalization.

3 Problem Definition

In this paper, we propose My Travel Itinerary Buddy – Automatic Itinerary (MITB – AI), where the goal is to generate multi-day travel itinerary for a tourist visiting Singapore, consisting of a sequence of POIs – including both attractions and hawker centres. This is also subjected to user-defined constraints (e.g. budget, number of days and person types) while optimizing for the following objectives: (1) minimize costs, (2) minimize travel time, and (3) maximize traveler's satisfaction from online ratings. This problem can be classified as a

* Corresponding Author. Email: danieljames.2023@mitb.smu.edu.sg.
** Corresponding Author. Email: leonardo.2023@mitb.smu.edu.sg.
*** Corresponding Author. Email: js.tai.2023@mitb.smu.edu.sg.
**** Corresponding Author. Email: valerianyap.2023@mitb.smu.edu.sg.
***** Corresponding Author. Email: hclau@smu.edu.sg.

multi-objective combinatorial optimization task, where the system must select and order POIs over multiple days while satisfying both hard and soft constraints (Fan, et al., 2024).

3.1 Assumptions

The assumptions that we have incorporated in our itinerary planners includes:

3.1.1 Daily time Windows

We assumed that all POIs are open from 9:00 AM to 10:00 PM and that travelers return to the hotel by the end of the day. Each day of the itinerary follows a fixed schedule:

Start of day	End of day	Lunch Window	Dinner Window
09:00	22:00	11:00 - 15:00	17:00 – 21:00

3.1.2 Additional Modelling Assumptions

1. All inter-POI travel times are retrieved using Google Maps API, segmented into four time-of-day brackets (morning, afternoon, evening, night) to reduce excessive API calls.
2. The itinerary assumes a fixed accommodation location, Marina Bay Sands, as the default hotel for all nights, does not optimize for hotel selection to simplify routing and focus on attraction and food planning. Budget input by traveler does not include the predefined hotel as well.
3. Different Transport fare models for public transport or ride-hailing may apply based on rider types (e.g., adult, senior, tourist), but we have normalized under a general fare assumption in this study.
4. In ensuring trip continuity, the planner supports multi-segment trips (e.g. attraction -> food -> another attraction), and travel pricing is computed cumulatively across trip legs.

3.2 Datasets

The following datasets have been used for our itinerary planner to meet multiple objectives:

- Attractions in Singapore: Geolocation, category, estimated visit duration, entrance fees and descriptions of attractions. Entrance fees enriched via LLM agents. Source: https://data.gov.sg/datasets/d_0f2f47515425404e6c9d2a040dd87354/view.
- Singapore Hawker Centres: Geolocation, names of hawker centres and estimated costs. Source: https://data.gov.sg/datasets/d_4a086da0a5553be1d89383cd90d07ecd/view.

3.3 Route Matrix Generation using Google Maps API

To support accurate travel time estimation between POIs, we constructed a route matrix using Google Maps API. This matrix serves as a critical yet feasible input for downstream optimization:

1. Data Collection - We first consolidated attractions and hawker centres from publicly available sources into CSV files. To support location retrieval and recommendation by the LLM/Agents, we hosted the tabular data in a Postgres Database within a docker container. These datasets provided key details such as location names, estimated visit durations and costs.
2. Using Google Places API, we extracted latitude and longitude coord. for each POI. These waypoints were to define nodes of our routing graph and serve as input to our route matrix computation.
3. We then employed the Google Compute Route Matrix API to generate a travel-time and distance matrix between all waypoints for different transport modes (drive, transit). This step provided the necessary data to formulate the optimization problem for itinerary scheduling.

To manage costs and API quota constraints, we binned the travel time estimates into four main brackets:

Morning	Afternoon	Evening	Night
08:00 – 12:00	12:00 – 16:00	16:00 – 20:00	20:00 – 08:00

For example, a query from Marina Bay Sands to Maxwell Food Centre at 17:58 would be approximated using the route data retrieved at 16:00. While this introduces some granularity loss due to timing of retrieval of data, it represents a practical trade-off between cost, accuracy and scalability. This approach would help us ensure reasonable accurate transit estimates while keeping the routing component efficient enough for integration with our optimization engine.

