École Polytechnique fédérale de Lausanne

Decision-aid Methodologies for Transportation

Operational Research Project

Jingren Tang

1 Introduction

Unki, a startup from EPFL, has launched a tourist itinerary planning project that uses a Mixed Integer Program (MIP) to optimize travel schedules based on individual preferences and constraints. This project incorporates various elements such as hotel accommodations, activities, and dining experiences to maximize enjoyment and minimize costs. In this report, we explore mathematical modeling, heuristic approaches, and sensitivity analyses to highlight how algorithmic strategies effectively address real-world constraints and improve travel planning across different scenarios.

2 Part 1

2.1 Variables

2.1.1 Binary Variables

— x_i : Binary variable indicating whether activity i is selected in the itinerary.
  — $x_i = 1$: Activity i is selected.
  — $x_i = 0$: Activity i is not selected.

— y_{ij} : Binary variable representing the transition from activity i to activity j.
  — $y_{ij} = 1$: Transition from activity i to activity j is present.
  — $y_{ij} = 0$: No transition from activity i to activity j.

2.1.2 Continuous Variables

— t_i : Continuous variable representing the arrival time at activity i.
  — t_i : Time at which the traveler arrives at activity i.

2.1.3 Constants

— reward_i : Reward or benefit associated with selecting activity i.
— cost_i : Cost associated with participating in activity i.
— distance_{ij} : Distance or travel time between activities i and j.
— early_i : Earliest allowable start time for activity i.
— late_i : Latest allowable start time for activity i.
— *Budget* : Maximum total cost allowed for the travel itinerary.

2.2 Explanation of Constraints

No self-loop constraints

These constraints ensure that an activity cannot transition to itself :

$$y_{ii} = 0 \quad \text{for all activities } i$$

Flow conservation constraints

These constraints enforce flow conservation at each activity node :

$$\sum_j y_{ji} = \sum_j y_{ij} \quad \text{for all activities } i$$

Activity selection constraints

These constraints link the binary decision variables x_i to the transition variables y_{ij} :

$$x_i = \sum_j y_{ij} \quad \text{for all activities } i$$

Budget constraint

This constraint limits the total cost of the itinerary to be within a specified budget B :

$$\sum_i \text{cost}_i \cdot x_i + \sum_{i \neq j} \text{distance}_{ij} \cdot y_{ij} \leq B$$

Time window constraints

These constraints enforce the time windows for each activity :

$$t_i \geq \text{early}_i \quad \text{and} \quad t_i \leq \text{late}_i \quad \text{for all activities } i$$

Transition time constraints

These constraints ensure that the transition from one activity to another respects the distance between them in terms of travel time :

$$t_j \geq t_i + \text{distance}_{ij} \quad \text{whenever } y_{ij} = 1$$

Dining follows constraints

These constraints ensure that each recreational or sightseeing activity is followed by at least one dining activity :

$$\sum_{j \in \text{Dining}} y_{ij} \geq x_i \quad \text{for all recreational or sightseeing activities } i$$

One activity per category constraints

These constraints ensure diversity in the selected activities :

$$\sum_{i \in \text{Recreational}} x_i = 1 \quad \text{and} \quad \sum_{i \in \text{Sightseeing}} x_i = 1$$

Choose two dining constraints

These constraints ensure that exactly two dining activities are selected :

$$\sum_{i \in \text{Dining}} x_i = 2$$

Start from hotel and end at hotel constraints

These constraints ensure that the travel itinerary starts from and ends at the hotel :

$$\sum_j y_{\text{Start},j} = 1 \quad \text{and} \quad \sum_i y_{i,\text{Start}} = 1$$

2.3 Optimal Solution

The optimization model has been successfully solved, and the following activities have been selected for the travel itinerary :

— Activity 0 : Selected with transition to Activity 41

— Activity 41 : Selected with transition to Activity 6

— Activity 6 : Selected with transition to Activity 40

— Activity 40 : Selected with transition to Activity 18

— Activity 18 : Selected with transition to Activity 0

The variables x_i indicate whether an activity is selected, with a value of 1 indicating selection. Similarly, the variables y_{ij} indicate the transition between activities, with a value of 1 indicating a transition from activity i to activity j.

