`

TravelPlanner

Manan Gupta (manang2), Saksham Jain (saksham8), Vansh Porwal (vporwal3)

1. Problem Statement:

Travel planning is essential in travel as it allows you to budget for your trip, think ahead about things to do, book restaurants and activities ahead of your visit so for doing all such planning, it takes a lot of hassle and bookkeeping. That is where “Travel Planner” comes in. To learn about where to go, what to do, and other travel-related information, we frequently search for travel blogs. We already have a list of items in our heads by the time we have looked at two to three pages.Important decisions we need to take are

- At what time we will start the trip
- Under what radius do we want our destinations to be, could be a city?
- Our starting location
- In how many days will we be able to complete the whole trip optimally?
- How much time do we want to spend at a place?

This causes a lot of confusion, and we frequently put off planning and traveling as a result. To solve this dire issue, we are optimizing the travel itinerary based on the various parameters.

TravelPlanner is an optimization model, which would input the radius of the search area and the place address/postal code you want to visit, for how many days the visit would last, At what time you want to start your trip and from where? Based on this information we would provide the user with the number of places he can visit optimally in the given number of days while maximizing the rating.

1.1 Implicit Assumptions:

Initially we are assuming the user would visit a place for 120 minutes and this trip would start from 9am in the morning.

2. Data Extraction and Pre-processing:

We extracted pertinent attributes using the places and distance matrix APIs (Google), then converted them before passing them to the itinerary planning optimization model.

2.1 Google Places API:

Each of the API services is accessed as an HTTP request, and returns either an JSON or XML response. All requests to a Places service must use the https:// protocol, and include an API key. The Places API uses a place ID to uniquely identify a place. Example of an extracted result


```
{'business_status': 'OPERATIONAL',
 'geometry': {'location': {'lat': 42.0396752, 'lng': -88.0338769},
  'viewport': {'northeast': {'lat': 42.04116762989273,
    'lng': -88.03247527010727},
   'southwest': {'lat': 42.03846797010728, 'lng': -88.03517492989272}}},
 'icon': 'https://maps.gstatic.com/mapfiles/place_api/icons/v1/png_71/generic_business-71.png',
 'icon_background_color': '#13B5C7',
 'icon_mask_base_uri': 'https://maps.gstatic.com/mapfiles/place_api/icons/v2/generic_pinlet',
 'name': 'LEGOLAND Discovery Center Chicago',
 'opening_hours': {'open_now': False},
 'photos': [{'height': 1836,
   'html_attributions': ['<a href="https://maps.google.com/maps/contrib/111755055144273606031">Kynyada Pipkin</a>'],
   'photo_reference': 'ARywPAJ0bqrdwQcaIuUK2x4arfJqT1WdB2BfJDeALvqx4maMvTJFA7f9iCIdBRzzfp2cqk-Om6REXbHWw7tYFiHsbClN-J
SuOio83ShdK6egGPvzdlq4olzl_oSuVczq7MkrOPp4r5kB40KnByqzb-fL1bS6MySD66pUEWrVa7-JHxjALG3J',
   'width': 3264}],
 'place_id': 'ChIJj_FcAL-vD4gRa1m_ByuRKGQ',
 'plus_code': {'compound_code': '2XQ8+VC Schaumburg, Illinois',
  'global_code': '86JH2XQ8+VC'},
 'rating': 3.9,
 'reference': 'ChIJj_FcAL-vD4gRa1m_ByuRKGQ',
 'scope': 'GOOGLE',
 'types': ['tourist_attraction', 'point_of_interest', 'establishment'],
 'user_ratings_total': 3163,
 'vicinity': '601 N Martingale Rd, Schaumburg'}
```

