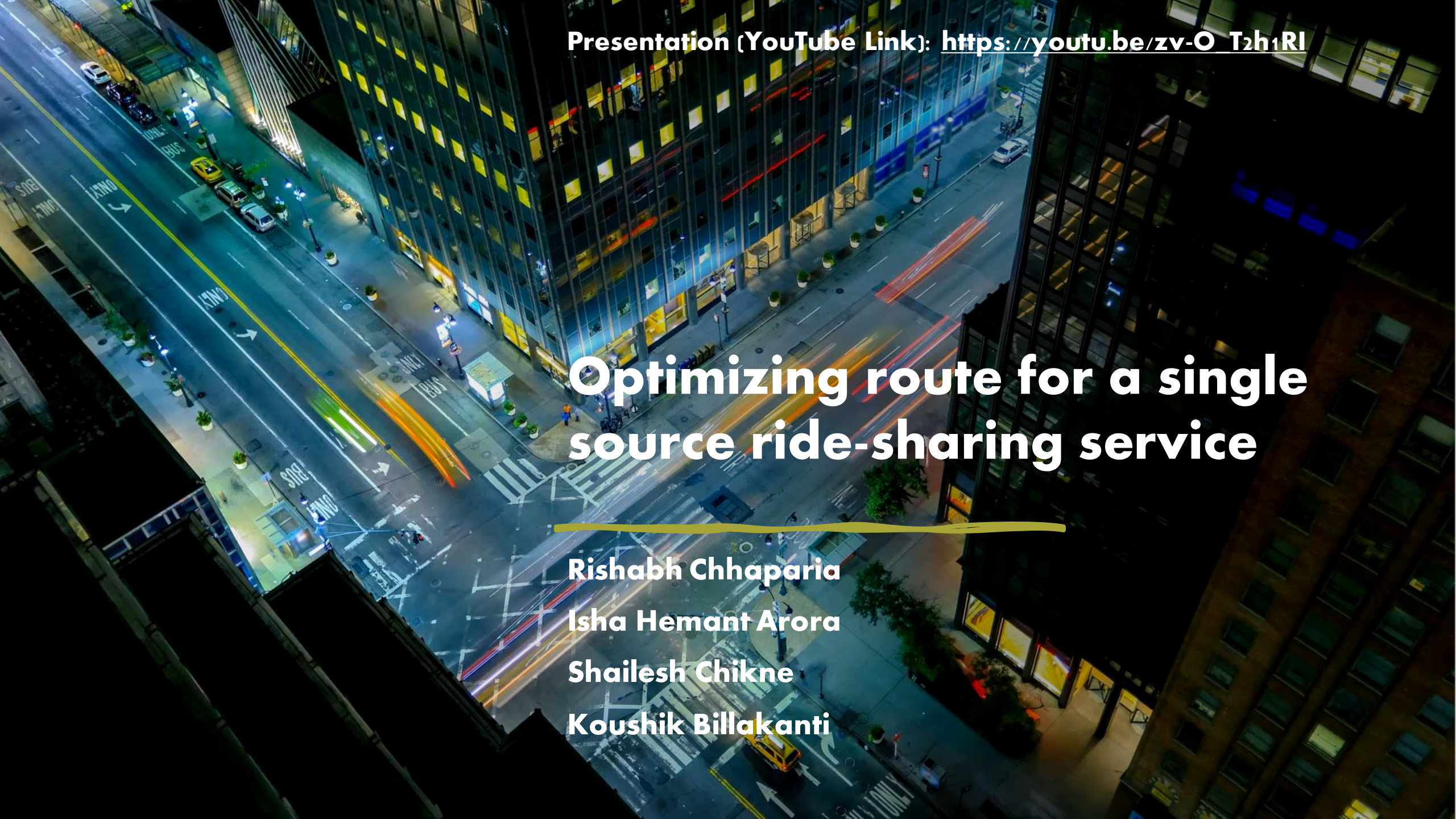


Presentation (YouTube Link): https://youtu.be/zv-O_T2h1RI

Optimizing route for a single source ride-sharing service

Rishabh Chhaparia

Isha Hemant Arora

Shailesh Chikne

Koushik Billakanti

Contents

Introduction

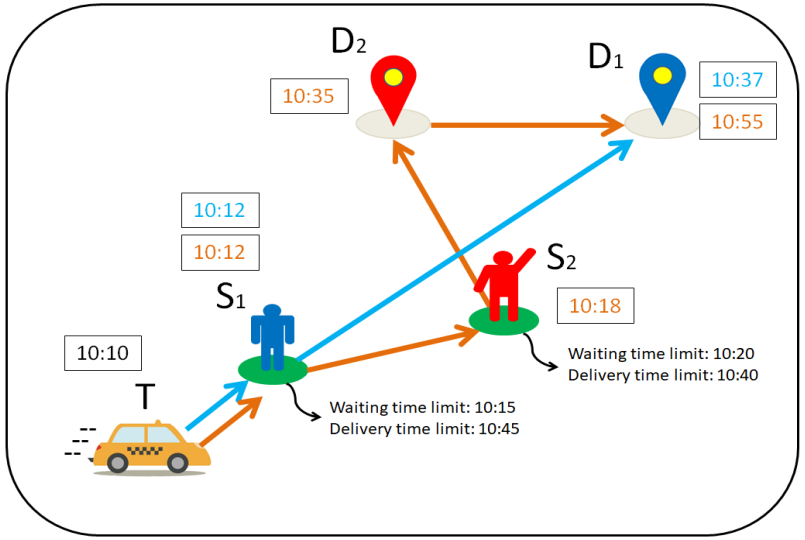
Why Car-pooling and what to achieve?

- It is the sharing of car journeys so that more than one person travels in a car and prevents the need for others to have to drive to a location themselves.
- This possess the following question: How does different people travel from one location to different destinations of many distances in an optimal time in one vehicle?
- We are using the same method in which Northeastern Red-eye is implementing in the daily rides.
- The Single-Source Ride Sharing Service was created with the idea of allocating seats to passengers that would result in a minimum cost, maximum capacity and the best route.
- We will be using algorithms to optimize the route for a single source ride-sharing service.
- By applying shortest path, we are trying to achieve the optimal time to travel to the destination for each individual passenger by finding the efficient routes and dropping each customer in a systematic manner.

Algorithms and Project goals

- The project we are working on mainly focuses on the idea of the shortest path algortihms. Some of the commonly used shortest path algorithms are BFS, DFS, Dijkstra's, Bellman-Ford and others.
- Basically, DFS (Depth First Search) goes to the bottom of a subtree, then backtracks.
- The only cons would be the exponential time complexity arising from multiple edges revisiting already visited nodes.
- DFS finds the shortest path in weighted graph by searching the entire graph and keep recording the minimum distance from source to the destination vertex.

- Our main motive of the project is to find the optimal time to reach the destination of every passenger in the vehicle who are travelling with the choice of car-pooling.
- Use Shortest path algorithms such as BFS to get the most efficient path to reach the destination of every individual.
- Applying the algorithms to the real-world datasets to understand about the uses after the implementation.



Analysis

Dataset and Adjacency Matrix

NYC Yellow Cab Service

- A dataset with multiple records of daily yellow cab trips in New York City.
- Includes parameters of pickup and dropoff locations, time taken, and distance travelled among others for each trip.
- The parameters: Pickup Location, Dropoff Location and Distance Travelled are considered.

Adjacency Matrix

- The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.
- Pickup and Dropoff Location IDs are vertices and Trip Distance are edge weights.