In the optimal solution :

$$\begin{aligned} x[0] = 1.0, &\quad y[0,41] = 1.0 \\ x[6] = 1.0, &\quad y[6,40] = 1.0 \\ x[18] = 1.0, &\quad y[18,0] = 1.0 \\ x[40] = 1.0, &\quad y[40,18] = 1.0 \\ x[41] = 1.0, &\quad y[41,6] = 1.0 \end{aligned}$$

Activity 0 is designated as a hotel, 41 as recreational, 6 and 18 as dining establishments, and 40 as a sightseeing location, thereby satisfying the specified criteria. This solution optimally satisfies the objective function while meeting all constraints, providing a feasible and optimal travel itinerary. As Figure 1 shows, the path begins at hotel and travel around within a relatively small scale. As the value of distance between activities is normally greater than the activity reward ant the negligible activity cost, the distance relatively dominate the decision for the choice of activities.

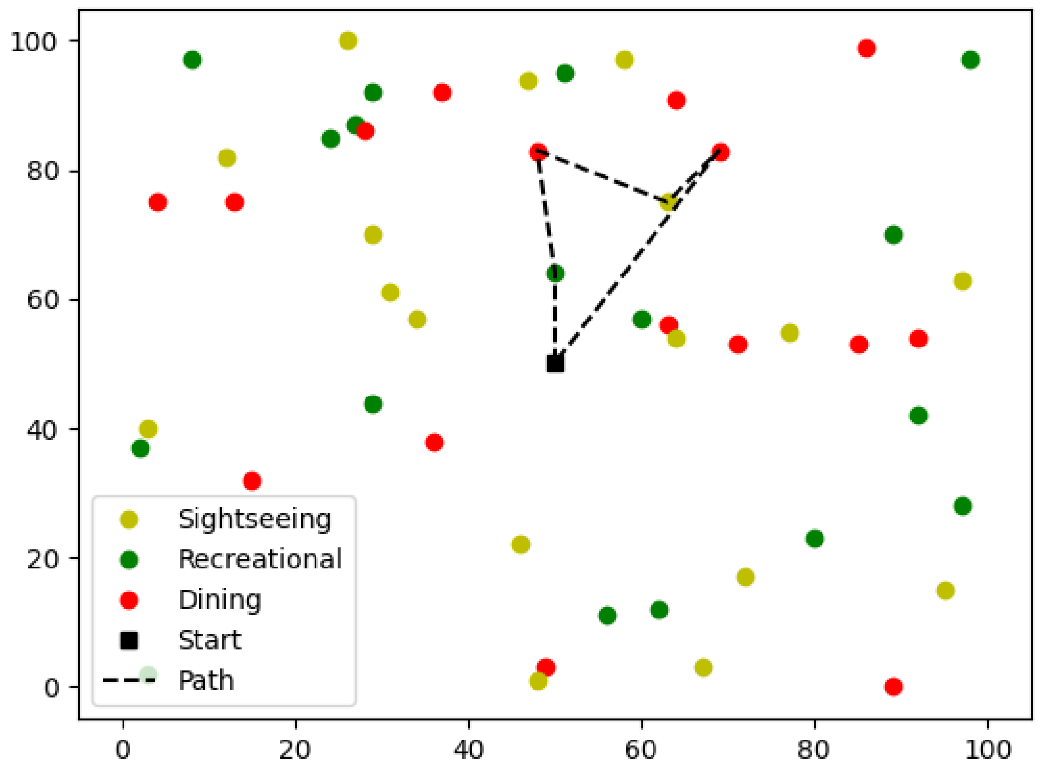


FIGURE 1 – Location of activities and tour track

3 Part 2 : Greedy Algorithm

3.1 The Reason of Choosing a Greedy Algorithm

In Part 1, we mentioned that due to the relatively large values of distances compared to other values, our initial consideration in selecting activities is the proximity of distances. As illustrated in the Figure1, we observe that the hotel is located centrally among the activities. This allows for the rapid identification of the first activity when using a greedy algorithm. Additionally, the distribution of these activities is relatively uniform, which enables the greedy algorithm to achieve relatively good results.