`

2.2 Distance Matrix API:

The Distance Matrix API returns information based on the recommended route between start and end points. You can request distance data for different travel modes, request distance data in different units such kilometers or miles, and estimate travel time in traffic. In the extracted example we are finding distance between two addresses -

```
Distance between 'LEGOLAND Discovery Center Chicago' and 'Lake Rishling'

{'destination_addresses': ['700 Fargo Ave, Elk Grove Village, IL 60007, USA'],
 'origin_addresses': ['601 N Martingale Rd, Schaumburg, IL 60173, USA'],
 'rows': [{'elements': [{'distance': {'text': '7.4 km', 'value': 7416},
     'duration': {'text': '10 mins', 'value': 621},
     'status': 'OK'}]}],
```

2.3 Creating Dataset:

As per our postal code and radius, we extracted latitude and longitude for each location using the Distance matrix. Using the duration between each node from our location and duration between each of them we created a data frame. We further added attributes such as opening and closing time and rating, to better optimize the given problem.

	LEGOLAND Discovery Center Chicago	Lake Rishling	Spring Valley Nature Center & Heritage Farm	Volkening Heritage Farm	Schweikher House Preservation Trust
LEGOLAND Discovery Center Chicago	1 min	16 mins	51 mins	23 mins	12 mins
Lake Rishling	13 mins	1 min	53 mins	25 mins	22 mins
Spring Valley Nature Center & Heritage Farm	51 mins	53 mins	1 min	45 mins	1 hour 0 mins
Volkening Heritage Farm	19 mins	24 mins	40 mins	1 min	31 mins
Schweikher House Preservation Trust	18 mins	23 mins	58 mins	30 mins	1 min

	opening_time	closing_time
LEGOLAND Discovery Center Chicago	600	1080
Lake Rishling	540	1260
Volkening Heritage Farm	600	960
Spring Valley Nature Center & Heritage Farm	480	1020
Busse Woods	480	1200
Schweikher House Preservation Trust	600	840
Elk Grove Historical Museum	720	1020
Busse Forest Elk Pasture	360	1080
Bison's Bluff Nature Playground	600	960
Itasca Historical Depot	540	840
Itasca Caribbean Water Park	660	1080
Fountain Square Park	600	1140
Wesley G. Usher Memorial Park	540	1260
Itasca Park District	540	840
Peppa Pig World of Play Chicago	720	1080
Sunset Park	420	1200
Falcon Park	300	1380
Robert T Jackson Clearwater Park	300	1380
Sandberg Christmas Display	960	1320
Medinah Park	360	1200

`

3. Proposed solutions:

Approach 1: In this approach we would ask the user to enter the radius,under which they want to explore all the famous tourist places in a city. For each day our Optimization model will try to maximize the number of places the user can visit within their respective operating hours. We will iteratively run the model for the remaining places, until we get the number of days a user needs to spend, to visit all of the tourist places. However, this approach might not always suit the user, that's where approach 2 fills the gap.

Approach 2: In this approach, instead of fixating on visiting all the tourist places, we take into account the number of days the user wants to spend on their trip. Based on the number of days, we then divide all the tourist places into clusters using K-means clustering such that the number of days matches the number of clusters. Then we use our optimization model to maximize the number of places that the user can visit within their respective operating hours, for each of the clusters.

4. Formulation:

Variables-

1. x_i - decision variable for the tourist place (Binary)
2. y_{ij} - decision variable for going from tourist place i to j (Binary)
3. t_i- Time of arrival at place i

Parameters-

1. F - the fixed time spent at a place
2. t_{ij} - time to traverse from place *i to j*
3. O_i - opening time
4. C_i - closing time

Maximize : $\Sigma_i x_i$

Subjected to-

1. $T_i \geq O_i * x_i$	$\forall i$
2. $T_i \leq (C_i - F) * x_i$	
3. $\Sigma_j y_{(\text{starting location}, j)} = 1$	$\forall i$
4. $\Sigma_i y_{(i, \text{ending location})} = 1$	$\forall j$
5. $T_i + F + t_{ij} * y_{ij} = T_j * y_{ij} + M(1 - y_{ij})$	$\forall i,j : i \neq j$
6. $\Sigma_j y_{ij} + y_{(i, \text{ending location})} = x_i$	$\forall i,j : i \neq j$
7. $\Sigma_j y_{ij} + y_{(i, \text{ending location})} = \Sigma_j y_{ji} + y_{(i, \text{ending location})}$	$\forall i,j : i \neq j$
8. $T_i \geq 0$	$\forall i$
9. $y_{ij} \in \{0,1\}$	$\forall i,j$
10. $x_i \in \{0,1\}$	$\forall i$

