Chris Xu

Mitchel Cole

Luciano Arango

# Modeling Taxi Driving as an MDP

## Introduction:

An Uber driver picks up a customer in East Village and drops him off in Washington Heights. Now the driver is faced with a choice: either stay in Washington Heights and wait for the next customer, or drive to another neighborhood to look for more customers. If he stays in Washington Heights, there might be less demand for rides than Harlem. But if he chooses to drive to Harlem, he has to pay the fuel cost of driving to Harlem. What is the optimal action for the driver to take? Drive elsewhere or wait it out?

This is a non-trivial problem that the millions of Uber, Lyft, and taxi drivers across the country face at the end of every single ride. We believe the problem can be represented as a Markov Decision Process, and the Uber driver ultimately cares about the optimal policy. Given that he is in Washington Heights, what should he do? In this project, we use 2015 NYC taxi data to model the states, actions, rewards, and transition probabilities of an MDP. The ultimate goal of this project is to solve for the optimal policy, using the value iteration algorithm we learned in this course. By doing so, we would essentially be creating a valuable practical tool that could be used by Uber/taxi drivers to optimize their driving policy.

A paper from MIT (Agussurja, Lau) discusses a similar topic, but uses a partially observable MDP where the customer success rate is derived from a Poisson process. They based their model on taxi data from Singapore. We did not base any of our algorithms on their algorithms, which are designed for a partially-observable MDP. In their model, every agent has

a belief state, and they use the “standard dynamic programming algorithm” for solving general POMDP’s.

Implementation/Algorithm:

First, we randomly sampled 36,000 trips from the yellow cab data from January to June 2015 (the most recent data available). This is freely available on the NYC Taxi and Limousine Commission website. We downloaded the monthly data from January to June, and imported it into python using Pandas. We had to use sampling because the data from each month were upward of 2.5 GB in size, and it was infeasible to use all of the data.

The next step was to take each trip’s latitude/longitude data for pickup and dropoff, and convert that into a new column of “Pickup Neighborhood” and “Dropoff Neighborhood”. This was a fairly time-consumptive step, because we essentially had to hard-code the latitude/longitude conditions that define each neighborhood. Essentially we had to write code analogous to, “If latitude/longitude in this certain range, then ‘Laguardia’.... If latitude/longitude in this certain range, then ‘Midtown’”. Defining the specific latitude/longitude range of a neighborhood was time-consuming because we had to manually look those up on Google Maps for all nineteen states. It should be noted that we first tried to use the geopy python library which lets you look up a neighborhood given a latitude/longitude location, except the module would time-out after a few thousand requests, so it wasn’t scalable for our sample size of 36,000. Finally, we exported our sampled and cleaned data into a final usable csv file.

The nineteen neighborhoods we defined above at a certain time (to the nearest half hour) would be the states of our MDP.

Our total action space is {commit to current neighborhood, drive north, drive east, drive south, drive west}. Within this action space,  there are two classes of action in our model: deterministic actions (“drive to an adjacent neighborhood” takes you to a specific adjacent

neighborhood) and nondeterministic action (“commit to current neighborhood and wait for a ride” can send you to any neighborhood, depending on where the customer wants to go). The rationale for this setup is based in part on the task graph used in the MAXQ reinforcement learning algorithm for the Taxi MDP by Dietterich (fig. 2).

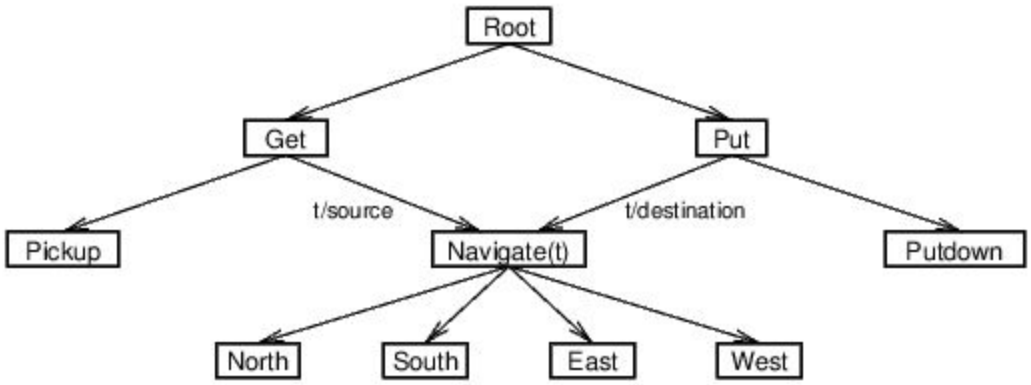



Figure 2: A task graph for the Taxi problem.