3.2 Operator Explanation

The heuristic itinerary algorithm is designed to optimize a travel itinerary based on a given set of constraints and preferences. The algorithm takes as input several parameters including the set of all possible points (activities), the pairwise distances between these points, the starting activity, individual activity costs, a budget limit, time constraints for each activity, and activity-specific reward values. The main goal of the algorithm is to maximize the net reward of the itinerary while adhering to budget and time constraints.

3.2.1 Initialization

The algorithm begins by initializing the current position to the start point, usually a hotel, and sets up tracking for the total cost incurred and the total reward gained. It also maintains a record of visited activities to avoid repetitions and a tally of different types of activities undertaken.

3.2.2 Activity Selection

A key component of the algorithm is the activity selection mechanism, implemented in the `select_activity` function. This function iterates through possible activities and evaluates them based on their net value (reward minus cost), while considering budget constraints and timing requirements. Activities are filtered out if they have already been visited, exceed the budget, or do not fit within the specified time window.

3.2.3 Adding Activities to the Itinerary

Once a suitable activity is selected, it is added to the itinerary in the `add_activity` function. This function updates the itinerary path, marks the activity as visited, updates the total cost and reward, and moves the current position to this new activity. Detailed cost and reward information is printed for monitoring.

3.2.4 Activity Type Determination

The `activity_type` function categorizes each activity into one of the predefined types : Dining, Recreational, or Sightseeing, which assists in ensuring a balanced itinerary in terms of activity variety.

3.2.5 Itinerary Construction

The itinerary construction involves selecting and adding activities in a sequence that starts with a main activity (either Recreational or Sightseeing), followed by a Dining activity. The process then attempts to add another main activity of a type different from the first, followed by another Dining activity. This sequence aims to balance the experience between different types of engagements.

3.2.6 Returning to Start

After all activities are selected, the algorithm concludes by potentially adding the start point (hotel) back to the path, accounting for the return journey's cost.

3.2.7 Output

The algorithm outputs the complete path of the itinerary and the net utility, calculated as the total reward minus the total cost. This output helps in evaluating the effectiveness and efficiency of the planned itinerary.

3.3 Result and Discussion

The output result is consistent with Part 1 :

— Path : [0, 41, 6, 40, 18, 0]

— Net Utility : 35.437892132289804

Using a greedy algorithm makes practical sense. When planning a trip, people tend to pick spots centered around a main area. With numerous options to choose from, the natural instinct is to go for the closest one to save time. Then, as each location is visited, the next one is considered, following the same logic of proximity. This approach is rooted in our tendency to prioritize nearby attractions first, making decision-making quick and easy, especially when excitement is high at the beginning of a trip. Greedy algorithms work by aiming for the best local solution.

However, greedy algorithms may not work well in some situations. For example, if there are several activities clustered together but far from the starting point, choosing the nearest one may not provide the most enjoyment. This is common when hotels are located far from the main attractions. Yet, in this case study, the hotel is conveniently placed among the attractions, making a greedy algorithm a great choice.

4 Part 3 : Sensitivity Analysis

In this section, we made the following adjustments :

1. Uniformly increasing all rewards.
2. Significantly increasing the rewards for several activities.
3. Reducing the budget.