`

4.1 Constraint description:

- *Constraint 1 and 2* specify that the time of arrival and departure from a place is within the opening and closing time i.e. the operating hours
- *Constraint 3 and 4* specify that the user starts from a particular starting location and end the trip in a particular ending location
- *Constraint 5* specifies that if the person traveled from a place $i \rightarrow j$, then the time of arrival at the place should be equal to the sum of⇒ the time of arrival at place *i,* the fixed waiting time at the place and the time taken to traverse from place $i \rightarrow j$
- *Constraint 6* specifies flow preservation i.e. all the nodes will have one in-degree and one out-degree
- *Constraint 7* specifies that we exit a place that we visited

5. Results and Discussion:

We are presenting the results for the two discussed approaches taking Chicago as Input. We got 20 tourist destinations and their respective opening and closing times from Google API.

5.1 Approach 1 : Iterative Solving

We ran our optimization model iteratively on all 20 places until we got the desired number of days that would be required to travel all the places. For instance, In the first Iteration, our optimization model gave us the six places that we can travel in the span of one day. In the next iteration, we removed those six places from our network and optimized for the 14 remaining places. Essentially, each iteration represented a day and our model solved the network problem to show that we can travel to all the places in a total of four days. The detailed results are as follows:

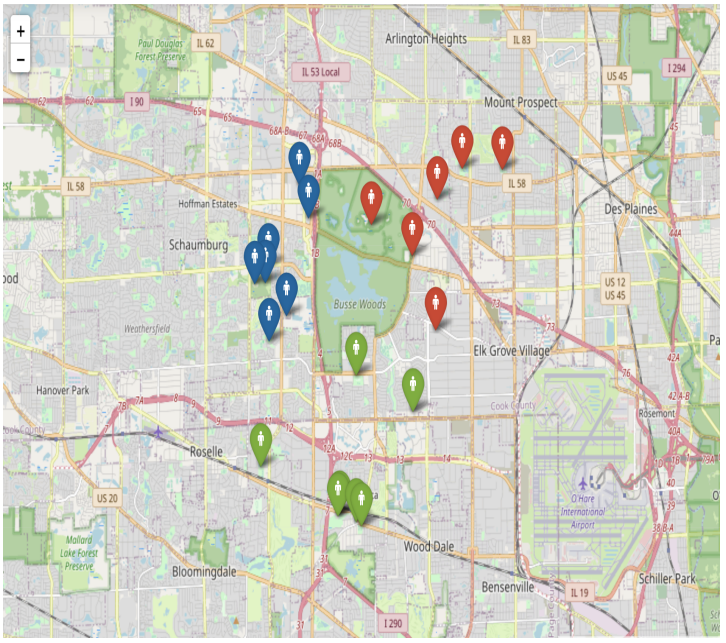
	Place	Arrival Time	Departure Time
Day 1	Schweikher House Preservation Trust	10:00	12:00
	LEGOLAND Discovery Center Chicago	12:07	14:07
	Elk Grove Historical Museum	14:14	16:14
	Fountain Square Park	16:21	18:21
	Wesley G. Usher Memorial Park	18:32	20:32
	Robert T Jackson Clearwater Park	21:00	23:00
Day 2	Itasca Historical Depot	09:00	11:00
	Volkening Heritage Farm	11:13	13:13
	Busse Forest Elk Pasture	13:26	15:26
	Itasca Caribbean Water Park	15:44	17:44
	Lake Rishling	17:56	19:56
	Sandberg Christmas Display	20:00	22:00
Day 3	Busse Woods	09:00	11:00
	Itasca Park District	11:15	13:15
	Bison s Bluff Nature Playground	13:28	15:28
	Peppa Pig World of Play Chicago	15:36	17:36
	Medinah Park	17:49	19:49
	Falcon Park	20:03	22:03
Day 4	Spring Valley Nature Center & Heritage Farm	09:00	11:00
	Sunset Park	11:18	13:18

