CS 581 - Database Management Systems
University of Illinois at Chicago
Spring (2020)

RIDESHARING
PROJECT REPORT

Team 10

Deblina Roy (droy7@uic.edu)
Peihong Man (pman3@uic.edu)
Srikanth Maganti (smagan20@uic.edu)

Contents

1. Introduction

Giving on-demand transportation benefits to the individuals is one of the effective concepts and is growing its reach each day. With numerous companies working within the open space and giving nearly the same encounter to their riders, it barely separates them. All these companies provide one where they offer a choice to its users/riders to share their ride with other users/riders with whom they have never met. There are numerous benefits of ridesharing that include savings of clients' money, expanding income to the company and being eco-friendly with more number of clients opting for this benefit than to require an individual vehicle.

Through this project, we implemented a ride sharing algorithm on spatio-temporal data. NYC taxi cab data (July 2015 - June 2016) has been used for evaluations.

2. Assumptions and Restrictions

2.1 Assumptions

- **LaGuardia** airport as origin and destination
- Period of time evaluated is **July 2015-June 2016**
- Max passenger taxi capacity is 4
- Pool Window - 2,5, and 10 minutes
- Delay tolerance allowed for each passenger is 0%, 10%, and 20%

2.2 Restrictions

- No Social Preferences
- At most 2 trips are shared
- No traffic conditions considered
- Walking within 1 min is allowed

3. Algorithms Evaluated

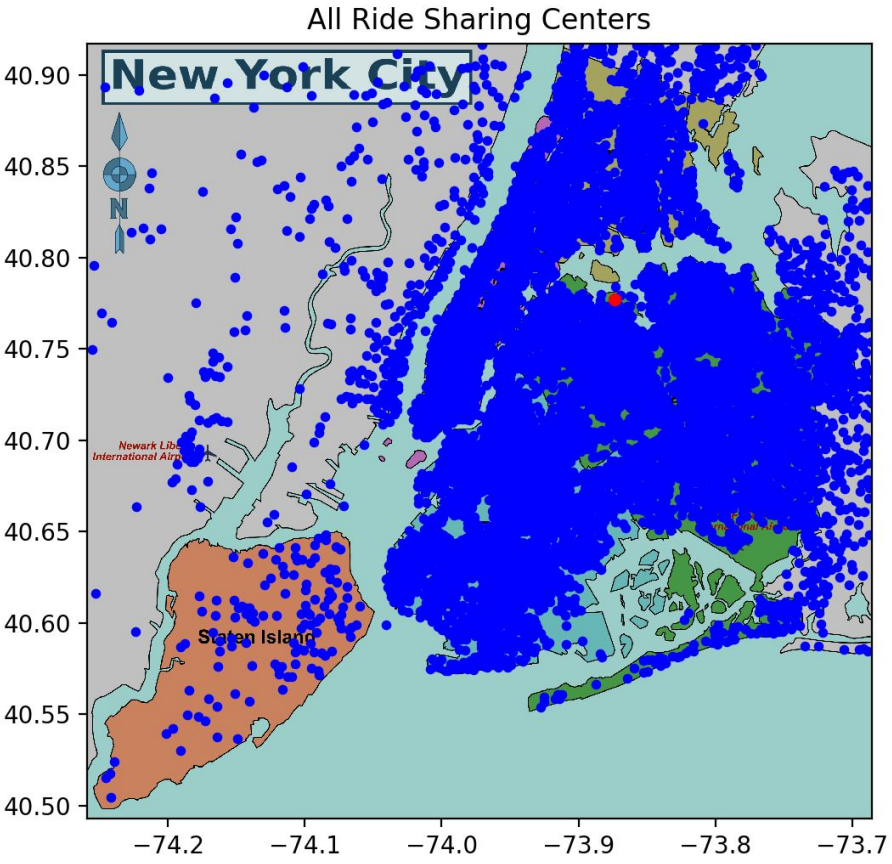
3.1 Overview

- Precomputation of all **the center points** using a threshold value
- **Selecting all rides** within the **pooling window** - 2 mins, 5 mins, and 10 mins
- **Check** whether a **pair of rides** is **shareable** by **2 constraints** adding delay as a variable
- **Figuring out the center point** of the trip requests locations and fetching the distance
- After computation of distances these distances will be passed to the **Recursive Depth First Search Algorithm**

3.2 Precomputation

- We have considered **6449 center points** of NYC with a threshold of **0.002 degree.** The radius of 0.002 degree corresponds to **0.035 mile** in terms of both latitude and longitude. So by considering this radius, we get to know the farthest point distance to its **nearest centre is approximately ~0.017 mile**
- The interesting point is that most people only need to walk less than 1 min and once the sharing pair is created the driver can pick them up from original location
- All these centers are created based on **real rideshare requests**. So the more rideshare requests in an area more the center points density
- The centers are **calculated in this way**: When a new ride share request comes in, check if there is a precious request close to it. If not, create a new center and all future requests that are close to it will be clustered to it.