4 Contribution

!!! HELP THIS SECTION. Not to rely on outdated city census data Domain expert view (pre-selection) not possible for large scale comparison and large n since it is continuous, it is not possible to find by manual methods (scalable) and need a smart way of approximation many applications

5 Case Study

TBA

6 Proposed Approach

6.1 LLM and Multi-Agent Framework

Although LLMs are highly capable at interpreting and generating human-like text, they are passive systems—limited to single-turn type of interaction without persistent memory. This presents clear limitations when applied to travel itinerary planning, a task that requires structured decision-making, retrieval of external data (e.g., Google ratings, Cost of Attraction Entrance Fees), and context tracking across multiple steps. A passive LLM may generate a generic itinerary but lacks the capability to provide personalization such as "I prefer scenic routes" or "maximize shopping time within budget."

To bridge this gap, we extend LLMs into autonomous agents by integrating three key capabilities: tool-calling (e.g., invoking Google Maps APIs), memory (to retain user goals and prior decisions), and Retrieval Augmented Generation (RAG) for incorporating external data into the reasoning process. This turns the LLM from a reactive text generator into a goal-driven planner capable of making informed decisions. Our system adopts this Agentic RAG architecture to retrieve attraction details, estimating hawker visit durations, and enrich itinerary planning with contextual knowledge beyond the LLM's static knowledge from pre-training.

While a single agent can technically handle the entire itinerary pipeline, research has also shown that such monolithic setups often struggle with domain specialization. To address the limitation, we extend our architecture into a multi-agent framework where each agent focuses on a well-scoped domain task.

!!! ADD ARCHITECTURE FIGURE HERE

6.1.1 Supervisor Agent

This agent serves as a centralized "guardrails". It evaluates incoming user queries for intent, toxicity, and feasibility before routing them to downstream domain-specific agents. This agent filters out harmful or nonsensical inputs (e.g., security threats development in Singapore) while passing valid requests (e.g., budget-aware itinerary planning) to the appropriate specialized agents.

6.1.2 Attraction Agent

This agent is responsible for processing the user's interests related to sightseeing, cultural landmarks, and activity-based preferences. It first performs semantic retrieval on the user query to identify relevant attraction-related intents, such as interest in "free attractions," "shopping," or "nature". It then queries a local database hosted on PostgreSQL (via Docker) to retrieve structured metadata about available attractions.

6.1.3 Hawker Agent

Very similar to Attraction Agent, this agent is responsible for processing the user's interests related to food preferences. It first performs semantic retrieval on the user query to identify relevant food-related intents, such as interest in "spicy food" or "halal-only". It then queries a local database hosted on PostgreSQL (via Docker) to retrieve structured metadata about available hawker centres.

6.1.4 Code Agent

This agent bridges the gap between natural language preferences and formal optimization logic. This agent transforms high-level user intent into structured parameters and constraint functions that can be directly ingested by the metaheuristic optimizer.

6.1.5 Variable Agent

Aside from the additional constraints added by code agents, there are also vectors for the weights of prioritization that the Optimizer will use to find the best itinerary for the current persona. The role of a Variable Agent is to translate traveler's persona into a weight prioritization for cost, travel duration, and satisfaction. For example, the agent will generate $w = [0.6, 0.2, 0.2]$ to reflect a user preference for minimizing budget over time and satisfaction.

6.1.6 Feedback Agent

TBA

6.2 Adaptive Large Neighborhood Search (ALNS)

!!! MIGHT NEED TO TRIM A LOT FROM THIS, AS THIS IS NOT THE FOCUS OF THIS PAPER. ALNS is designed to solve optimization problems by iteratively destroying and repairing solutions. While traditional methods offer precise formulations, they struggle with scaling and flexibility in real-world data scenarios. In our project, ALNS was found to deal with our optimization problem better.

6.2.1 Construction Heuristic

Our Initial Heuristic provides a feasible solution and good starting point for further iterations.

- Starting each day at the hotel and returning to the same hotel Order attractions and hawker centres (descending) based on a value ratio: $\frac{Satisfaction}{(Cost+Duration)}$
- Adding attractions before lunch time $\rightarrow$ Scheduling lunch at a highly rated hawker centre
- Adding more attractions in the afternoon $\rightarrow$ Scheduling dinner at another hawker centre
- Maintains budget constraints, time windows, and ensures each POI is visited at most once across the entire itinerary.

