

Table of Contents

1. Project overview.....	1
1.1. Introduction.....	1
1.2. Problem description	1
2. Optimisation model	1
2.1. Mathematical formulation.....	1
2.2. Decision variables	1
2.3. Objective function.....	1
2.4. Constraints	2
3. Data collection & reprocessing	2
3.1. City dataset.....	2
3.2. Route dataset.....	2
3.3. Data cleaning & filtering	2
4. Integration of TSP and DFS.....	3
4.1. Integration of TSP	3
4.2. Integration of DFS	3
5. Results & Analysis	3
5.1. Optimised itinerary	3
6. Prototype usage & instructions	4
6.1. User input.....	4
6.2. Running the model.....	4
7. Conclusion & future works.....	4
7.1. Summary of findings.....	4
7.2. Potential improvements	4
8. Appendix.....	5

1. Project overview

1.1. Introduction

One of the advantages of studying at LBS, aside from the stellar courses learnt, such as the Decision Analytics & Modelling course, is the proximity to Europe, which allows students to explore this continent in their free time. However travel planning can be a complex process, especially when visiting multiple cities with varying costs, transportation options, and accommodation expenses. With the increasing availability of transportation and accommodation data, optimisation techniques can help travellers find the most cost-effective and efficient itinerary.

This project aims to create a model that optimises multi-city European travelling, striking a balance between key factors, from transportation costs to accommodation expenses, travel duration, and most importantly personal preferences. By using optimisation algorithms, the model seeks to generate an optimal itinerary to meet the traveller's preferences and constraints.

1.2. Problem description

The main challenge of this project lies in solving a selective Travelling Salesman Problem (TSP) where a traveller must visit a subset of cities, based on given constraints, all while minimising expenses. Unlike a traditional TSP, where every city must be visited, this problem allows for flexibility in the number of city visited, introducing several constraints such as budget limitations, time constraints, transportation modes, destination and lodging preferences, and route feasibility.

By tackling these constraints, the project provides a structured and optimised itinerary that minimises the traveller's expenses while fulfilling their specific requirements. The model is implemented using an Integer Linear Programming (ILP) with additional heuristics for route optimisation and constraint handling.

2. Optimisation model

2.1. Mathematical formulation

At the core of the optimisation model is the set of cities defined by C , as well as different parameters, such as lodging costs ($lodging[i]$), travel costs ($cost[i,j]$), total number of days (T), and budget (β).

The objective is to find a closed-loop itinerary that starts and ends in the same city while minimising overall expenses.

Mathematically, the problem is structured as follows:

- A binary decision variable $x_{i,j}$ determines whether the route between cities i and j is used
- Another binary variable y_i determines whether city i is going to be visited
- A continuous variable s_i defining the number of nights spent in each city
- The total budget constraint incorporates both travel and lodging costs

2.2. Decision variables

The following variables were defined:

- $x_{i,j} \in \{0,1\}$ $x_{i,j} = 1$ if the solution includes a direct travel between cities i and j , 0 otherwise
- $y_{i,j} \in \{0,1\}$ $y_{i,j} = 1$ if city i is visited, 0 otherwise
- $s_i \in \mathbb{Z} \geq 0$ represents the number of nights spent in city i

These variables interact to ensure that the model not only finds a feasible route but also adheres to user preferences such as budget constraints, minimum number of cities to visit, and accommodation type.

2.3. Objective function

The primary goal is to minimise total travel and lodging costs, which is formulated as:

$$\min \left(\sum_{(i,j) \in A} cost[i,j] \times x_{i,j} + \sum_{i \in C} lodging[i] \times s_i \right)$$

Where $cost[i,j]$ represents the travel cost between cities i and j

Lodging[i] represents the accommodation cost per night in city i

By optimising this function, the model ensures that the traveller selects routes and accommodations that yield the lowest total cost while still covering the required destinations.

