BC2411 Prescriptive Analytics with Generative AI
AY 2025/26 Semester 2
Group Project Report
Instructor: Prof. Tang Qingshen
Seminar Group: 2
Group: 3

Group Members	**Matriculation Number**
Chong Jinsheng	U2340252J
Nasywa Hanan Huriyah	U2340940H
Loo Jin Rong	U2210944H
Luey Zhong Han	U2310908J
Nicole Ang Yi Ning	U2310687L

Executive Summary

Travel planner apps like Wanderlog have made trip planning significantly easier and are popular among young travellers. Wanderlog was founded in 2019. It simplifies itinerary creation, provides recommendations, and offers travel optimization through its premium membership. Despite its popularity, the app still faces some key shortcomings:

1. Limited customer preferences
2. Lack of flexibility for Start-End Timing Support

To tackle these challenges, we developed WanderWise, an enhanced itinerary optimization system. WanderWise combines the power of OpenStreetMap for accurate distance data with a tailored Linear Programming Model, inspired by the Prize-Collecting Travelling Salesman Problem. It is designed to create highly personalized itineraries where customers get involved in the planning process. Unlike Wanderlog, WanderWise enables travelers to prioritize attractions, set precise start and end times, and include individual visit durations, ensuring realistic and enjoyable itineraries.

Testing of WanderWise using simulated scenarios demonstrated clear improvements in itinerary efficiency. However, we acknowledge current limitations, including a fixed set of attractions and constant average travel speeds. Future enhancements could integrate real-time traffic data, allow users to freely add new locations, and factor in personal preferences like dining or rest breaks.

Ultimately, adopting a solution like WanderWise would significantly boost Wanderlog's premium membership offering, improve user satisfaction, and promote stronger customer loyalty, making the app not just convenient but truly personalized for travelers.

This report provides an in-depth analysis of the proposed solution, including the formulation of the optimization model such as parameters, decision variables, objective function and constraints and its practical applications in addressing current limitations in itinerary planning.

1. Introduction

In this digital age, travel planning has become increasingly convenient with the rise of travel planner apps (Market.us, 2023). These travel planner applications such as Skyscanner, streamline flight bookings, while travel planning tools like Wanderlog simplify itinerary planning. As a result, travel planning has become increasingly convenient for travellers.

Travel agencies, once a staple for convenient trip planning, are slowly being phased out as people seek more personalized and unique experiences tailored to their personal preferences rather than generic itineraries based on common tourist attractions. In the following sections, we will explore Wanderlog, and identify potential business opportunities based on its limitations.

2. About Wanderlog

Founded in 2019, Wanderlog simplifies the trip planning process with features like automated itinerary creation, collaboration between users and attraction information display, all in a single user-friendly interface. In the app store, Wanderlog is currently rated 4.9 stars out of 22.6k reviews, showing how highly regarded it is.

Wanderlog - Travel Planner 4+
Itinerary & Road Trip Guide
Road Trip and Travel Planner LLC
Designed for iPad
#78 in Travel
★★★★★ 4.9 • 22.6K Ratings
Free · Offers In-App Purchases

2.1 Existing Services

Travellers can choose to manually create their itinerary by selecting destinations from a library of user-published guides or app-generated recommendations. For indecisive travellers, Wanderlog auto-generates itineraries based on popular destinations at user's selected location.

Wanderlog then calculates the travel time between destinations and provides essential details, such as operating hours and Google reviews for each location. With Wanderlog's premium membership, users can also optimize the routes based on the user's selected destinations, minimising the total travel time through intelligent route planning.

3. Problem Description

Despite the positive reviews of the app, we have identified some limitations of the application.

1. **Limited Route Customization in Optimization Feature:** Currently, the Wanderlog optimization feature focuses on saving time and fuel by selecting the shortest route. However, it does not take note of customer preferences for certain locations, always defaulting to the quickest path instead.

2. **Lack of Hourly Start-End Timing:** Currently, Wanderlog's itinerary planner does not allow users to specify the time they want to start their trip, nor does it let them input the estimated time spent at each destination. As a result, this could lead to unintended issues, such as arriving after closing hours or spending excessive travel time due to poorly chosen, geographically scattered locations.

The current limitations of Wanderlog's itinerary building system stem from its inability to account for key real-world constraints like start time and visit durations.