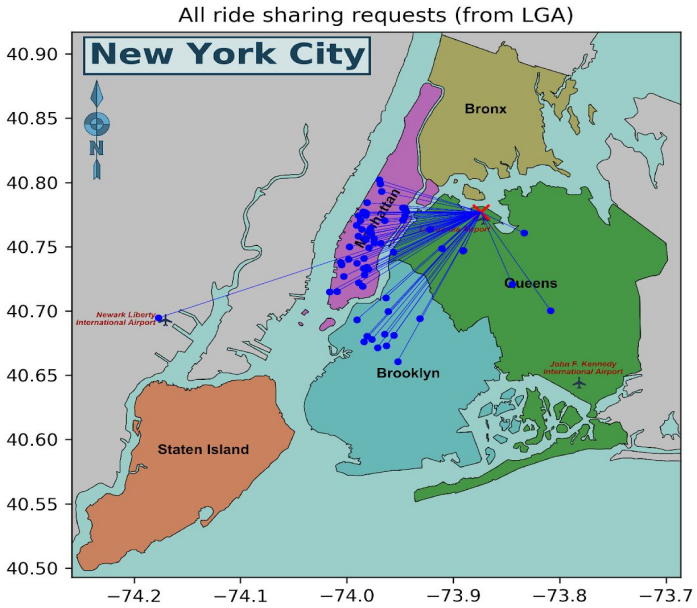
3.3 Ride Sharing Centers visualization

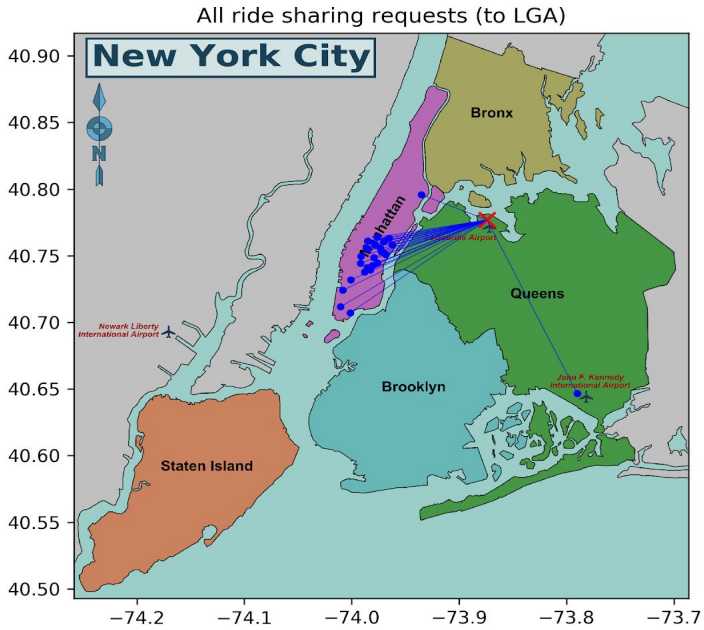


- The above figure depicts that Manhattan has the highest density
- Low density areas like Forest, Lake, etc which clearly shows no ride share requests

3.4 Selection of ride requests

- All rides within the specified pooling window are calculated with either trips “to Laguardia” or “from Laguardia”;
- Sample figures for all ride sharing requests on 05/20/2020 at 5:30PM to 5:35PM “From Laguardia” and “To Laguardia”





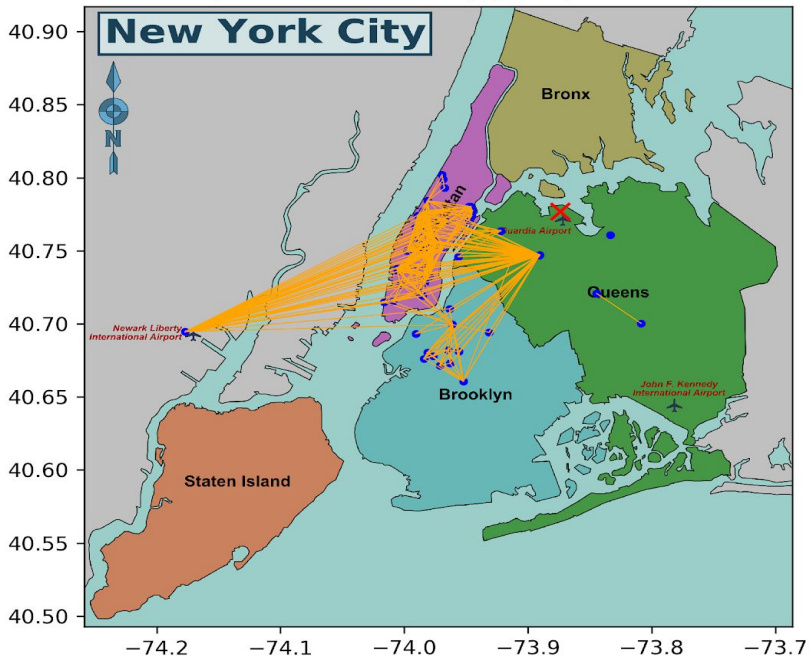
3.5 Figuring out the center points of trip requests

- **Calculating absolute value difference** between trip request latitude and center point latitude and the same process is done for longitude as well
    - abs(trip[lat]-center[lat]) < (tolerance_distance)*(latit_minimum)
    - abs(trip[lon]-center[lon]) <(tolerance_distance)*(long_minimum)
- **Iterating through all the center points and finding the least corresponding value** by simultaneously checking both the above conditions
- Finally **replacing trip request** latitude and longitude with **nearest center**