Test Dataset

Passenger Dataset

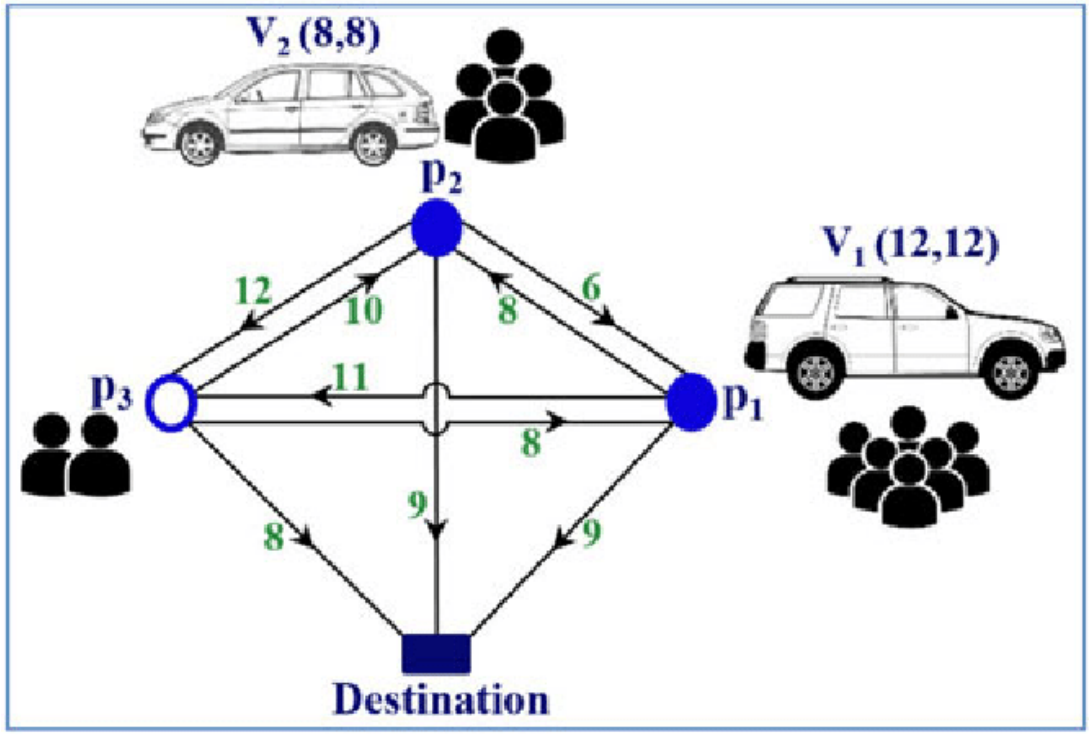
- A dummy dataset created to mimic passenger data and act as a test dataset to test out the algorithm created.
- Four parameters were considered:

  Passenger ID (does not repeat in a single timeframe booking)
  Booking Timestamp (to determine the booking was made)
  Dropoff Location (randomly decided location ID, does not include Pickup Location)
  Number of Denials (number of times passenger turned down by the system)

Implementation - DFS

- Use Case: Given a weighted directed graph with a dense network, the source-destination, and an adjacency matrix. find the MST such that it passes through all the required nodes.
- Algorithm Steps:

1. Traverse the graph with DFS, starting with the source node
2. Mark all the neighboring nodes as new sources and traverse the graph using DFS
3. After reaching destination, check if all the required intermittent nodes are visited, if not then reach the destination again via difference node(s)
4. Update the minimum cost and route once all conditions are satisfied.



Continue...