---

4. Business Opportunity

Currently, Wanderlog earns a small percentage of its revenue from bookings made through the app, as well as from its paid premium membership. However, the launch of Wanderlog's premium membership features, specially the optimization feature, has received negative feedback, with users pointing out that the premium tier fails to address key inefficiencies.

Therefore, **our main objective of this project** is to develop an enhanced Network Optimization Model that addresses these limitations. This model will generate more desirable itineraries, strengthening the value proposition of its premium membership, ultimately increase subscription conversion rates, reduce customer churn and potentially improve the company's long-term sustainability and profitability.

---

5. Proposed Solution: WanderWise

To address these limitations, we develop **WanderWise**, a Travel Itinerary Optimization System that leverages mathematical programming using the Gurobi solver, and comprises two building blocks: OpenStreetMap and Linear Programming Model.

5.1. OpenStreetMap

To create our linear programming model, the most important data needed was the distance between each pair of locations. However, we were unable to find an existing dataset containing this information. To overcome this, we used **OpenStreetMap API** to calculate the distances between locations based on their longitude and latitude coordinates.

5.2. Linear Programming Model

Our optimization problem can be classified as a **Prize Collecting Travelling Salesman Problem** (Bienstock, Goemans, Simchi-Levi, & Williamson, 1993).

Unlike the classical Traveling Salesman Problem, our model allows users to specify their preferences by assigning priority rankings (with 1 being the highest), as well as input their desired start and end times, along with visit durations for each destination, respectively.

We made several assumptions to facilitate the development of our model and reduce its complexity, while still ensuring that the key limitations of the original system are appropriately addressed. These assumptions include:

Assumptions	Description
Fixed Start and End Location	The itinerary begins and ends at a fixed location (e.g. Marina Bay Sands), simulating how a tourist staying there can plan their day.
Accurate Visit Duration per Attraction	We assume users know the exact time they will spend at each site, with no partial or extended stays beyond the specified preferences.
Single Average Travel Speed and Shortest Paths	We assume a constant travel speed of 800 meters/min, with no modeling of real-world traffic variability.
No Partial Visits or Revisited Sites	Attractions can only be fully visited (1) or skipped (0). Once visited, attractions are not revisited in the same itinerary.
Sufficiently Large Big-M Value	When enforcing conditional constraints (e.g., operating hours apply only if an attraction is visited), a large constant M is used to "relax" constraints when necessary, ensuring it's large enough to avoid incorrectly constraining the solution.
Single User or Homogeneous Group	We assume a single traveler's perspective or a group making unified decisions.
Perfect Information	We assume road network data, attraction details, and operating hours are accurate.
No Additional Activities or Breaks	Meal times, rest breaks, and unscheduled stops are not included in our features. Users can adjust visit durations or route times to approximate such needs.

Below is an image of the initial user interface of WanderWise. Users will input the necessary information, and a step-by-step demonstration of WanderWise will be provided in **Section 7**.