`

5.2 Approach 2 : K-means Clustering

Assuming the user had three days to travel in Chicago, we performed K-means clustering on the 20 places based on their latitudes and longitudes to form 3 clusters. Then we used the optimization model for the 3 clusters of places and found the maximum places the user can travel in the given clusters. For instance, our model solved the network problem to show that we can travel 4 places in the given cluster of seven places on the first day of travelling. The detailed results are as follows:

	Latitude	Longitude
LEGOLAND Discovery Center Chicago	42.039675	-88.033877
Lake Rishling	42.013491	-87.976167
Volkening Heritage Farm	42.024350	-88.058098
Spring Valley Nature Center & Heritage Farm	42.024618	-88.053424
Busse Woods	42.038122	-88.005295
Schweikher House Preservation Trust	42.016974	-88.043616
Elk Grove Historical Museum	42.003090	-88.012202
Busse Forest Elk Pasture	42.030823	-87.986801
Bison's Bluff Nature Playground	42.028296	-88.052080
Itasca Historical Depot	41.971089	-88.019578
Itasca Caribbean Water Park	41.970444	-88.020316
Fountain Square Park	42.011130	-88.051743
Wesley G. Usher Memorial Park	41.969255	-88.012309
Itasca Park District	41.968054	-88.009675
Peppa Pig World of Play Chicago	42.047253	-88.038049
Sunset Park	42.050898	-87.946040
Falcon Park	42.043845	-87.975589
Robert T Jackson Clearwater Park	42.051094	-87.963935
Sandberg Christmas Display	41.994823	-87.986480
Medinah Park	41.981938	-88.055257



	Place	Arrival Time	Departure Time
Day 1	Spring Valley Nature Center & Heritage Farm	09:00	11:00
	Bison s Bluff Nature Playground	11:02	13:02
	Fountain Square Park	13:07	15:07
	Peppa Pig World of Play Chicago	15:16	17:16
Day 2	Lake Rishling	16:47	18:47
	Busse Woods	12:37	14:37
	Busse Forest Elk Pasture	14:41	16:41
	Sunset Park	10:25	12:25
	Falcon Park	21:00	23:00
	Robert T Jackson Clearwater Park	18:57	20:57
Day 3	Itasca Historical Depot	11:02	13:02
	Itasca Caribbean Water Park	15:13	17:13
	Wesley G. Usher Memorial Park	17:15	19:15
	Itasca Park District	09:00	11:00
	Sandberg Christmas Display	19:22	21:22
	Medinah Park	13:07	15:07

`

5.3 Inference from the above two approaches-

- The running time or the time complexity for the K-means clustering approach was much better than the iterative approach
- The iterative approach will always provide us with the most optimal solution whereas K-means clustering approach may not always give the optimal solution

6. Future Scope:

Although the current project covers the basic architecture to predict any itinerary. However, due to the limitation of time and scope of this project we undertook some implicit assumptions. Those on detailed analysis and thinking can be taken into account. We would like to present you with 3 ideas that we aim to work on in the future:

1. For now we have assumed a X person would spend a constant time at a location (taken 180 minutes in our example). Instead, we can take an average of x's history of visit to tourist places using his google maps data and do a predictive analysis of finding an average time he/she would likely to spend in a place, another way of approaching this problem would be to find the average time users are spending at a particular place, exploring any of these two methodology can help us skip the generalization and get a more concrete results.

2. Another thing that can be added would be to prioritize the particular places based on the reviews given by the users who previously visited. Ratings can be extracted from the google API as shown in our example above. Weighting these nodes based on the reviews can be beneficial to the user visiting, who would like to make the most from his limited travel time.

3. Lastly, right now for the multiple days problem we are using K means clustering and assigning K, what we can do is generalize the same and make our code flexible which is not vulnerable to changing of K values.