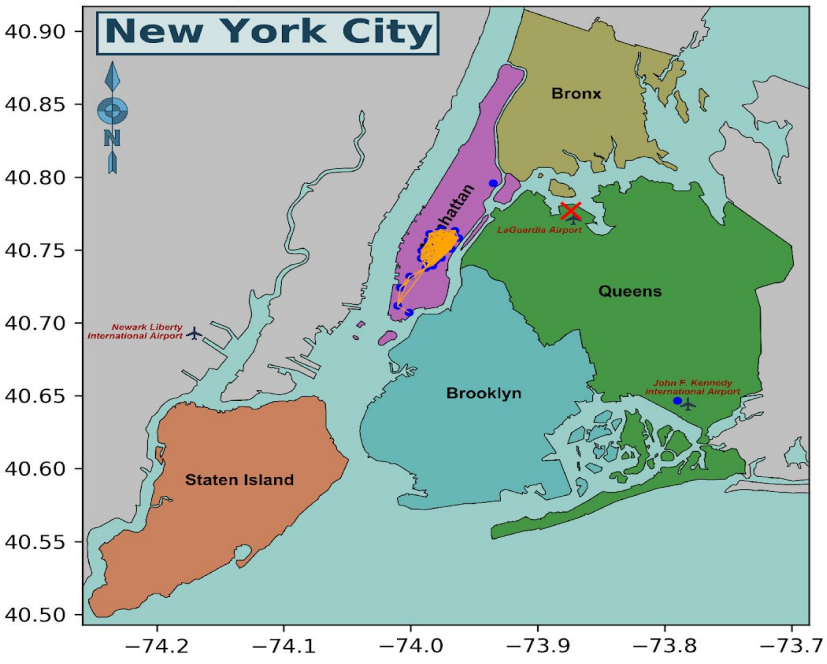
3.6 Check whether a pair of rides is shareable

- **Checking the below constraints and adding into the shareable rides**
- All the trip requests are shareable except the trips which are not satisfying below constraints with varying delay constraint
- **First Constraint: S(DA) < S(BA) and S(DB) < S(AB)**
- **Second Constraint S(DA) + S(DB) > S(DB)(Delay) and**
- **S(DB) + S(DB) + S(BA) > T(DA)(Delay)**
- In the above constraints S(IJ) represents time taken for I->J, D represents to/from Laguardia
- A,B represents 2 different locations for trip requests
- Sample figures for all possible sharing requests on 05/20/2020 at 5:30PM to 5:35PM “From Laguardia” and “To Laguardia”

All possible ride sharing pairs (from LGA)

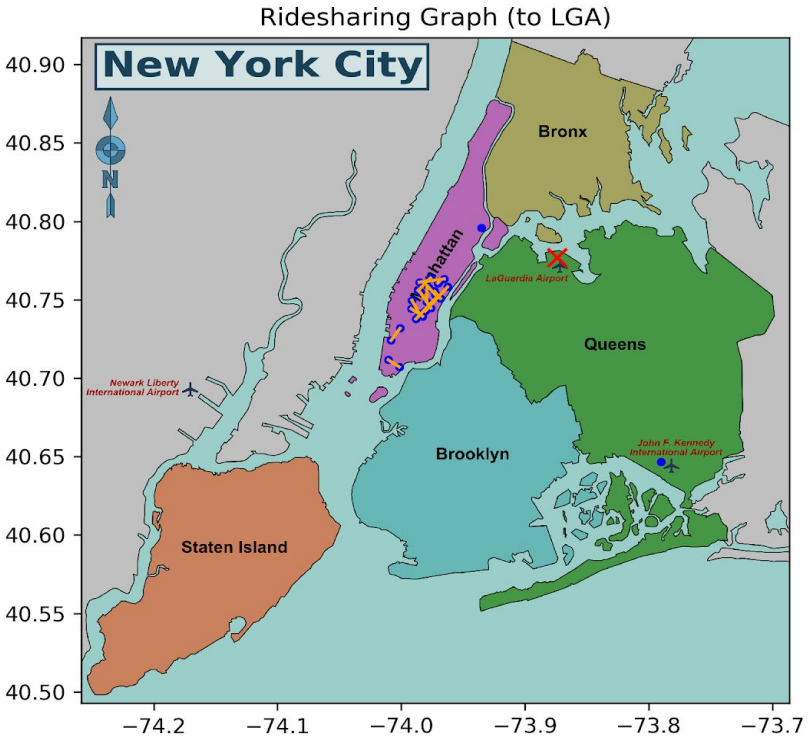


All possible ride sharing pairs (to LGA)