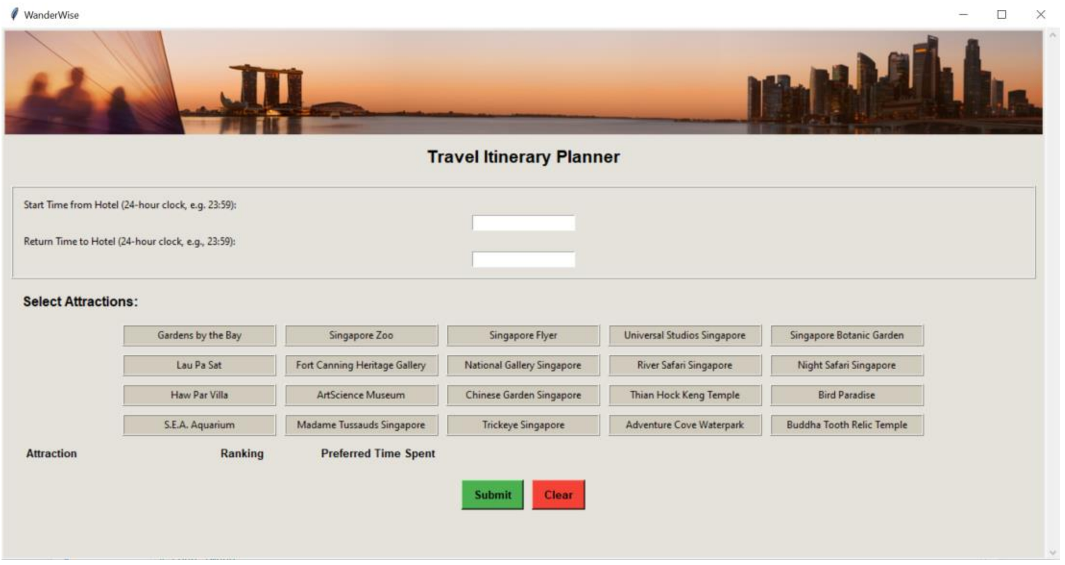


Fig 1: *The GUI of WanderWise*

---

6. Formulation of the LOP Model

6.1 Parameters

1. let N be the set of nodes, $N = \{hotel\} \cup \{attraction_1, attraction_2, \ldots, attraction_n\}$
2. let d_{ij} be the travel distance (in meters) node i and node j
3. let $t_{ij} = \frac{d_{i,j}}{average\ speed}$, here we assume a constant average speed of 800m/minute
4. let Oi be the opening time of attraction i (calculated in minutes from midnight)
5. let Ci be the closing time of attraction i (calculated in minutes from midnight)
6. let p_i be the preferred visit duration of attraction i
7. let S_i be the start visiting time of attraction i (in minutes from midnight)
8. let r_i be the interest rank of node i(lower rank indicates higher interest)
9. let b_i be the bonus of node i, and it is calculated with the following equation:
$$b_i = 100 \times (r_{max} - r_i + 1)$$
where r_{max} is the maximum rank among the attractions. This system is set up to ensure that the system prioritizes the higher-ranking attractions.
10. let D be the departure time from the hotel (in minutes from midnight)
11. let R be the return time back to the hotel (in minutes from midnight)
12. let M be the big M constant, here we fix M to be 10000
13. let λ be a constant that penalize each additional distance needs to be travelled, here we Set λ to be 0.01. The reason behind this number is because the distance between each attraction ranges from 4 to 5-digit numbers while the bonus point ranges from 3 to 4 digit, therefore to ensure a proportional tradeoff between the two an additional distance penalty of 0.01 is set

6.2 Decision Variables

1. Let y_i be a binary variable to indicate whether attraction in node i is visited in the tour, where $y_i = 1$ if node i is visited during the tour and $y_i = 0$ if not visited.
2. Let $X_{i,j}$ be a binary variable to indicate the path from node i to node j, where $X_{i,j} = 1$ if the path is selected and $X_{i,j} = 0$ if not selected.

6.3 Objective Function

The objective of this LOP is to maximize the satisfaction level of the user, indicated by the bonus point, while minimizing the travelling distance.

$$\max \sum_{i \in N\ \{depot\}} (b_i \times y_i) - \lambda \sum_{(i,j):i \neq j} d_{i,j} \times X_{i,j}$$

6.4 Constraints

1. Path Continuation Constraints

Constraint	Description
$\sum_{j \neq hotel} X_{hotel,j} = 1$	The trip starts from hotel to location *j*, this ensures the path continues to exactly one node
$\sum_{i \neq hotel} X_{i,hotel} = 1$	The trip ends at the hotel from location *i*, this ensures the path ends at the hotel from exactly one node
$\sum_{i \neq j} X_{ji} = y_i \quad , \forall i \neq hotel$	This ensures that if attraction *i* is visited ($y_i = 1$), the sum of path coming out from node *i* must exactly be 1. Conversely, if attraction *i* is not visited ($y_i = 0$), then no path should leave node *i*.
$\sum X_{j,i} = y_i$ $i \neq j, \forall i \neq hotel$	This ensures that if attraction *i* is visited ($y_i = 1$), the sum of path coming into node *i* must exactly be 1. Conversely, if attraction *i* is not visited ($y_i = 0$), then no path should enter node *i*.
$X_{i,j} \leq y_i$, $\forall (i, j), i \neq j$	If path from node *i* to node *j* is used, then it forces y_i to be equal to 1, that is node *i* must be visited
$X_{i,j} \leq y_j$, $\forall (i, j), i \neq j$	If path from node *i* to node *j* is used, then it forces y_j to be equal to 1, that is node *j* must be visited

2. Time Ordering Constraints

Constraint	Description
$S_j \geq S_i + p_i + t_{i,j} - M(1 - X_{i,j})$, $\forall (i, j)$ where $j \neq hotel$	If path from *i* to *j* is used, this ensures that the start visiting time of node *j* is later than the sum of the start visiting time of node *i*, preferred visiting duration of node *i*, and the travel time from node *i* to node *j*. If the path is not used the big M constant will relax the constraint.
$Si + pi + ti{,}hotel \leq R + M(1 - Xi{,}hotel)$, $\forall (i, j)$	This ensures that the trip finishes before the required return time. If the last visited location is node I, this constraint will ensure that the sum of the start visiting time at that location, the preferred visiting duration and the time to travel back to the hotel is still within the desired return time. If arc $X_{i,ot}$ is not used big M constant will relax the constraint.

