Trip Planner Optimizer

Kelvin Christian

March 7, 2022

The trip planner optimizer maximize the expected satisfaction while minimizing the total distance of the trip. It uses two hybrid optimization methods: Integer Programming and Network Flow. The required data are the restaurant data and the recreational spots data.

One of the methods used is **Scaled Optimization**. Since there's no need to have huge amount of data for testing the algorithm back and forth, the fake data is generated. This small data will be used to evaluate the model result and once the result is acceptable, the model can be fed with a larger size data.

The other method that is used is **Master and Sub-Problem**.

- Master Problem: Finding the optimal route from given locations
- Sub-Problem: Choosing the potential locations that will maximize the satisfaction metric.

The master problem is one of the variation of Vehicle Routing Problem with single source. It uses network optimization to find the route with the shortest total distance. There are some restrictions that are problem specific such as:

1 On each day, the last place visited must be a restaurant (dinner)

2 Lunch time can't exceed the 3 hours from the starting trip hours

3 Total time spent during the trip musn't exceed 12 hours

The sub-problem used **Decentralized Optimization** method. It consists of two independent satisfaction optimization: Restaurants and Recreational spots. Each of the components is optimized using mixed integer programming to maximize satisfaction metric with constraint of budget, number of days, and hours spent on each locations. The satisfaction metric is quite ambigous, one potential formula for this metric S_i is the following:

$$S_i = rating_i \times log(numReviewer), \forall i \in locationSet$$

The reasoning behind this metric is the variable *ratings* represent the score for a location which is the average of ratings from a bunch of reviewer. The other metric is the *numReviewer* which is the number of reviewer. It represents the famed of a location. The argument for using this variable is people tend to consider famous places during trip. The log function is used to reduce the impact of exponentially large number of reviewer.

Satisfaction Optimization (Restaurant)

- **Sets and Indices**

F	Set of all restaurants
i	Index for i-th restaurant in F

- **Data**

s_i	Satisfaction metric for i-th restaurant
c_i	Cost per person in restaurant i
$DAYS$	Number of days to travel
$PERSON$	Number of people in the travel
$BUDGET$	Budget amount

- **Decision Variables**

X_i	Indicates whether restaurant i is selected

- **Formulation**

$$\begin{aligned}
&max \sum_{i \in F} s_i X_i && && (1)\\
s.t. &\left(PERSON \sum_{i \in F} c_i X_i \right) \leq BUDGET && \forall i \in F && (2)\\
&\sum_{i \in F} X_i = 2(DAYS) && \forall i \in F && (3)\\
&X_i \in \mathbb{B} && \forall i \in F && (4)
\end{aligned}$$

- **Discussion**

  Equation (1) is our objective function which is maximizing total satisfaction.

  Equation (2) is the constraint of the total expected cost can't be exceeding the budget.

  Equation (3) is the constraint of selecting exactly two restaurants for each trip day.

  Finally, each decision variable is a binary variable which represents whether the restaurant is selected or not. It is given in equation (4).

Satisfaction Optimization (Recreation)

- **Sets and Indices**

R	Set of all recreational spots
i	Index for i-th spots in R

- **Data**

s_i	Satisfaction metric for i-th spots
c_i	Cost for recreational spot i
t_i	Expected time spent on spot i
$DAYS$	Number of days to travel
$BUDGET$	Budget amount

- **Decision Variables**

X_i	Indicates whether spot i is selected

- **Formulation**

$$\begin{aligned}
max \sum_{i \in R} s_i X_i & & (1)\\
s.t. \sum_{i \in R} c_i X_i \leq BUDGET & \quad \forall i \in R & (2)\\
\sum_{i \in R} t_i X_i \leq 9(DAYS) & \quad \forall i \in F & (3)\\
\sum_{i \in R} X_i \leq 2(DAYS) & \quad \forall i \in F & (4)\\
X_i \in \mathbb{B} & \quad \forall i \in F & (5)
\end{aligned}$$

- **Discussion**

  Equation (1) is our objective function which is maximizing total satisfaction.

  Equation (2) is the constraint of the total expected cost can't exceed the recreation budget.

  Equation (3) is the constraint of the total time spent on recreational spots can't exceed 9 hours per day.

  Equation (4) is to ensure that we don't go to more than 2 places each day on average because we will be exhausted.

  Finally, each decision variable is a binary variable which represents whether the recreational spot is selected or not. It is given in equation (5).