Limitations

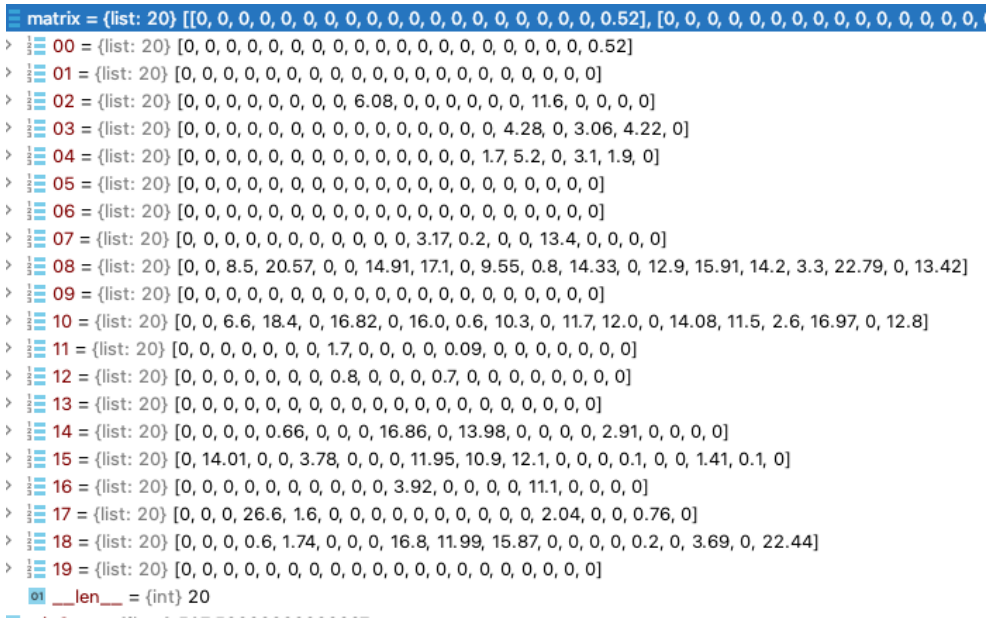
- Multiple iterations need to be run for the same batch of passengers to find the optimal route.
- The algorithm requires a dense matrix to find the best route from source to destination or it will fail.
- This can be considered as a rigid approach to the shortest path problem as it depends on the intermediate nodes.

Performance Evaluation

- Works well for smaller passenger pools but time complexity will increase for a larger number of passengers.
- Time Complexity: O(V + E)
- Space Complexity: O(V)

Results

passenger_id	booking_timestamp	dropoff_location	denials
10549	100420	216	5
10374	100502	201	5
10790	100950	215	5
10919	101103	209	5
10245	101311	206	5
10321	101525	217	4
10682	102026	211	4
10562	102026	208	3
10461	102242	212	3
10446	104107	214	3
10481	105105	220	3
10066	105229	213	2




```
{'passenger_ids': [10682, 10562, 10481, 10549, 10374, 10790, 10919, 10245],
 'route': [216, 201, 209, 212, 204, 205, 207, 217],
 'cost': 160.13}
```

Conclusion

Conclusion

- The algorithm implemented works as a proof-of-concept for our problem statement.
- We tried to cover all possible test cases for the problem statement.
- Limited testing of the model due to lack of relevant data or missing values.

Future Work

- Test the model on relevant data from multiple sources and verify the efficiency (such as real data from the RedEye service).
- Adjust the algorithm to include additional factors that affect the travel route (such as prevailing traffic conditions).

Thank You!

CS 5800 Final Project Report

(Summer 2022)

Optimizing route for a single source ride-sharing service

Shailesh Chikne
Rishabh Chhaparia
Isha Hemant Arora
Koushik Billakanti

Code - python (GitHub Link): https://github.com/chiknes/Algo_Project

Introduction

Why this Problem?

To carpool is essentially to enter an arrangement in which a group of people commute together in a car essentially preventing everyone from driving to a location themselves.

In the real world, the car-pooling concept works with private vehicles, commercial vehicles, and with institutes like schools, universities, and corporate offices.
There are multiple scenarios based on the type of car-pooling, Single Source - Multiple Destination, Multiple Sources - Single Destination, Random pickup, and Destination that includes Multiple Cars.

As students of Northeastern University one of the majorly encountered instances of carpooling are the Red-Eye services (Single Source Multiple Destination Ride Sharing).