3. **Operating Hours Constraints**

Constraint	Description
$S_i \geq O_i - M(1 - y_i)$, $\forall i \in \{1,2, \ldots , n\}$	If attraction *i* is visited, the starting visit time of location *i* must be after the opening hour. If attraction *i* is not visited big-M constraint will relaxes the constraint.
$S_i + p_i \geq C_i - M(1 - y_i)$, $\forall i \in \{1,2, \ldots , n\}$	If attraction *i* is visited, the total of the starting time and the preferred visit duration must be before the closing hour. If attraction *i* is not visited big-M constraint will relaxes the constraint.

---

7. Simulation and Results

Upon completing the model, we ran the model on two simulated cases – a simplified version and an extensive version – to obtain some insights on the implementation of the model.

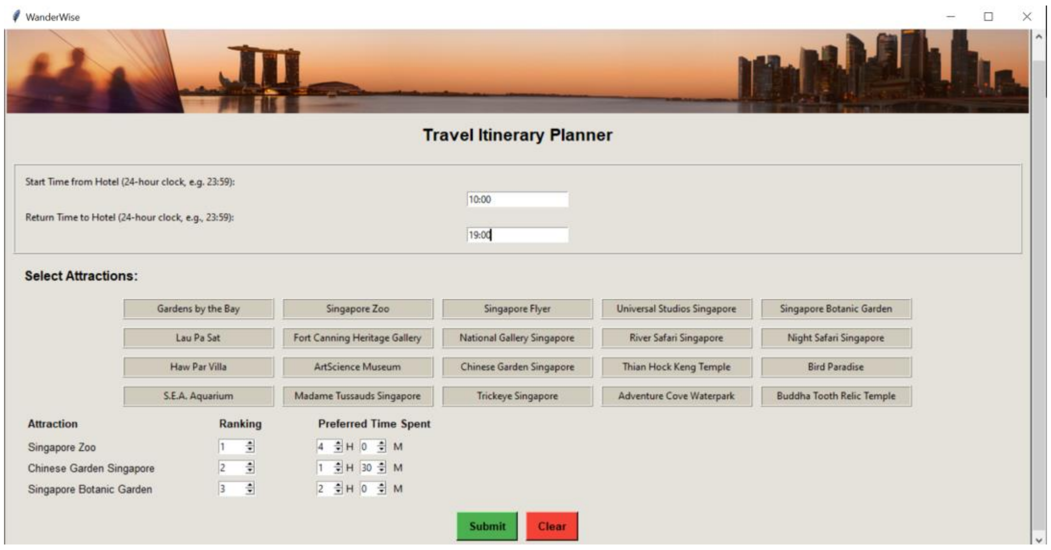
After the user inputs the necessary information in WanderWise, the linear programming model will aim to maximize user's satisfaction based on their ranking preferences, while minimizing total distance travelled. The final output will include a list of visited attractions in order, each with exact start and end times, as well as the total travel path, mapped on a visual interface.

7.1 Simulation Cases - Simplified Case

The user, a tourist staying at Marina Bay Sands, is planning a sightseeing itinerary in Singapore for the following day. The user is available from **10 AM to 7 PM**, and has **identified three attractions** to visit. Ranked by priority, the attractions are:

1. **Singapore Zoo** – Preferred visit Duration: 4 hours
2. **Chinese Garden Singapore** – Preferred visit Duration: 1 hour and 30 minutes
3. **Singapore Botanic Garden** – Preferred visit Duration: 2 hours

Figure 2 is a screenshot of the GUI interface after the user has provided the necessary inputs:



Travel Itinerary Planner

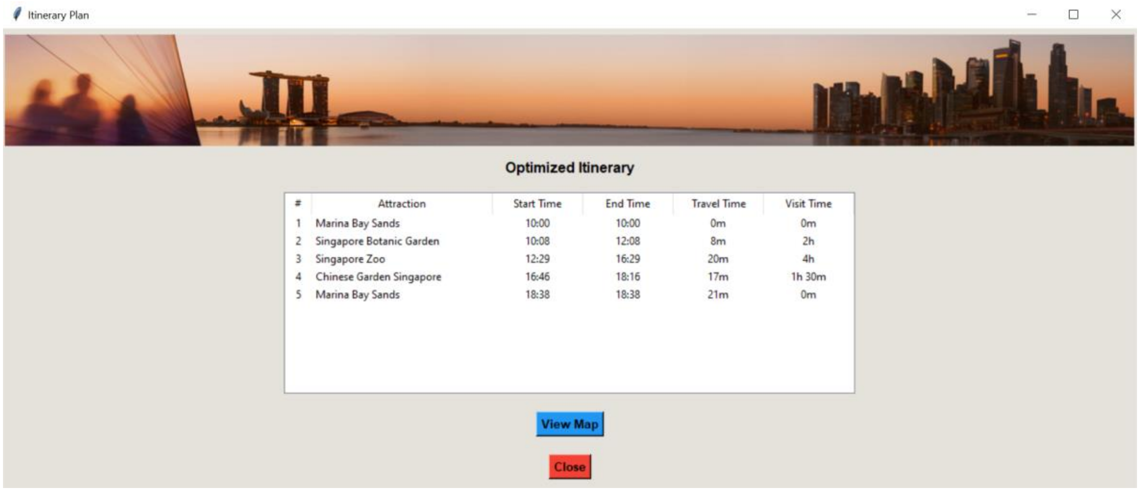
Start Time from Hotel (24-hour clock, e.g. 23:59): 10:00

Return Time to Hotel (24-hour clock, e.g., 23:59): 19:00

Attraction	Ranking	Preferred Time Spent
Singapore Zoo	1	4 H 0 M
Chinese Garden Singapore	2	1 H 30 M
Singapore Botanic Garden	3	2 H 0 M

Fig 2: Screenshot of Simulation Step 1

After entering these inputs, the user clicks the "Submit" button to generate the optimized route. After the optimized route is generated, the sequence of locations — along with the start time, end time, travel time to the next location, and the user's preferred visit duration for each attraction is displayed in Figure 3. Additionally, there is a "View Map" option that allows users to view a more detailed visualization of the optimized route mapped out on a Singapore map.



#	Attraction	Start Time	End Time	Travel Time	Visit Time
1	Marina Bay Sands	10:00	10:00	0m	0m
2	Singapore Botanic Garden	10:08	12:08	8m	2h
3	Singapore Zoo	12:29	16:29	20m	4h
4	Chinese Garden Singapore	16:46	18:16	17m	1h 30m
5	Marina Bay Sands	18:38	18:38	21m	0m

Fig 3: Screenshot of Simulation Step 2

This is what the map looks like, as shown in Figure 4. The red marker with a white star represents the starting and ending point at Marina Bay Sands. The coloured markers with numbers represent the attractions in the order they are to be visited, while the corresponding coloured lines represent the travel paths between each attraction.