4.1 Uniformly Increasing All Rewards

Here are the results :

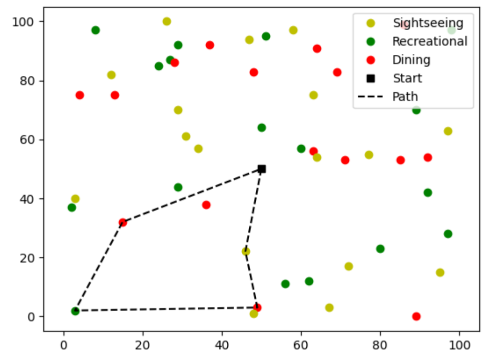
— Greedy Path : [0, 17, 3, 19, 39, 0]

— Greedy Net Reward : 165.4818254406101

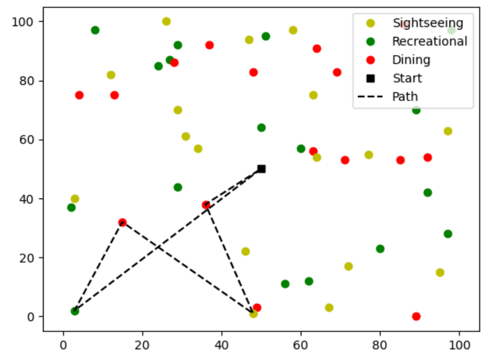
— Gurobi Tour Path : [0, 46, 9, 17, 3, 0]

— Gurobi Tour Net Reward : 179.347339315347

In the first scenario, where we doubled the rewards for all activities, we observed a certain degree of overlap in the activities selected by both algorithms. However, as figure 2a and figure 2b show, there were differences in the order in which they visited these activities. Despite this difference, the total reward accumulated by both algorithms was relatively close. This can be



(a) Big Reward Gurobi



(b) Big Reward Greedy

FIGURE 2 – Big Reward

attributed to the fact that the activities with significant overlap experienced a larger increase in reward due to our adjustments. For instance, Activity 17 saw a substantial increase in reward. When rewards become excessively large, the greedy algorithm tends to prioritize the activity with the highest reward first, as at that moment, the distance and cost factors are relatively insignificant compared to the reward. Thus, we observed that although Activity 17 was located far from the hotel, the greedy algorithm selected it as the first activity. On the other hand, the result obtained using Gurobi also included Activity 17, but it chose a more direct route. However, the advantage of using the greedy algorithm is that it allocates the most energetic time of the day to the activity with the highest reward. This ultimately leads to a minimal difference in total reward between the greedy algorithm and the Gurobi method.

4.2 Increasing the Rewards for Several Activities

Here are the results :

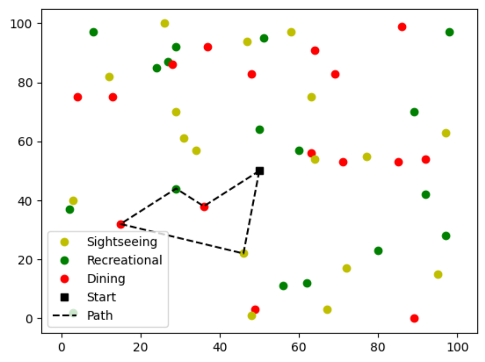
— Greedy Path : [0, 46, 39, 17, 3, 0]

— Greedy Net Reward : 223.79033844574354

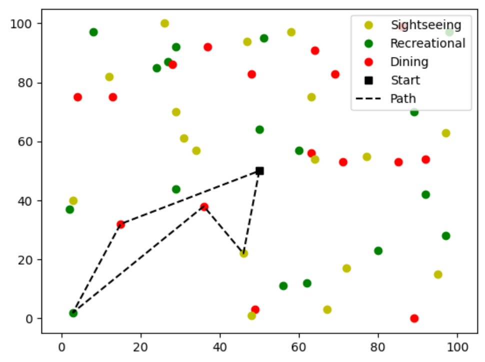
— Gurobi Path : [0, 46, 3, 26, 39, 0]

— Gurobi Net Reward : 232.12561224515045

In this scenario, the reward of activities 46, 50, 8, 1, 32 is 6 times its original reward. We observe that activities 46, 39, and 3 are selected by both algorithms. In the greedy algorithm, the first activity chosen is also activity 46. However, as figure 3a and figure 3b show, after activity 46, the greedy algorithm selects activity 39 directly to find a quicker dining option. This leads to a situation where the next destination after the meal is quite far away. This exemplifies a potential downside of greedy algorithms, where they may choose a local optimum that leads to higher costs later on. Despite this, the greedy algorithm still manages to visit many of the best locations, albeit with some detours, ensuring that it doesn't miss out on the most worthwhile destinations. Therefore, the total reward of the two algorithms is close to each other.