The Red-Eye services being a major means of transportation for the students living off-campus (especially during late nights and harsh winters), the major motivation behind this project lies in the optimization of transit time and vehicle capacity for the ride-sharing service.

As a part of the project, we are trying to efficiently accommodate as many passengers as we can from one location to different destinations in the given range of distance allowed

by selecting the location of the starting point of the cab service. Optimization of locations selected by the passengers will be done so that the nearest destination of a particular passenger will be dropped off at first and other destinations will be interlinked by considering the nearest drop-offs of the previous dropoff. We believe that this will help minimize the fuel consumption of vehicles and average transit time for the passengers between various dropoff points.

Context

As per Shailesh Chikne

I've had vast experience of taking pooled vehicle services during corporate days where companies arrange shuttle buses for employees and with the RedEye service provided by NEU. I'd observed a few gaps while traveling with a few of these rides and realized there is a scope for applying an efficient algorithm to this problem. Through this project, I expect to learn how some of the best ride-sharing services function and how I can make inefficient ride-sharing services more efficient. In the beginning, the NYC carpooling service is a great example to analyze the working of such services and build a solution. Upon successful application of an algorithm, this can be extended to much larger problems.

As per Rishabh Chhaparia

Major cities nowadays struggle with crowded trains and buses and other forms of public transportation. Additionally, given the high cost of gasoline, using private cars or taxis seems to be a less attractive alternative for everyday transportation. If we can solve problems like the average amount of time a person spends in the car and reduce the number of miles driven, I believe ridesharing is a more affordable choice for travel than a private cab and also enables us to avoid congested public transportation. Since I utilized ridesharing services frequently when working as a corporate employee and through using the university's Redeye service, I found them to be interesting. I have noticed that the vehicle's route was not the most effective in terms of time and distance traveled several times when utilizing the service. So, I am interested in learning how to improve these services as doing so might possibly resolve a significant issue that most of us encounter on a daily basis.

As per Isha Hemant Arora

In this fast-paced world where time of essence with everyone dealing time constraints, saving time in any area of life can be beneficial and useful. Yet with the roads being packed with vehicular traffic, a single wrong turn can add minutes, even hours to your journey. One of the ways to help reduce traffic is the amazing concept of ridesharing. It has the benefits of traveling comfortably, while making sure to reduce a decent amount of automobile traffic on the roads. Yet not everyone is open to the idea of using these services. A lot of this reluctance stems from the fact that ridesharing generally leads to spending more time on the roads.
We hope to optimize routes and in essence, transit time as we believe not only will it help reduce the transit time when using ridesharing services, but also help with the moral predicament that most people face; environmental carbon emissions versus transit time. For me, this model is like what we understand the university RedEye services to be. The RedEye services are extremely beneficial to the university students living off campus especially when it becomes difficult to use any other form of public transportation (also while making sure that we save travel costs), whether it be because of harsh weather or be due to the timing of travel (traveling late into the night). Matching rides to optimize the total distance traveled by the cab while making sure that the average passenger transit time is minimized is what I hope to achieve from this project.

As per Koushik Billakanti

Algorithms have made our life so easy that they are implemented in almost every field of work and according to our project I could say that our problem statement and the algorithm which we are implementing in it will make a human life easier than before. We chose a simple topic of carpooling and we are trying to implement different algorithms to get a way which is easier to add on to in our daily lives. The topic which we selected will mostly decrease the use of fuels by pooling different passengers to reach their destination in an efficient way and this could decrease the road traffic rate by decreasing the usage of the number of vehicles. In the past few years, many universities in the USA have introduced this way of transit usage for the safety of the students during late night hours. But in some carpooling dealers or drivers are using the old technique of taking their own route to their destination, and many are waiting in traffic due to avoiding directions by apps. But this usage of technology should be used wisely to achieve the smooth ride.