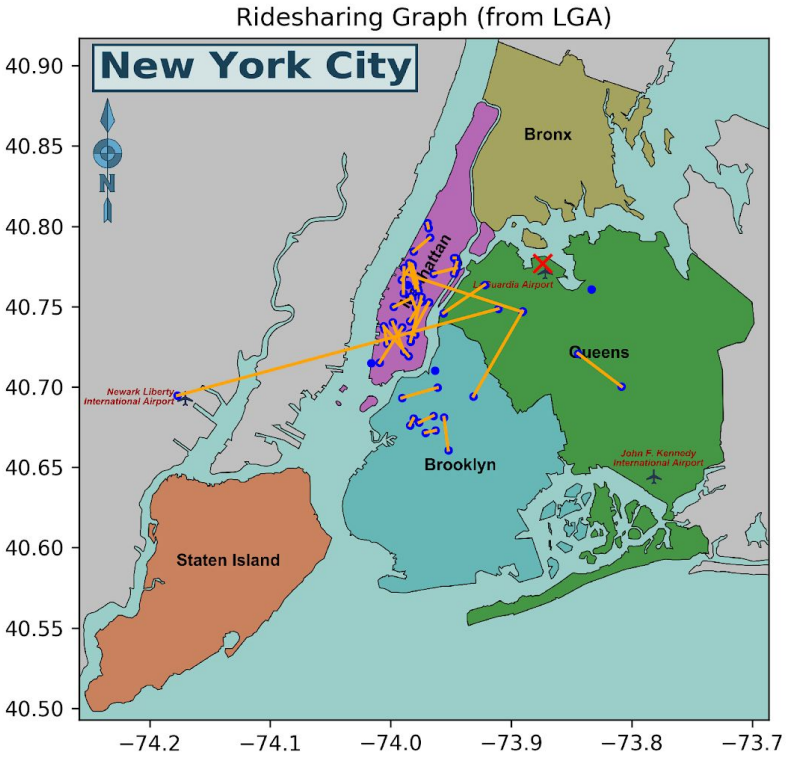


3.7 Recursive Depth First Search Algorithm

- All possible shares which were obtained from previous step will be used in this algorithm
- **Total time saved, rides shared, and shared people are parameters** for this algorithm
- As the algorithm works in a recursive way, we have established a base condition to make dead one of the branch
    - **Base Condition: If any of the people in new iterated values are already in shared people then that branch becomes dead**
- Once all the possible shares got iterated total time saved, shared rides, and shared people will be assigned globally
- Sample figure for final rides which got shared on 05/20/2020 at 5:30PM to 5:35PM “To Laguardia”



- Sample figure for final rides which got shared on 05/20/2020 at 5:30PM to 5:35PM “From Laguardia”



4. Experiments Conducted

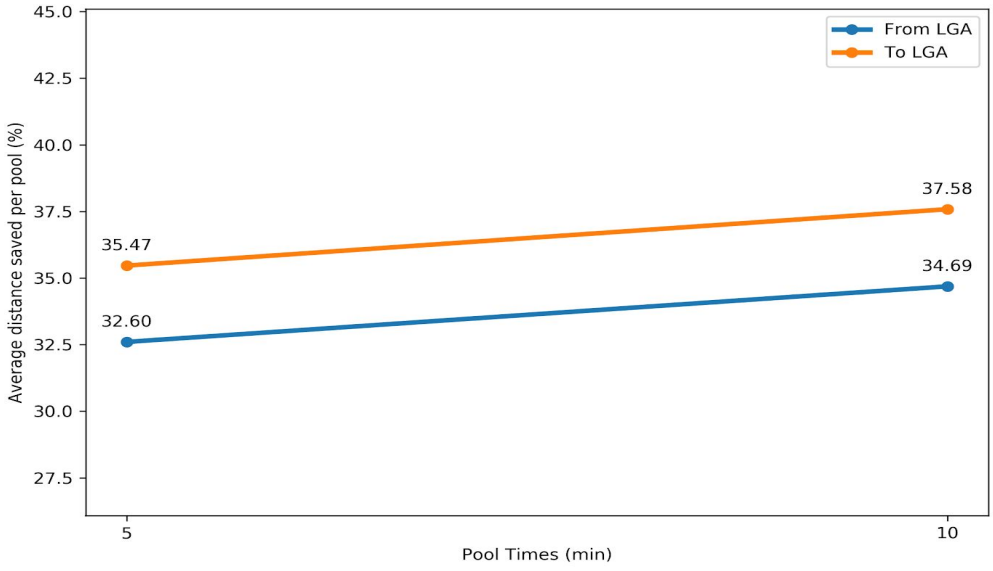
- Different combinations of pooling windows and delay times were considered
    - **Pooling windows 2 mins, 5 mins, and 10 mins**
    - **Delay times of 0%, 10%, and 20% are considered**
    - **July 2015 to June 2016 NYC** data was considered for the experiment
- During the initial stages of the project it took lot of time to compute all the values. Also, **GraphHopper** doesn't work for long. It used to throw an error as we have crossed max retries
- **Precomputation of all values** -
    - As explained in the above steps, we have precomputed all the values by installing GraphHopper in the local API
    - First we ran the algorithm with 1707 center points Later on as the algorithm works better we modified the algorithm to run with 6449
    - For the calculation of (6449*6449) points it took more than 72 hours in 2 individual systems with high processing power
- **Software/Technology used** -
    - Project Tools - Python, Jupyter Notebook
    - Collaboration Tools - Github for code sharing, Google drive for resource sharing
- **Libraries and Dependencies** -
    - Numpy python package
    - Pandas python package
    - GraphHopper Directions API
    - CSV package for python
    - Matplotlib python package
    - New York City Map (in .pbf format)
    - Datetime package
    - Json,Requests