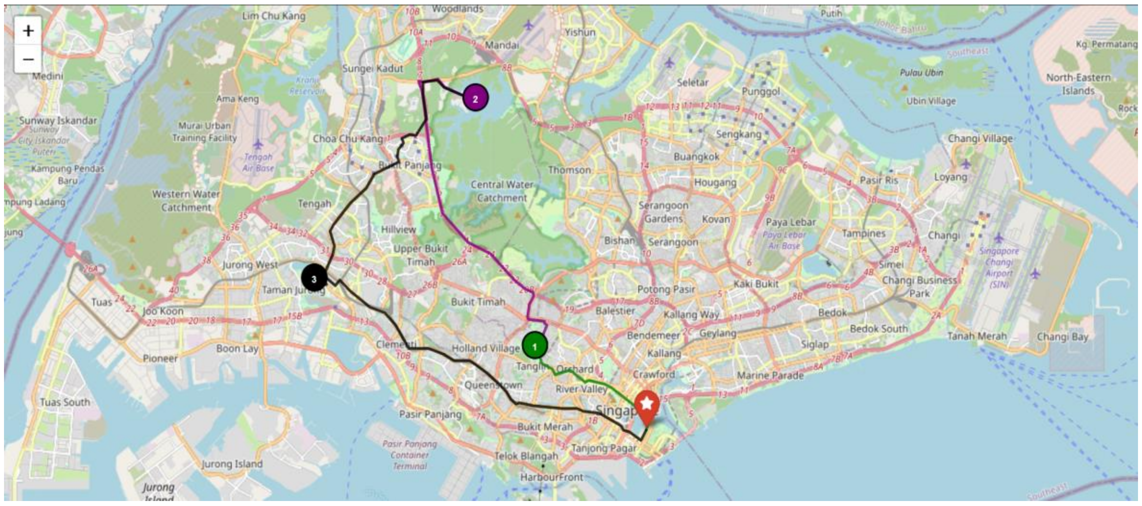


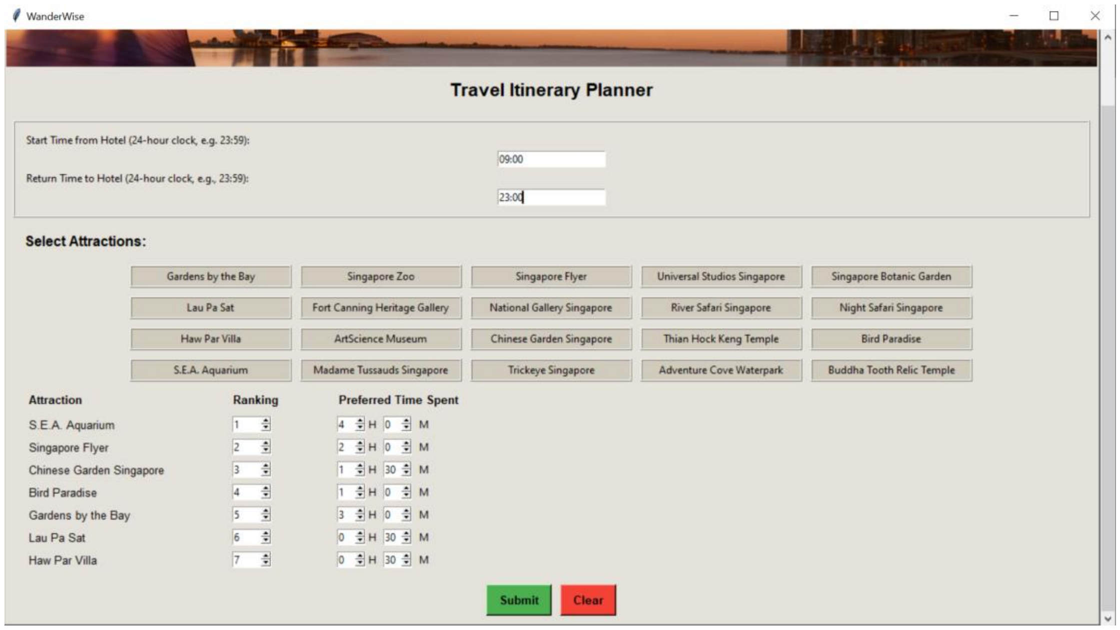
Fig 4: Screenshot of Simulation Step 3

7.2 Simulation Cases - Extensive

Moving to a more complex case: Suppose the user is available from **9 AM to 11 PM**, and has **identified seven attractions** they would like to visit. Ranked by priority, the attractions are:

1. S.E.A. Aquarium – Preferred visit Duration: 4 hours
2. Singapore Flyer – Preferred visit Duration: 2 hours
3. Chinese Garden Singapore – Preferred visit Duration: 1 hour 30 minutes
4. Bird Paradise – Preferred visit Duration: 1 hour
5. Gardens by the Bay – Preferred visit Duration: 3 hours
6. Lau Pa Sat – Preferred visit Duration: 30 minutes
7. Haw Par Villa – Preferred visit Duration: 30 minutes

Figure 5 is a screenshot of the GUI interface after the user has provided the necessary inputs:



Travel Itinerary Planner

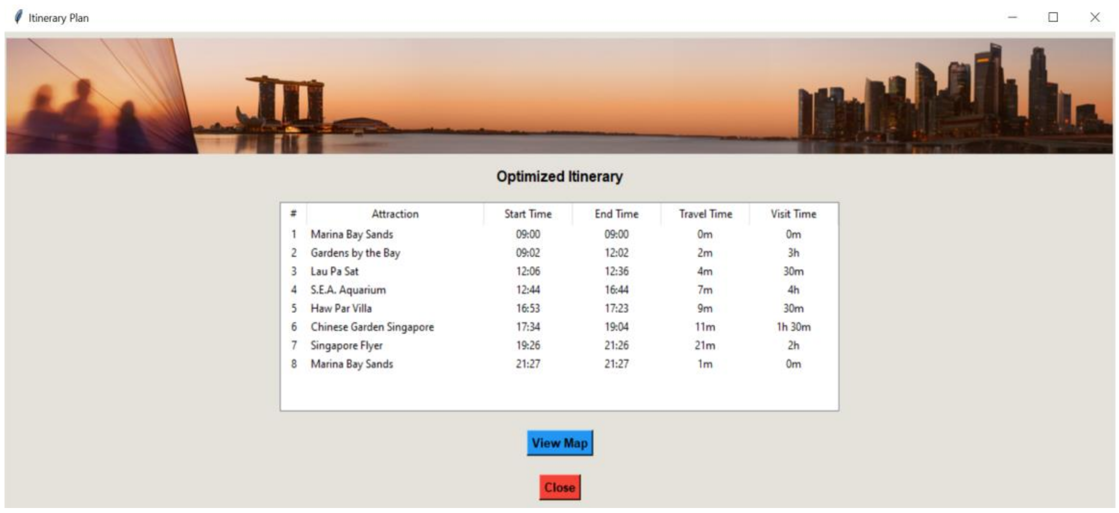
Start Time from Hotel (24-hour clock, e.g. 23:59): 09:00

Return Time to Hotel (24-hour clock, e.g., 23:59): 23:00

Attraction	Ranking	Preferred Time Spent
S.E.A. Aquarium	1	4 H 0 M
Singapore Flyer	2	2 H 0 M
Chinese Garden Singapore	3	1 H 30 M
Bird Paradise	4	1 H 0 M
Gardens by the Bay	5	3 H 0 M
Lau Pa Sat	6	0 H 30 M
Haw Par Villa	7	0 H 30 M

Fig 5: Screenshot of Simulation Step 1

After entering these inputs, the user clicks the "Submit" button to generate the optimized route, as shown in Figure 6.



Optimized Itinerary

#	Attraction	Start Time	End Time	Travel Time	Visit Time
1	Marina Bay Sands	09:00	09:00	0m	0m
2	Gardens by the Bay	09:02	12:02	2m	3h
3	Lau Pa Sat	12:06	12:36	4m	30m
4	S.E.A. Aquarium	12:44	16:44	7m	4h
5	Haw Par Villa	16:53	17:23	9m	30m
6	Chinese Garden Singapore	17:34	19:04	11m	1h 30m
7	Singapore Flyer	19:26	21:26	21m	2h
8	Marina Bay Sands	21:27	21:27	1m	0m

Fig 6: Screenshot of Simulation Step 2

This is what the map looks like, as shown in Figure 7.