Analysis

The Single-Source Ride Sharing Service was created with the idea of allocating seats to passengers that would result in a minimum cost, maximum capacity and the best route. We have analyzed an algorithm with the scenario of having 5 vehicles, each with the capacity of 8. Thus, in each iteration of an algorithm, a maximum of 40 seats can be occupied.

The Dataset

For the Graph

The dataset that was primarily used and helped in the creation of the basic network of locations was that from the New York City Taxi and Limousine Commission for the period of March 2017 (the data of the Yellow Cabs was considered).

This dataset was extremely extensive, with 17 parameters that included details about the pickup and dropoff location IDs, pickup and dropoff date and time, trip distances, fare amount among other things.

After careful consideration and scrutinizing the dataset thoroughly, we were able to narrow down our requirements to the three parameters of Pickup Location ID, Dropoff Location ID and the Trip Distance (in miles).

The location IDs (for both pickup and dropoff) ranged from 1-265.
Since creating a fully connected network for 265 location IDs would not only have been a little difficult to handle, but would also have given us a hard time understanding the efficiency of our algorithm, we narrowed the range of the location IDs down to 200-220.

Since, the data considered was collected over a span of a month, there were multiple instances of the pair of pickup and dropoff locations being the same. We believe that change in trip distance for such kinds of pairs would generally be dependent on external factors, ones we did not have enough information about. To ease the calculation and since the aim of the project is, essentially, to find the shortest possible path that can be taken, in such instances, the minimum distance was considered.

Building the Adjacency Matrix

After cleaning the dataset, we choose to build the graph by taking the adjacency matrix of 3 columns in the dataset. We choose Pickup Location ID, Dropoff Location ID as vertices and Trip Distance between the location IDs as the edge weights.

We took the pickup location ID, dropoff location ID as two sets of unique points and later we used the built-in function enumerate to represent those unique points as index and value of the matrix. Before the formulation of the matrix, the columns were represented in the form of lists and those lists were taken in the form of a dictionary to separate the key and value to build the matrix.

This created adjacency matrix is what was used when applying the algorithm on the dataset with the vertices being the rows and columns of the matrix and the edge weights being corresponding matrix element values.

Dummy Passenger Dataset

This dataset was not created explicitly, but rather was created as a part of the code (randomly generated each time). The created dataset aims to mimic all the different kinds of scenarios that could occur during customer bookings.

The dataset has four parameters - Passenger ID, Booking Timestamp, Dropoff Location and the Number of Denials.

To account for a different number of bookings each time and to make sure many possible scenarios can be covered, multiple dummy datasets were created over multiple different timeframes.

The Passenger ID was generated as a random number between 10000 and 11000 (without repetition in the dataset).
The Booking Timestamp indicated the time that the booking was made and was set within the timeframe indicated for the dataset.
The Dropoff Location, also generated randomly was a location ID between 200-220 (with the Pickup Location ID decided)
The Number of Denials is to indicate the number of times a passenger has previously been turned down by the system (is an arbitrary value in this dataset which is added to

account for any cab bookings that could have been denied in the iteration before it). More on this will be explained a little below, in the algorithm.

Algorithm (Depth First Search)

A Pickup Location ID is finalized that will act as the Source and will be static. All Dropoff Location IDs are dynamic.

A booking time-frame is selected within which the cabs will accept bookings.

Next, create a priority queue of the passengers that have requested a cab, with passengers having a higher number of denials and an earlier booking timestamp given more priority.

For the first 15 records in the queue, all possible combinations of 8 are created. For each of these combinations, a shortest-path network (using the Dropoff Location IDs and the Pickup Location ID) is created and the total cost value is calculated (total distance traveled) and the Location ID combination with the lowest cost value is selected as the optimal route.
When finding out the total cost value for each shortest-path network, the DFS algorithm is used.