5. Results

*5.1 Test Results for **distance saved per pool** (July 2015 to June 2016)*

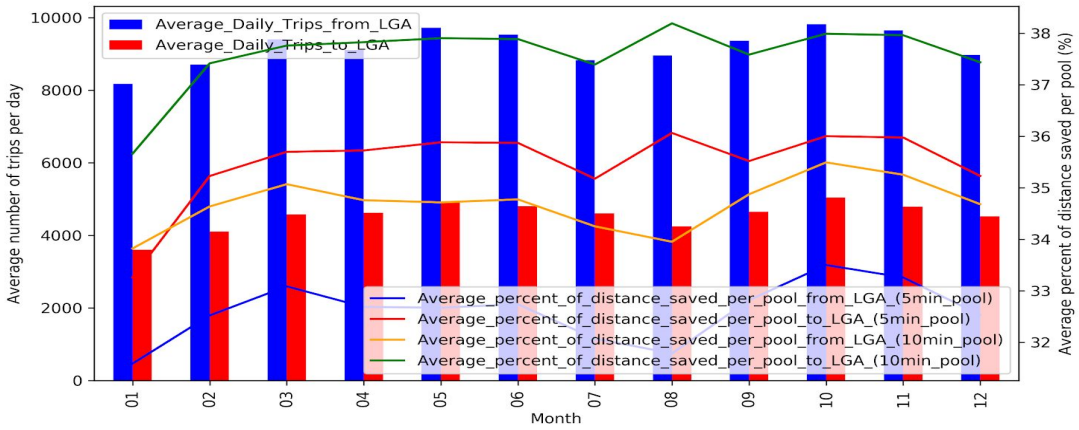
From the visualization of the trip requests location, we found that drop off locations from LGA usually go very far and cannot be shared with anyone. As a result, the amount of **distance saved is a bit higher for 'from LGA' compared to 'to LGA'.**
Also, there are more requests to merge in a large pool size, So, **higher the pool size, higher the savings**.

*5.1.1 Average **Distance** Saved per Pool (%) (with Delay 20%) (for- July 2015 to June 2016)*



From the above figure we can conclude that as pool time increases the average distance saved increases as well. As mentioned earlier, due to drop off locations from Laguardia going very far, distance saved is a bit higher for 'to LaGuardia'.

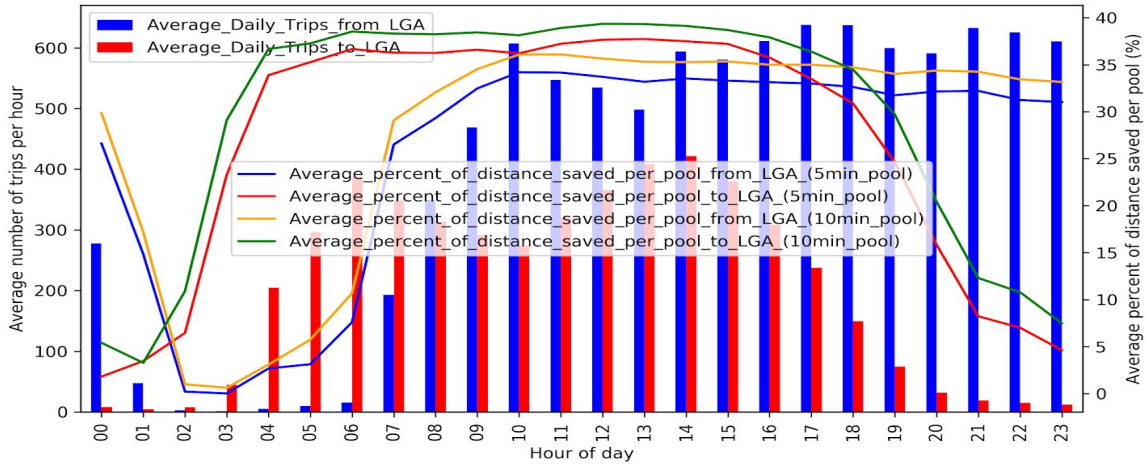
*5.1.2 Average **Distance** Saved Per Pool (with Delay 20%)* ***Monthly Breakdown***



X- axis: month starts from 01: July 2015 and ends with 12: June 2016

The above figure clearly shows that the average daily trips reach the peak stage during the holiday season that is late November(X-axis:05) and during the initial days of summer(10).

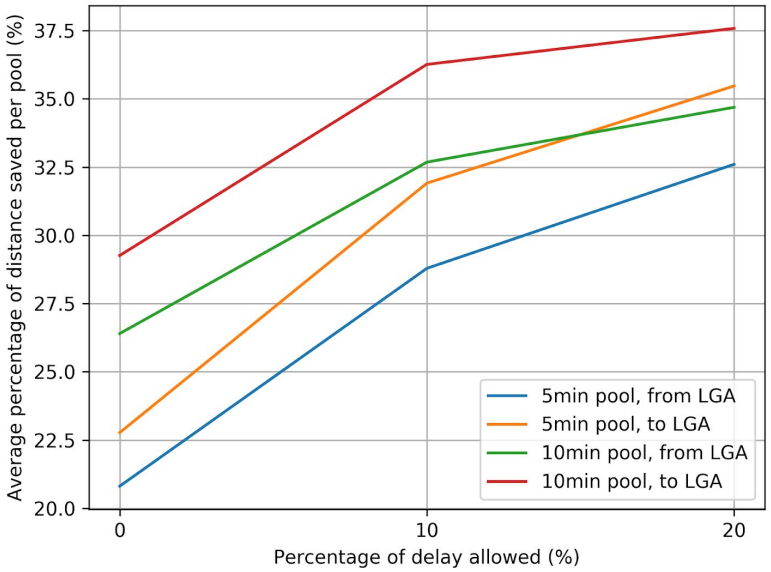
*5.1.3 Average **Distance** Saved Per Pool (with Delay 20%)* ***Hourly Breakdown***



In the above graph, we can observe a pattern where Average daily trips from Laguardia continuously increase and it reaches its peak late night whereas 'to Laguardia' trips follow the opposite trend. It is clearly evident if there are less number of trips, less distance is saved.