Route Optimization

- **Sets and Indices**

F	Set of all restaurants (food)
R	Set of all recreational spots
$N = F \cup R$	Set of all spots without the current lodging
A	Set of all arcs between two locations
i	Index for i-th location (alias j)
k	Index for current lodging spot
$S = N \cup \{k\}$	Set of all spots (restaurant, recreational, current lodging)

- **Data**

$dist_{ij}$	Distance between two locations
$time_i$	Time spent on i-th location
$numDays$	Number of days on a trip
$maxHourToLunch$	Maximum hour spent on recreation before lunch time
$isRestaurant_i$	Indicates if spot i is a restaurant

- **Decision Variables**

X_{ij}	Indicates whether the route involves going from location i to j
$CUMTIME_i$	Helper decision variable that represent cumulative time spent from lodging
$CUMNUMEATS_i$	Helper decision variable that represent cumulative number of restaurant visited

- **Formulation**

$$
\begin{aligned}
& min \sum_{i,j \in A} dist_{ij} X_{ij} && && (1)\\
& s.t. \sum_{j \in S, i \neq j} X_{ij} = 1 && \forall i \in N && (2)\\
& \sum_{i \in S, i \neq j} X_{ij} = 1 && \forall j \in N && (3)\\
& \sum_{i \in N} X_{ik} = numDays && && (4)\\
& \sum_{j \in N} X_{kj} = numDays && && (5)\\
& X_{ij} \implies CUMTIME_i + time_j = CUMTIME_j && \forall i,j \in A, i \neq 0, j \neq 0 && (6)\\
& time_i \leq CUMTIME_i \leq 12 && \forall i \in N && (7)\\
& X_{ij} \implies CUMNUMEATS_i + isRestaurant_j = CUMNUMEATS_j && \forall i,j \in A, i \neq 0, j \neq 0 && (8)\\
& CUMNUMEATS_i \leq 1 && \forall i \in N && (9)\\
& \sum_{i \in F} X_{ik} = numDays && && (10)\\
& 1 - X_{ik} \implies CUMTIME_i \leq maxHourToLunch + time_i && \forall i \in F && (11)\\
& X_{ij} \in \mathbb{B} && \forall i,j \in A && (12)\\
& CUMTIME_i \in \mathbb{R} && \forall i \in N && (13)\\
& CUMNUMEATS_i \in \mathbb{N} && \forall i \in N && (14)
\end{aligned}
$$

- **Discussion**

  Equation (1) is our objective function which is minimizing total distance.

  Equations (2) and (3) are the constraints of the total outflow and inflow for all nodes repectively must equals to one. This ensures that each location will be visited exactly once (except the lodging).

  Equations (4) and (5) are the constraints for the number of days trip.

  Equation (6) is the constraint for the CUMTIME decision variable which is the total cumulative time spent after visiting a location. It uses logical operation.

  Equation (7) is the lower bound and upper bound for CUMTIME. The lower bound is the time spent for that location, and the upper bound is restricts that the total time spent in a day. Both equation (6) and (7) are used to prevent subtour.

  Equation (8) and (9) are to prevent visiting restaurants more than twice a day (lunch and dinner).

  Equation (10) is the constraint to ensure that we're not visiting any other places after dinner.

  Equation (11) is to prevent late lunch, which means that if a restaurant is not selected during dinner time, than the cumulative time after eating at the restaurant must not exceed the $maxHourToLunch$ variable plus time spent for eating. Since the equation will be non linear, the constraint is substitute with $CUMTIME_i \leq (12X[i,k]) + (maxHourToLunch + time_i)$ which has the same result.

  The Decision variables X_{ij} are binary variables which represents the selected route of our trip. It is given in equation (12).

  The helper decision variables $CUMTIME_i$ and $CUMNUMEATS_i$ have a non negative real domain and non negative integer respectively, which is represented in equation (13) and (14)

Further Discussion

In terms of scalability, there'll be a significant reduction of the algorithm speed when fed with a bigger dataset. There are some potetential alternatives for solving the sub-problems (satisfaction optimization) such as **Dynamic Programming** and **Metaheuristic Search**. Since the sub-problems are modified knapsack problem, we can easily substitute the Integer Programming with the **Dynamic Programming** and still get the optimal solutions. The more desirable method is **Metaheuristic Search** which will be implemented in the future. There are several reasons why it is desirable:

- It's significantly faster than the MILP method
- The randomness output from heuristic makes the trip unpredictable (and full of surprise) while maintaining the near optimality of satisfaction
- The randomness also makes the app reusable since each result will be new on every run.

We can also change the optimal solution by tweaking the problem parameters. Several constants that could highly impact the solutions are:

- maxTimeSpentPerDay: which is the constant 12 in the master route optimization problem (5). This constant represents the number of allowable time spent on each day
- maxHourToLunch: the data for master route optimization problem. This number is the maximum allowable hour that can spent before lunch. Changing this parameter will bring lots of changes to the optimal routing solution

It is also guaranteed to converge to the optimal solution using the method described above. The only requirement is the feasibility of the solution (budget, number of days, etc).