The graph network is stored as a dictionary. Each key of the dictionary is a Location ID (as the starting location). The value for each key is a list of lists in the format of it being list for the pair of Location ID (as the ending location) and edge weight i.e.
if the trip distance between 200 and 201 is 5 and 200 and 202 is 8, then a key-value pair of the dictionary will look like:
graph[200] = [[201, 5], [202, 8]]
This graph includes all location IDs (as seen in the records being assessed), including the starting location ID as keys.

Depth First Search (DFS): The algorithm is generally used to traverse and search on graph data structures. The algorithm begins at the root node and proceeds to explore as far as it can along each branch before turning around. The idea is to start at a random node, mark it and then proceed to any unmarked adjacent node, and then follow the same procedure until there is no unmarked node. Following this, go back and look for any other unmarked nodes that are left to mark.

DFS is commonly used in determining shortest paths in a graph majorly because it is extremely helpful in calculating the minimum distance of all nodes from the source to the intermediate and destination nodes (for this algorithm to work, we need to find distance between intermediate nodes).

The graph is traversed starting from the Pickup Location ID node using DFS. Each neighboring Location ID node to the Pickup Location is marked and each of these new nodes can be considered as an intermediate source location for a DFS traversal. Once all the intermediate nodes are traversed and the destination (final) Dropoff Location ID is reached, the total cost value of the network created from the combination of location IDs is calculated. It should be noted that the final Dropoff Location ID for each cab is also decided by checking all possible location IDs involved and getting the least value possible for the total cost (distance traveled).

The route with the least possible total cost is assigned to the first cab and the allocated passengers are removed from the queue.

Passengers that were evaluated in the iteration but not allocated now have an increment in their ‘number of denials’ parameter by 1.

Once the combination for the first cab is decided, a similar algorithm is carried out for the next 4 cabs (40 passengers allocated) or until passengers request for the ride-sharing service (less than or equal to 40 passengers).

Time Complexity: O(V + E)
Since the DFS algorithm traverses each node and an edge only once, the maximum it could traverse is all edges and all vertices. Hence, for our problem statement of small chunk of dataset with 20 nodes and 380 edges, complexity is O(400).

Space complexity: O(V)
We maintained an array to store visited nodes, max size could be 20 in our case.

Conclusion

We believe that the algorithmic approach that we took, as discussed above, would help take the shortest route possible with the incoming bookings. While we may not have been able to test the model thoroughly (due to a lack of relevant data), we can definitely be assured that it is a proof of concept for the problem statement of single source ride-sharing services that we were working towards.

Summary

Everyday transportation is becoming a serious challenge and some problems like pollution, traveling cost, road traffic, fuel consumption are having a great impact on our daily lives. The car-pooling method can, to a very high extent, help solve the above mentioned problems.
This method has been in use for a long time in our lives, but many people are not choosing this option to save the problems happening in transportation.
Some of the problems people face with car-pooling are delayed rides, congestion in the vehicle, late rides to the destination.
Northeastern University has been using a car-pooling strategy to overcome the issues of student transportation to their homes during late nights. An intelligent and reliable carpooling solution aids not only the students but also enrich the societal benefits in terms of strengthen the social relationship. This car-pooling idea is based on techniques of different algorithms and finally the efficient algorithm is taken into consideration for optimal ride.

We considered a dataset having pickup and drop location with trip distance. Parameters like Booking Timestamp and Number of Denials are implemented to select the passengers within, say example, 10 minutes. Even while making sure that the ride accepts passengers within the smallest range of distance between each passenger, and if not, then transfers the passenger to the next ride, the number of denials will take care that no passenger is left waiting too long .

The DFS (Depth First Search) algorithm is used on the combinations of the two datasets using the adjacency matrix and other passenger relevant parameters that were used to find the optimal ride of the journey.