*5.1.4 **Distance** Saved for **delay**- 0%, 10% and 20%*

In this graph, the delay time is varied. As the delay increases, the savings increase as well.

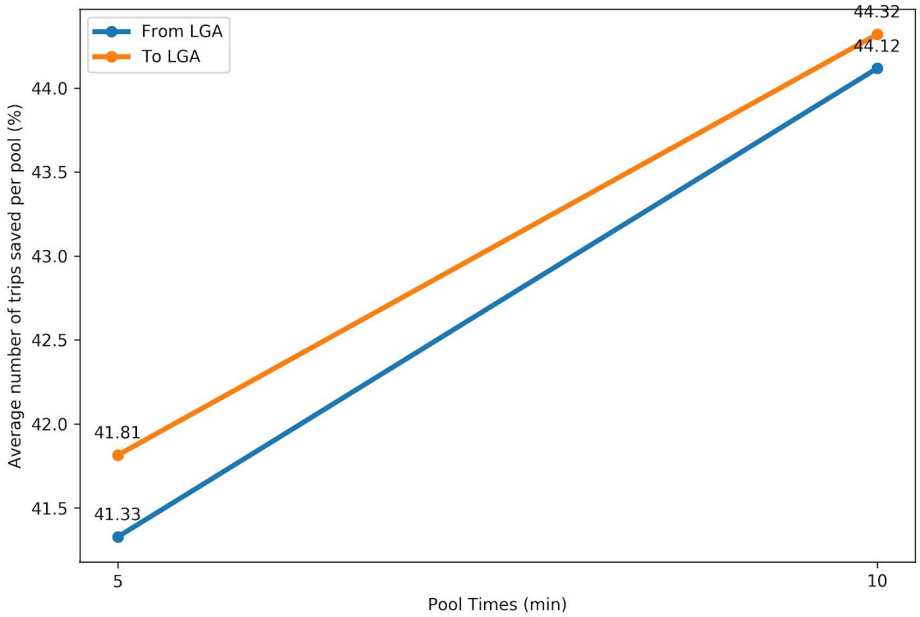


In the above graph, we can observe that delay time has a good impact on ride savings for almost 33-38% of rides are saved when delay is increased from 0% to 10%. We see the same trend for an increase to 20% from 10%.

*5.2 Test Results for **trips saved per pool** (July 2015 to June 2016)*

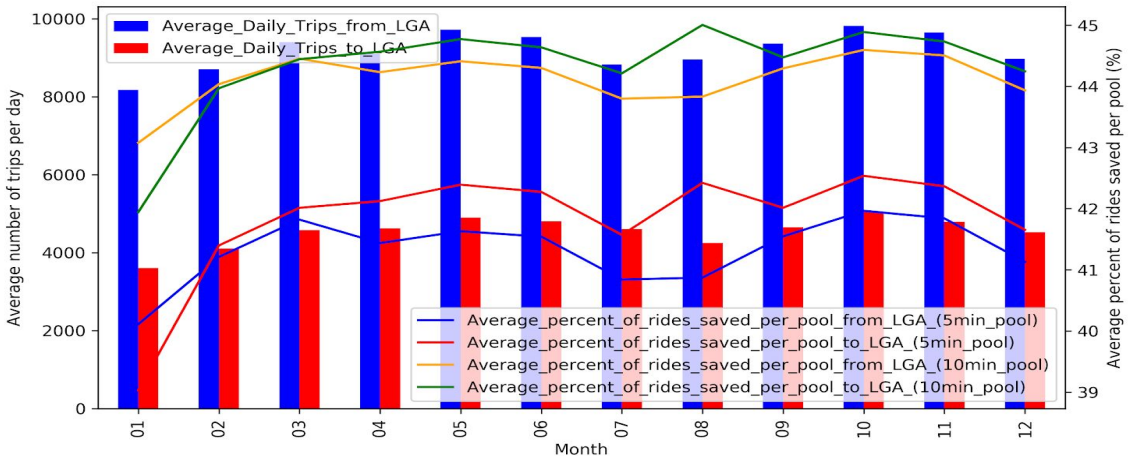
*5.2.1 Average Number of **Trips** Saved per Pool (%) (with Delay 20%)*

The graphs for the number of trips saved are **in-line with the graphs for distance saved** as expected. Again, the number of trips saved for 'to laguardia' is slightly larger than 'from laguardia'.