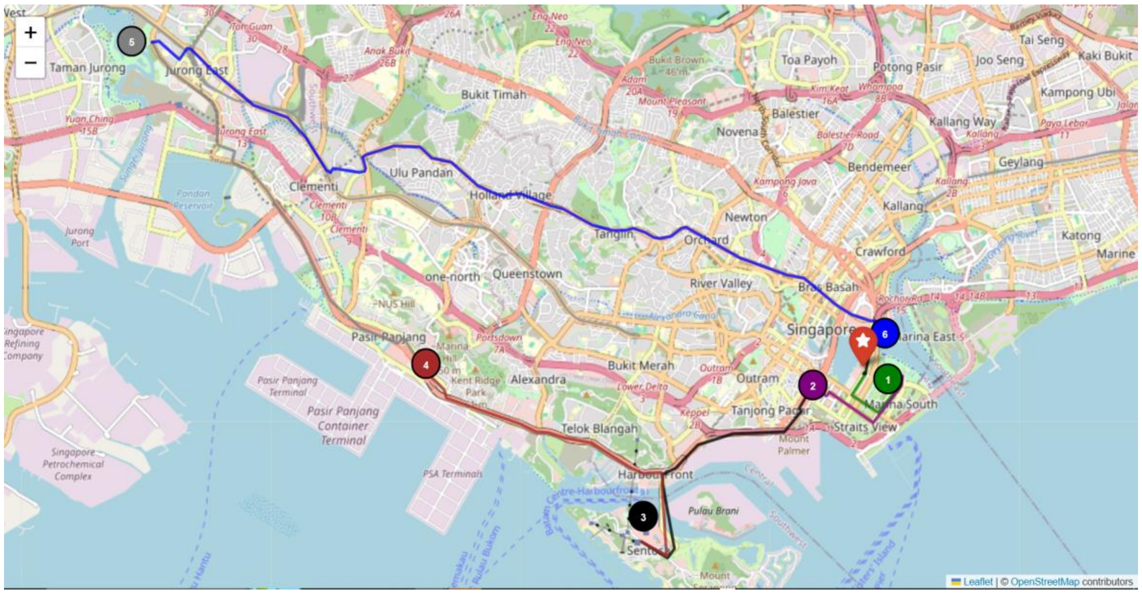


Fig 7: Screenshot of Simulation Step 3

Based on our observations: The optimized itinerary included six out of the seven selected attractions within the user's available time of 9 AM to 11 PM. The user will visit Gardens by the Bay, Lau Pa Sat, S.E.A. Aquarium, Haw Par Villa, Chinese Garden Singapore, and Singapore Flyer, before returning to Marina Bay Sands. Bird Paradise was excluded from the optimized itinerary for the day due to time constraints.

8. Limitations

8.1 Fixed Attractions

In our model, users are unable to enter a new location beyond those shown in the list of attractions, as the location data is extracted from a CSV file populated manually by us. The number of locations is limited compared to the extensive location databases used by established travel planner apps.

Initially, we considered allowing users to input the longitude and latitude of new locations, but this idea was abandoned as it was time-consuming for users and defeats the purpose of having a smart itinerary planner. That said, there is definitely room for improvement by integrating a location-based API such as Google Geocoding API. Whenever a user inputs a new location, the API will retrieve the corresponding latitude and longitude of this location, which will be fed into our OpenStreetAPI to determine the distances between locations, and into our LOP model for optimization.

8.2 Fixed Speed

In our model, the average speed is fixed and calculated based on typical traffic conditions. However, it does not account for situations where traffic might be better or worse than expected. Improvements could be made by considering both pessimistic and optimistic average timings, which can then be presented to users, allowing them to make more informed choices when planning their trips.

Another solution can be integrating a real-time traffic API, like Google Maps. This would allow the system to fetch current traffic data and update the travel times accordingly.

---

9. Future Implementation

To improve the model, we can allow users to also input their preferred dining timings as well as specific locations where they would like to dine in. This would make the itinerary more personalized and realistic, especially for users who prioritize their food experiences over sight-seeing during their trips. However, this was not included in our model due to assumption 8, which states that dining preferences and timings are not considered in the itinerary planning.

---

10. Conclusion

With the concept of our Travel Itinerary Optimization System, WanderWise. We demonstrated that it is possible to address the limitations of Wanderlog by striking a balance between user preferences and shortest travel paths. This approach improves the optimization features currently offered by Wanderlog's premium membership — ultimately leading to increased user engagement, higher satisfaction rate, and better premium membership retention rate.

Applying Optimization technique concepts to a real-world challenge was a valuable and insightful learning experience. Although we may not have the opportunity to formally propose this idea to Wanderlog, we hope that more travel planning app companies will adopt similar approaches in their products to create smarter, more user-centric travel solutions.

11. References

Market.us. (2023, July 21). *Navigating the future: The travel planner app market reach USD 1,455.1 billion, globally by 2032*. LinkedIn. https://www.linkedin.com/pulse/navigating-future-travel-planner-app-market-reach-usd-14551/

Ruangkanjanases, A., & Kaoian, P. (2018). Travel planning application: Combining linear programming and shortest route problem to optimize travelers' satisfactions. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10*(3–2), 13–19. https://jtec.utem.edu.my/jtec/article/view/4705

Zelst, M. van. (2016, February 1). *Personalised city trip itinerary using integer linear programming*. Trifork Blog. https://trifork.nl/blog/personalised-city-trip-itinerary-using-integer-linear-programming/

Bienstock, D., Goemans, M., Simchi-Levi, D., & Williamson, D. (1993). A note on the structure of LP-relaxations of the Steiner tree problem. *Mathematical Programming*, *59*(1–3), 413–420. https://doi.org/10.1007/BF01581298

Wanderlog**.** (n.d.). *Optimize route*. Wanderlog Help Center. https://help.wanderlog.com/hc/en-us/articles/13545624787867-Optimize-route