6.2.2 Destroy Operators

Destroy operators remove portions of a solution to allow exploration of new regions in the search space.

1. Destroy Targeted Subsequence: Removes a sequence of attractions while ensuring meals remain scheduled.
   - Given a sequence $S = \{v_1, v_2, ..., v_n\}$, a randomly chosen subsequence $S' \subset S$ is removed: $S' = \{v_k, ..., v_m\}$, where k, m are randomly chosen indices
2. Destroy Worst Attractions: Removes attractions with the lowest satisfaction-to-cost ratio.
   - Ranking function: $R(i) = \frac{satisfaction_i}{cost_i}$
   - Attractions in the lowest percentile are removed.
3. Destroy Distant Locations: Eliminates locations requiring excessive travel.
   - Distance weight: $w_{dist} = \alpha * d(i, j) + \beta * cost_{travel}$
   - Locations with the highest w_{dist} are removed.
4. Destroy Expensive Attractions: Eliminates attractions when budget constraints are exceeded.
   - If total cost exceeds the budget, as expressed in this equation: $\sum_{i \in V} cost_i > B$, then most expensive locations are iteratively removed until constraints are met.
5. Destroy Selected Day: Entire day schedule is replaced while keeping mealtimes with probability p.

6.2.3 Repair Operators

Repair operators insert new locations into an incomplete solution.

1. Repair Regret Insertion: Selects the best insertion location based on regret value. Locations with the highest regret are inserted first, where $Regret(i) = \sum_{j=1}^{k} (best\, cost(j) - cost(i, j))$.
2. Repair Transit Efficient Insertion: Prioritizes minimizing travel time.
   - Transit weight is defined as: $w_{transit} = \gamma * t_{travel} + \delta * cost_{transport}$ where γ and δ are tuning parameters.
3. Repair Satisfaction Driven Insertion: Maximizes overall satisfaction using: $w_{satisfaction} = \lambda * satisfaction - \mu * (cost + duration)$ where λ and μ balances satisfaction vs cost.

6.2.4 *Acceptance Criteria*

The acceptance criterion determines whether a new solution should replace the current one.

Using Simulated Annealing-based acceptance:

- **Improved solutions always accepted**: $f(S_{new}) < f(S_{current}) \Rightarrow S_{current} \leftarrow S_{new}$
- **Worse solutions accepted probabilistically**: $P_{accept} = e^{((-\Delta f)/T)}$ where $\delta f = f(S_{new}) - f(S_{current})$ and T is the current temperature.
- **Temperature Update**: $T = \alpha T, \quad 0 < \alpha < 1$. gradually reducing acceptance probability over time.

This allows ALNS to escape local optima and explore diverse solutions.

6.2.5 *Objective Function*

The optimization problem in our itinerary planning framework aims to balance cost, travel time, and user satisfaction. To achieve this, we define a weighted objective function, where each component is normalized and constrained to ensure practical and meaningful solutions. The formulation is as follows:

$$Z = w_1 . \min(\frac{Cost}{Budget}, Limit_{cost}) + w_2 . \min(\frac{TravelTime}{TravelTime + TravelTime_{reference}}, Limit_{time}) + w_3 . \max(\frac{Satisfaction}{Satisfaction_{max}}, Limit_{satisfaction}) + P.1_{infeasible}$$

Where:

- w_1, w_2, w_3 are the **weight factors** assigned to each objective based on the user's persona.
- **Limits** $Limit_{cost}, Limit_{time}, Limit_{satisfaction}$ are introduced to prevent the algorithm from over-prioritizing any objective beyond practical levels.
- **Reference values** ensures travel time is assessed relative to a baseline, preventing the optimizer from producing unrealistic itineraries (e.g. min. travel time by staying at home except for meals).
- **Penalty Term** $P.1_{infeasible}$ strongly discourages infeasible solutions by adding a large penalty when any hard constraint is violated.

This objective function effectively balances competing factors, ensuring itineraries are cost-efficient, time-conscious, and aligned with user preferences while maintaining feasibility and practicality.

7 Experimental Result

TBA

8 Conclusion

TBA