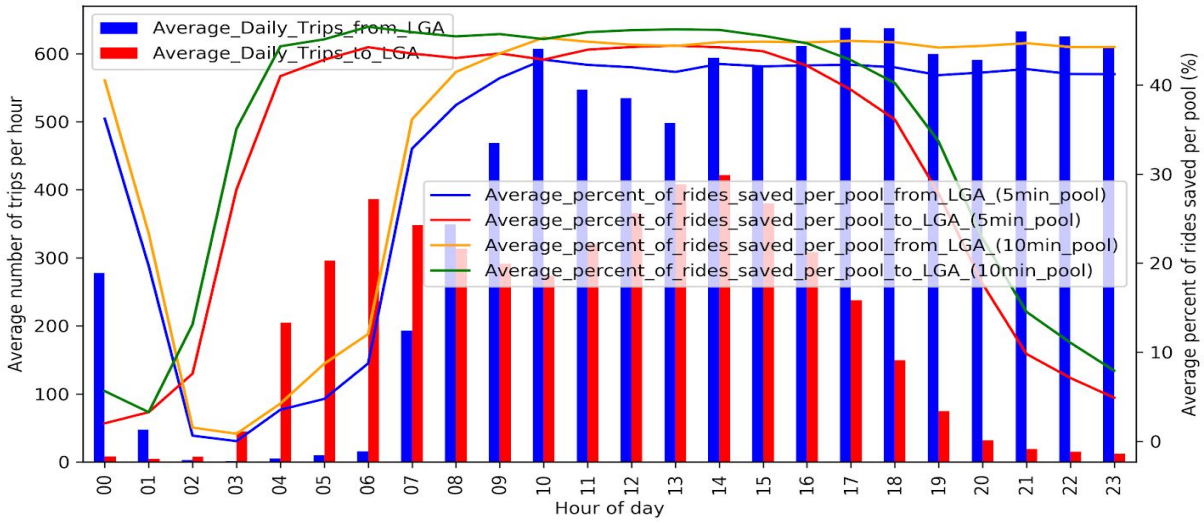
Same as the graphs for distance savings, from the above figure we can depict that as pool time increases the average number of trips saved increases. As mentioned earlier, due to drop off locations from Laguardia going very far, trips saved are a bit higher for 'to LaGuardia'.

*5.2.2 Average Number of **Trips** Saved per Pool (%) (with Delay 20%) **Monthly Breakdown***



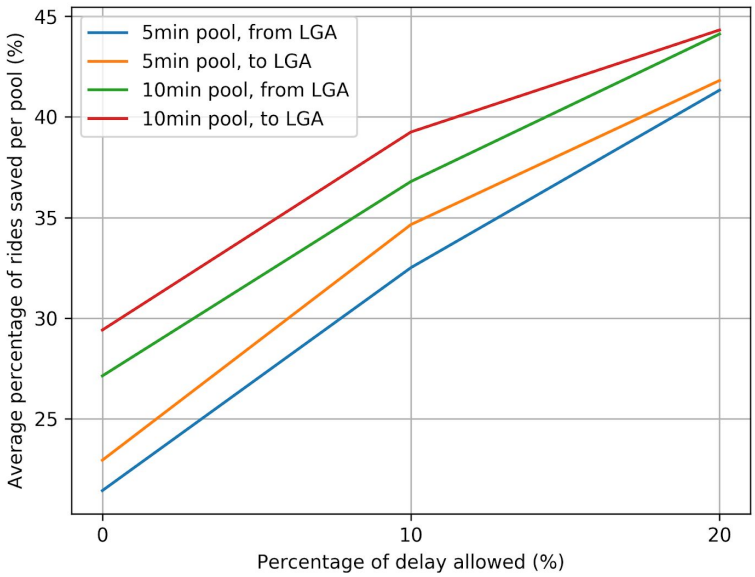
The above figure is almost similar to figure 5.1.2 and the same insights were observed here as well.

*5.2.3 Average Number of **Trips** Saved per Pool (%) (with Delay 20%) **Hourly Breakdown***



The above figure is almost similar to figure 5.1.3 and the same insights were observed here as well.