Limitations with the Dataset

- We had to clean a large part of the data including columns like CO2 percentage, trip data, passenger count to get an optimal solution using a limited number of rides.
- After performing the adjacency matrix, we came to a conclusion that the large amount of data is Sparse data.
- Some of the data had inconsistent information. Similar pickup and dropoff location IDs, let alone they be interchanged, did not equate to them having the same trip distance. Understandably, in the real world, it is not surprising, working with such data made it a little difficult for us to discern the most efficient way to calculate trip distance between location IDs.
- While the locations were numbered, they were not sequential. With the Pickup Location ID 1 (assume, static) and a Dropoff location ID as 20 could have a trip distance as 5 miles while another Dropoff location ID as 10 could have the trip distance of 15 miles making the basic algorithm incapable to work with.

The sparse adjacency matrix produced from the dataset produced a challenge for other MST algorithms, while DFS could manage to find a solution.

Once the limitations of the studied dataset were identified, building a new dataset was attempted to help find the optimal ride for the passengers.

Future Work

As the results of our algorithmic approach look extremely promising, we hope to be able to employ the better, relevant and larger datasets to test out and determine the feasibility of the model. We believe that this approach can be the core algorithm behind improving the models for Single-Source Ride-Sharing Services.

A set of suggestions are listed below to give a significant addition to the current method. We would also consider these to be the next step for our future work:

- Use large datasets with different parameters like user rating, multiple source rides, multiple rides at one go.
- Evaluate how factors other than shortest distance between locations can affect the cost of the travel and consequently affect the transit time of the cab.

- Employ Machine Learning algorithms like k-NN, Convolutional Neural Network(CNN).
- Deploy mobile or web applications similar to Uber and Red-Eye.

Takeaways

Shailesh Chikne

It was an experience to work again on an academic project, the outcome was satisfying and the amount of work put in was worth it. I personally liked the project idea and the learning was awesome. Although we couldn't manage to apply the DFS algorithm to the full-fledged dataset, we could definitely learn the core concepts and application of the DFS algorithm on a real dataset. The tools, coding language, meetings, collaboration, everything that was a part of this project, taught me something new which I'll use in my upcoming professional journey.

Rishabh Chhaparia

I've learned from working on this project that you can't just plug in an algorithm and get a good result. They must be carefully adjusted to meet the problem description. To accomplish that, it is necessary for us to comprehend how each algorithm we use operates at its core. It was surprising to observe that algorithms from the same family perform differently depending on the dataset they are used on. I'm curious to see how this algorithm performs on a real-time dataset, to better understand the use of algorithms in real-world situations.

Isha Hemant Arora

Its always said that implementation of concept generally never goes the way you'd expect it. That's probably one of the best ways to summarize this project. When we went into this project, we had hoped to implement multiple shortest path algorithms and try to find the best of them. While ending this project with a single implementable algorithm was definitely not on the agenda, it did give me a chance to really understand data and how its implications really land on choosing algorithms. I majorly enjoyed learning how supposedly similar working algorithms can really change pace when faced with a tough dataset. While I do believe that this algorithm would be very helpful to Single-source ride

sharing services, I hope to be able to extend the project a little more (by using reference from the future work mentioned above).

Koushik Billakanti

From this project, I have learnt how the different shortest path algorithms work on the data. On paper Dijkstra's algorithm is the most efficient algorithm among all other shortest path algorithms, but I understood that everything depends on the data considered and the best algorithm might change to the worst in some cases. This project would be helpful for the optimal implementation of red-eye in the campus and this can be taken as a basis to work on the data using machine learning models in the future and the deployment part would give the best approach for the ride-sharing approach.

References

[1] Dataset: https://www.kaggle.com/code/cavfiumella/nyc-cab-carpooling/data

[2] Ma, Changxi, Ruichun He, and Wei Zhang. "Path optimization of taxi carpooling." PLoS One 13.8 (2018): e0203221.

[3] Zhu, C., Ye, D., Zhu, T., & Zhou, W. (2022). Time-optimal and privacy preserving route planning for carpool policy. World Wide Web, 25(3), 1151-1168.