2.4. Constraints

To ensure a realistic and practical travel plan, the model imposes several constraints, including:

- 1- Closed loop constraint: the trip must start and end in a predefined city

$$y_s = 1$$

Where s is the starting city

- 2- Flow conservation constraint: if a city is visited, exactly one travel arc must enter and leave it

$$\sum_{k \neq i} x_{k,i} = y_i, \quad \sum_{k \neq i} x_{i,k} = y_i, \quad \forall i \neq s$$

- 3- No travel without visiting a city: a city cannot be part of the route unless it is selected

$$x_{i,j} \leq y_i, \quad x_{i,j} \leq y_j, \quad \forall i \neq j$$

- 4- Minimum and maximum number of cities: ensures the user visits at least the required number of cities

$$\sum_{i \in C} y_i \geq \text{min_cities}$$

- 5- Budget constraint: the total cost must not exceed the user's budget

$$\sum_{(i,j) \in A} \text{cost}[i,j] \times x_{i,j} + \sum_{i \in C} \text{lodging}[i] \times y_i \leq \beta$$

- 6- Diversity constraint: to avoid spending too many nights in a single city

$$y_i \leq \alpha \times T, \quad \forall i$$

Where α is a fraction regulating the maximum stay, and T is the total number of days

- 7- Subtour elimination: ensures that the trip remains a single, coherent tour without isolated sub-loops

$$\sum_{i,j \in S} x_{i,j} \leq |S| - 1, \quad \forall S \subset C, \quad s \notin S$$

3. Data collection & reprocessing

3.1. City dataset

The city dataset serves as foundation for the model, containing the information of each city that can influence travel decision, including the city's name, latitude and longitude, lodging costs, temperature, tourism preferences (binary indicators for whether the city has a beach or is near mountains), and daily budget estimate.

3.2. Route dataset

The route dataset provides information about possible travel connections between cities. Each row in the dataset represents a direct route and contains the route's origin and destination, travel costs (flights, trains and buses), travel time, and available transport modes.

For routes where multiple options exist, the model dynamically selects the cheapest and fastest feasible mode, unless the user specifies a preference.

3.3. Data cleaning & filtering

Since raw data often contains inconsistencies, multiple preprocessing steps were performed:

Handling missing data:

- If a travel mode (flight, train, bus) was missing from a missing route, it was imputed using an estimated value from similar city pairs

- Missing lodging costs were interpolated based on neighbouring cities within the same country

Removing unrealistic routes:

- Routes with excessive travel times (e.g., bus journeys over 30 hours) were removed
- Cities that had no reasonable transport connections were excluded from the dataset

Currency & standardisation:

- All cost data was converted to Euro for consistency
- Time data was rounded to the nearest 10 minutes to avoid precision errors

Indexing & formatting:

- Cities were indexed numerically for faster computation
- Routes were sorted as dictionary lookups to improve search efficiency within the model

By applying these steps, the dataset remained accurate and computationally efficient.

4. Integration of TSP and DFS

4.1. Integration of TSP

In the model, a selective TSP was used, where instead of visiting all cities, an optimal subset based on budget, temperature, and user preferences was selected. The optimisation ensures that the selected cities form a valid, cost-efficient travel loop while adhering to user-defined constraints, such as minimum and maximum trip duration and lodging budgets. By leveraging TSP principles, the model guarantees that no unnecessary routes are included, therefore reducing overall travel costs.

4.2. Integration of DFS

To prevent subtours (disconnected loops within the itinerary), a Depth First Search (DFS) was used to ensure that the selected cities form a single and connected travel route. When the initial solution includes multiple disjoint cycles, DFS is applied to detect these subtours, and lazy constraints are introduced dynamically to force all cities into a unified path. This process is repeated iteratively until the model guarantees a single, feasible, and logically structured itinerary, avoiding fragmented travel plans that could lead to inefficiencies or infeasibility in real-world scenarios.






5. Results & Analysis

5.1. Optimised itinerary

The model computed a 10-day, well-rounded trip across Europe that combines different landscapes (beach and mountain), weather, and regions, to allow the traveller to discover as much as possible during the trip.

The selected cities to visit were Paris, Geneva, Athens, Amsterdam, and Berlin, in that order. The complete breakdown is showed in Table 1 below.

Table 1. Optimal 10-day Europe Trip Breakdown

City	Days spent	Daily expenses (€)	Transport time
Paris 	1	80	1hr (Flight)
Geneva 	1	100	2h40min (Flight)
Athens 	2	60	3h10min (Flight)
Amsterdam 	1	80	1h20min (Flight)
Berlin 	5	70	1h30min (Flight)

The trip's total expenses amounted to €2,873, divided into travel costs (€433) and lodging costs (€2,440). To improve interpretability, the model generated an interactive map displaying the direction of travel, showing the loop that starts and ends in Paris. A screenshot of the map is in the Appendix (Figure 1), and the interactive version in the html file.