*5.2.4 **Trips** Saved for **delay**- 0%, 10% and 20%*

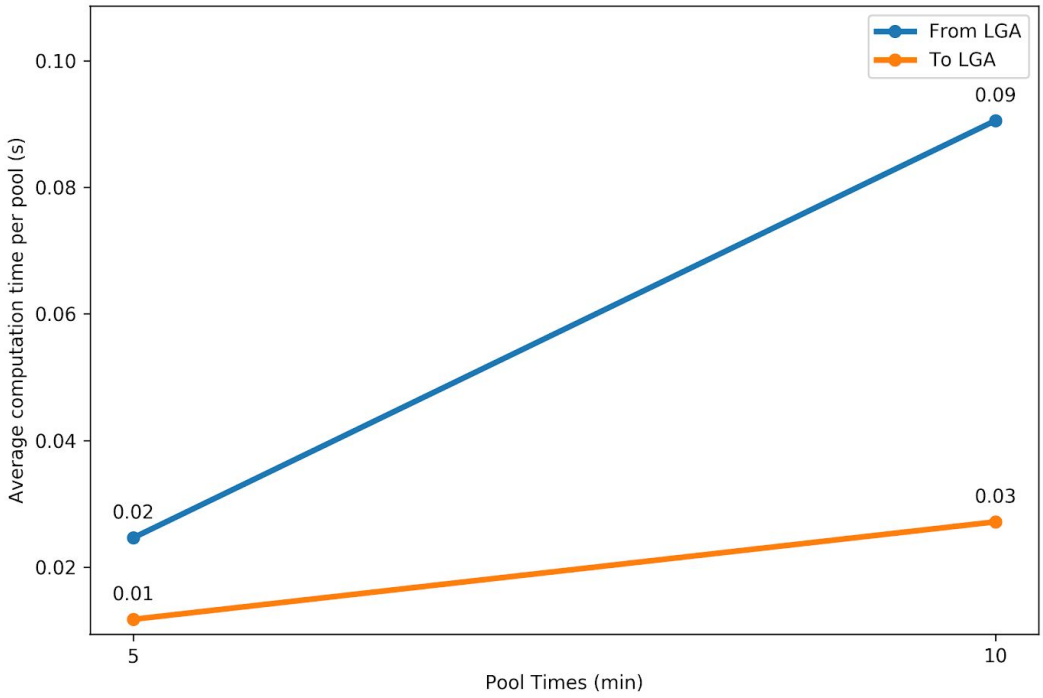


In the above graph, we can observe that delay time has a good impact on ride savings for almost 37-39% of rides are saved when delay is increased from 0% to 10%. We see the same trend for an increase to 20% from 10%.

*5.3 Test Results for **Computation Time per pool** (July 2015 to June 2016) (s)*

5.3.1 Average Computation Time per pool (s)

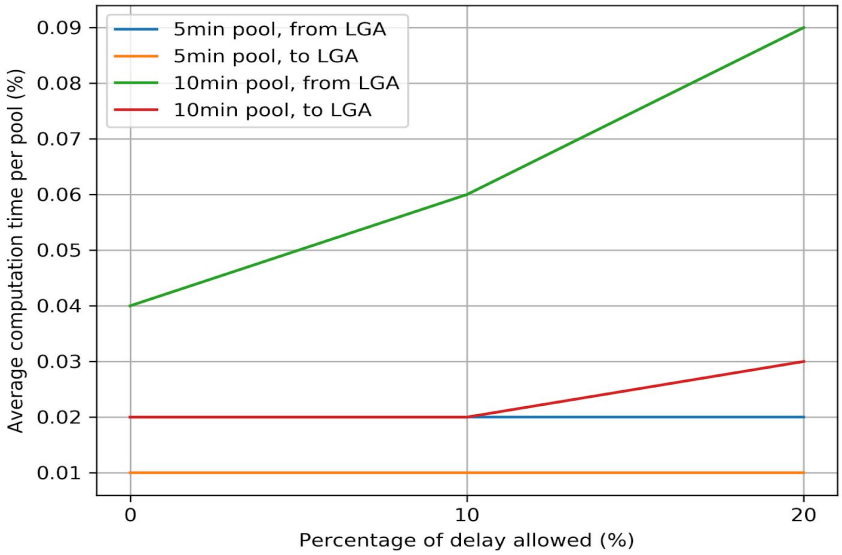
The computation time increases as the pool size increases. The reason being there are more number of ride sharing requests to process in larger pools.



Above graph clearly states that for a 5 min pool it takes only 0.01 seconds to compute the results which indeed is very fast as we did some precomputation. The highest running time it takes is 0.09 seconds for 10 min pool from Laguardia

*5.3.2 **Computation time** for **delay**- 0%, 10% and 20%*

The more the delay tolerance, the more the possibility of rides being shared. The **computation time increases with delay tolerance** for 10 min pools as 10 min pools have more trips to process and in turn more possibility of rides being shared.



In the above graph we can see that the computation time increases as the delay and pool size increase. The seemingly horizontal lines represent linear increase when we convert the metrics into milliseconds.

6. Issues Raised during Final Presentation

No issues were raised during the Final presentation. The issues raised during the demo were addressed in the Final Presentation.

7. Collaboration and Project Management

7.1 Team Members Responsibilities

Responsibilities	Deadline	Collaboration
Data Cleaning and Extraction	02/20/2020	Srikanth,Deblina
Shortest paths precomputation; Initial result from algorithm	03/05/2020	Peihong,Deblina
Results from different algorithms and algorithm finalization	03/10/2020	Peihong,Srikanth
Analysis/Visualization	03/30/2020	All members
Additional Changes/modifications	04/15/2020	All members
Final presentation	04/29/2020	All members
Report	05/04/2020	All members

7.2 Collaboration Tools Used- What worked and what didn't

All of the following tools have been extremely handy and useful. Can't really think of any collaboration tool that didn't do any favour.

- We used **github** to share **code**. Github has always been pretty useful for colab with coding.
- In order to have a **discussion** regarding the project or any issues, we used **zoom and whatsapp.**
- To create slides for the **presentation**s, we used **Google Slides.**
- We used Google docs to write the final report simultaneously.

7.3 How was the project affected by Lockdown and Online Class

We once again realized that being humans, we are merely social beings. Nothing can beat meeting in person and having a discussion as it includes gestures, expressions and more ways to share ideas compared to having a video call. But we had to cope up with this unprecedented situation somehow. Online classes didn't have much impact on the final project though. The professor was more comprehensive with what is expected as the final outcome and what not. We had some issues with the BB collaboration tool though as we were using it for the first time. Two of the members faced Microphone issues but that was resolved later.

8. References

Piazza course resources: https://piazza.com/uic/spring2020/cs581/resources
GraphHopper Directions API https://docs.graphhopper.com/
Jupyter Notebook https://jupyter.org/
OpenStreetMap https://www.openstreetmap.org