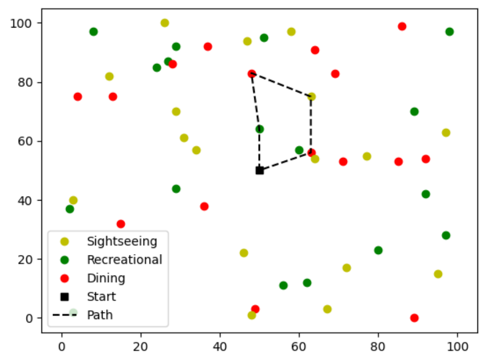


(a) Several Big Reward Gurobi

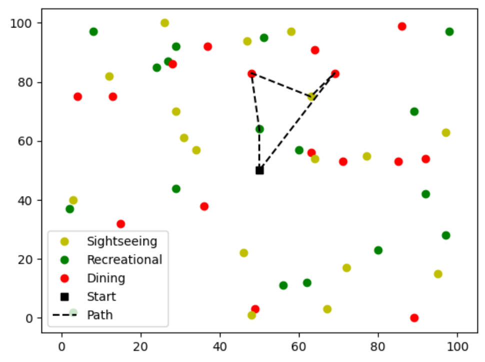


(b) Several Big Reward Greedy

FIGURE 3 – Several Big Reward



(a) Limited Budget Gurobi



(b) Limited Budget Greedy

FIGURE 4 – Limited Budget

4.3 Limited Budget

Here are the results :

— Greedy Path : [0, 41, 6, 40, 18, 0]

— Greedy Net Reward : 35.437892132289804

— Gurobi Path : [0, 41, 6, 40, 24, 0]

— Gurobi Net Reward : 22.252047176143684

In this scenario, we first halved the budget and observed no change compared to the previous results. So, we decided to further reduce the budget to one-third of its original value. As figure 4a and figure 4b show, we found that activities 40, 41, and 6 were still selected by both algorithms, with only one activity differing between them. Moreover, the order of activities in both algorithms was very similar, with only the final dining location before returning to the hotel being different. This is reasonable because when the budget is reduced, the relationships between rewards and distances remain unchanged. In other words, the desired destinations remain the same ; it's just that there's less money available now.

5 conclusion

Through the analysis presented in this study, we have demonstrated the effectiveness of mathematical modeling and heuristic algorithms in solving the complex problem of travel itinerary optimization. The use of binary and continuous variables, along with well-defined constraints in a linear programming model, allowed us to ascertain an optimal solution that adheres to budgetary, temporal, and activity variety requirements.

The application of a greedy algorithm provided a practical and intuitive approach to itinerary planning, leveraging proximity and reward metrics to make real-time decisions. Despite the simplicity of the greedy method, it showcased remarkable performance, often producing results that closely matched those of the more complex Gurobi optimizer, particularly when the geographical distribution of activities was favorable.

Sensitivity analysis revealed how changes in parameters such as rewards, budget constraints, and specific activity rewards impact the choices made by both the heuristic and optimization models. This analysis is crucial as it provides insights into the robustness of the algorithms under varying economic conditions and preferences, reflecting real-world scenarios where budget adjustments and value perceptions frequently change.

In conclusion, this study not only provided a comprehensive methodology for optimizing travel itineraries but also highlighted the adaptability and efficiency of combining various algorithmic approaches. These findings are valuable for future applications in tourism management, event planning, and other fields where such optimization is beneficial. The balance between computational efficiency and result accuracy in the heuristic approach also suggests its potential for real-time travel planning applications, making it a viable solution for dynamic and on-the-go itinerary adjustments.