6. Prototype usage & instructions

6.1. User input

The prototype allows users to customise their trip by providing the following parameters:

- Starting city: the initial city of departure
- Total trip duration: the number of days available for travel
- Budget: the maximum amount the user is willing to spend on the trip
- Transport preferences: users can choose between flights, trains, buses, or a combination
- Minimum and maximum cities to visit: defines the trip's flexibility
- Temperature range: filters cities based on user comfort preferences
- Lodging type: users select between budget, mid-range, or luxury accommodation
- Diversity constraint: limits the maximum nights spent in a single city

6.2. Running the model

Once the user provides their inputs, the model performs the following steps:

- 1- Data filtering: cities and routes that do not match user constraints are excluded
- 2- Optimisation execution: the solver computes the best possible itinerary using Gurobi
- 3- Route verification: DFS ensures a single connected travel plan, eliminating subtours
- 4- Results generation: the final itinerary is displayed, showing the selected cities and number of nights per city, best transport mode for each leg of the trip, and total travel costs and budget breakdown

The model runs in a matter of seconds, providing an optimised, ready-to-use itinerary.

7. Conclusion & future works

7.1. Summary of findings

The optimisation model successfully generated a cost-efficient, constraint-based travel itinerary, balancing transport and lodging costs while considering user-defined preferences. By leveraging selective TSP principles and integrating DFS-based subtour elimination, the model ensures that the generated travel route is both economically viable and logistically feasible.

Key takeaways from the project include:

- Automation of trip planning, reducing the need for manual research
- Flexible itinerary generation based on user-defined constraints
- Efficient route selection, ensuring the most cost-effective and time-efficient transport modes
- Data-driven decision-making, providing users with an optimised travel plan that adapts to real-world constraints

7.2. Potential improvements

While the current model provides a strong foundation, there are several areas for future improvement:

- Real-time pricing integration: connecting to APIs for live travel costs instead of relying on static datasets
- Sustainability considerations: introducing an eco-friendly travel mode selection by factoring in carbon emissions
- Multi-user travel planning: extending the model to support group travel, where multiple users have different budget and preferences
- Dynamic adjustments: allowing users to modify parts of their itinerary post-optimisation
- User interface development: creating a web-based or mobile application for an interactive and user-friendly experience

8. Appendix

Figure 1. Screenshot of the travel route

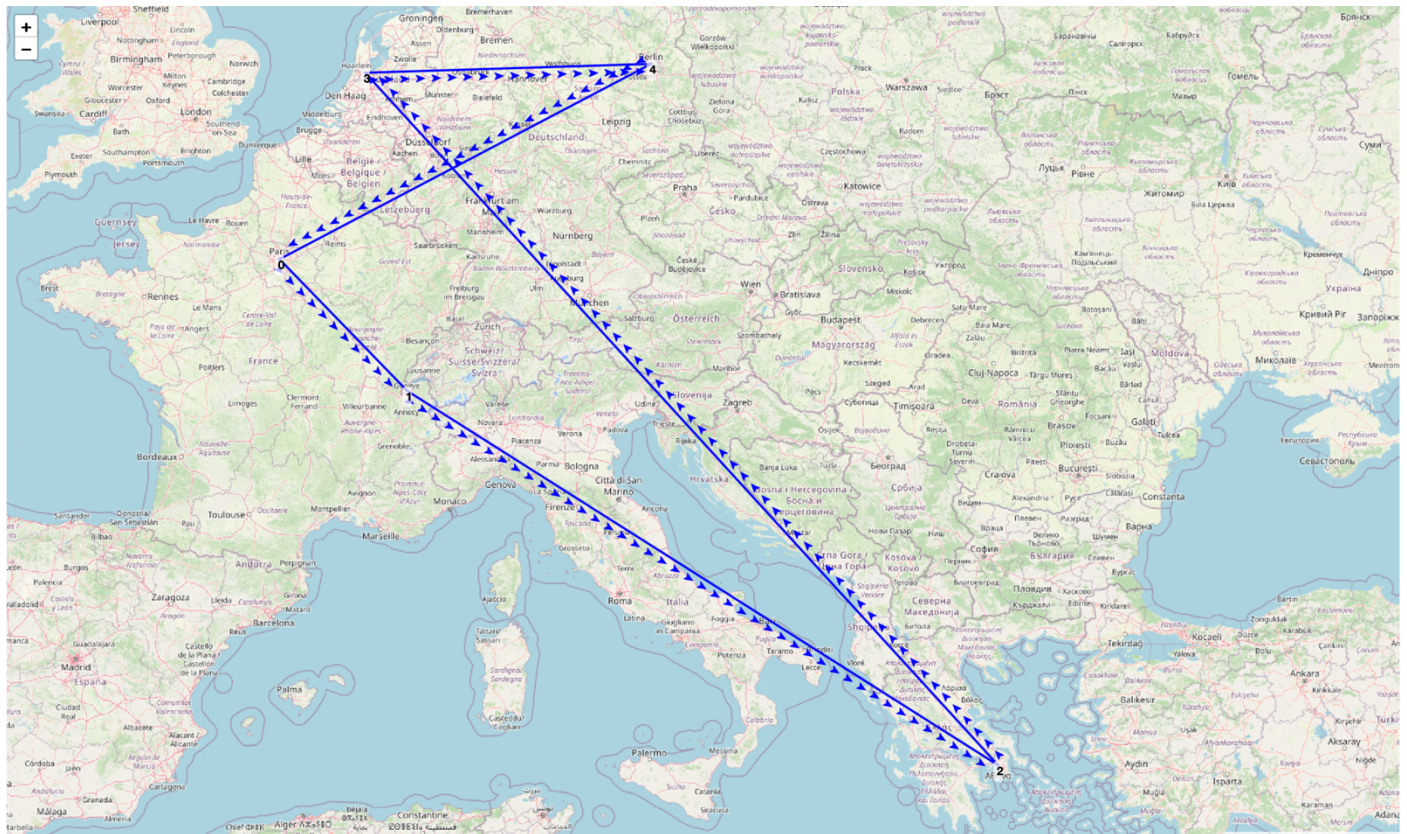


Figure 1. Optimised Europe Travel Trip

Figure 2. Sensitivity analysis

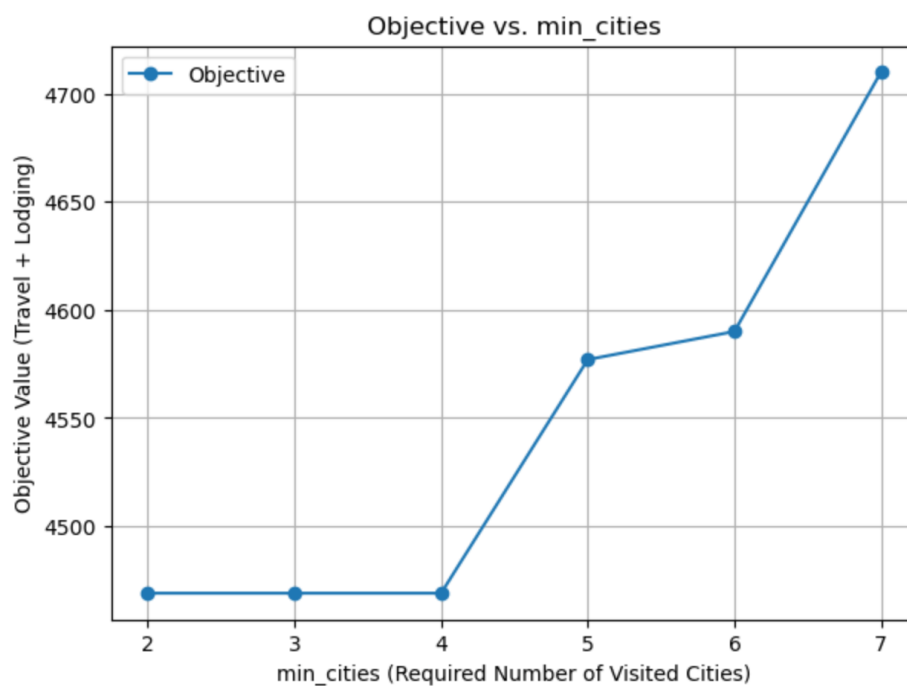


Figure 2. Sensitivity analysis