In this scenario, however, each action takes one timestep, while in our model, “pickup”, navigating to destination, and “putdown” all occur in one timestep as a timestep in our model is half of an hour.

Let’s revisit the deterministic class of actions first. This class consists of the actions {drive north, drive east, drive south, drive west}. Naturally, these actions have deterministic transition probabilities. For example, if I’m in Midtown and I take action “drive east”, I’ll end up in Midtown East with probability 1. If I’m in East Village, and I try to take action “drive east”, I’ll end up in East Village again (in real life, you’ll end up in swimming in the East River), because there is no neighborhood that is geographically east adjacent of East Village. The deterministic transition matrix for these deterministic actions was hard-coded based on the actual physical geographies of the various neighborhoods.

Now we will discuss the nondeterministic action. If we commit to the current neighborhood, the transition probabilities are based on the data. If I commit to Washington Heights, we can use the historical percentage of trips originating from Washington Heights and ending in Midtown, East Village, etc. to estimate the transition probability from Washington Heights to other neighborhoods. We created this deterministic transition matrix by combing through our csv file, and implementing a double dictionary. The primary key would be the pick-up neighborhood (i.e. Washington Heights). The corresponding value would be another dictionary that keeps track of the count of dropoffs for each of the other neighborhoods (for all trips originating from Washington Heights). Knowing these counts for dropoff neighborhoods, we just normalized to arrive at our transition probability.

Finally, we use average distance between neighborhoods as reward, with distance being a proxy for fare (in the case of giving a ride) and fuel cost (in the case of driving to another neighborhood). We calculate these average distances between neighborhoods in the same double dictionary as mentioned above. Let's say, for example, that we calculated that the average distance of a trip from Midtown to Midtown East was 0.3 miles. Then, the reward for R(Midtown, Commit, Midtown East) is 0.3. The reward for R(Midtown, Drive East, Midtown East) is -0.1. The actions from the class of deterministic actions (driving to another neighborhood) have negative reward because the driver is not generating revenue and paying fuel costs. The nondeterministic action has positive reward because the driver is generating revenue. Lastly, if the driver commits to a location and does not pick up a customer (a plausible scenario) there is also a negative reward of -0.1.

Putting it all together, let us explicitly define the variables in the Bellman equation used to solve our MDP using Value Iteration:

$$U(s) = \max_{a \in ACT(s)} \sum_{s' \in S} P(s'|s,a)(R(s,a,s') + \gamma \times U(s'))$$

The probability P is given by our transition model, where driving actions (other than commit) occur with probability one, and the commit action uses frequencies of rides originating from s and ending at s' to calculate P. Next, our reward function R is the average distance of trips originating from s and ending at s' from our 2015 yellow cab taxi data. This can serve as a proxy for the fare gained for the taxi driver if such a trip were to occur. Moreover, for the driving actions, the reward is defined as a constant, -0.1 km, which denotes a penalty for not having a customer in the taxi for the hour in question. Therefore, the value U of each state, denoted in km, designates the relative value of states, where a higher value signifies a greater distance travelled with a passenger and thus increased cab fares.

Analysis of Implementation/Limitations and Next Steps:

For future directions, we would like to better approximate the reward for the taxi driver. Currently we have the reward as the distance travelled between pickup state and dropoff state. However, it seems like the most applicable metric for "reward" would be the fare of the ride. However, this would require us to incorporate traffic and time to reach destination in order to calculate the value of the state. We could also consider fuel prices which would be applicable to the negative reward that comes from driving to another neighborhood without a customer.

Another significant limitation of our model is that we don't have data for "unsuccessful rides" - rides in which the driver committed to stay in Washington Heights but failed to pick up a customer. In other words, we do not have data pertaining to the amount of time and the location for which a driver did not have a passenger. We used "Percentage of total Manhattan rides that originate from Washington Heights" as a proxy for the likelihood of getting a ride from Washington Heights. This might not be the best proxy: consider the case in which taxi drivers

never have trouble picking up customers (100% success rate), but that the vast majority of taxi rides originate from Midtown because of more demand. Therefore, in the future we might be able to do better if we had data on “unsuccessful rides” as well.

In the future, we would like to implement more temporal granularity, and build models that incorporate day of the week/season of the year in the state space. Moreover, Midtown at 10pm on a Saturday night will be different than Midtown at 10pm on Sunday. Finally, Midtown during the height of the summer tourist season (July) should behave differently than Midtown during February.

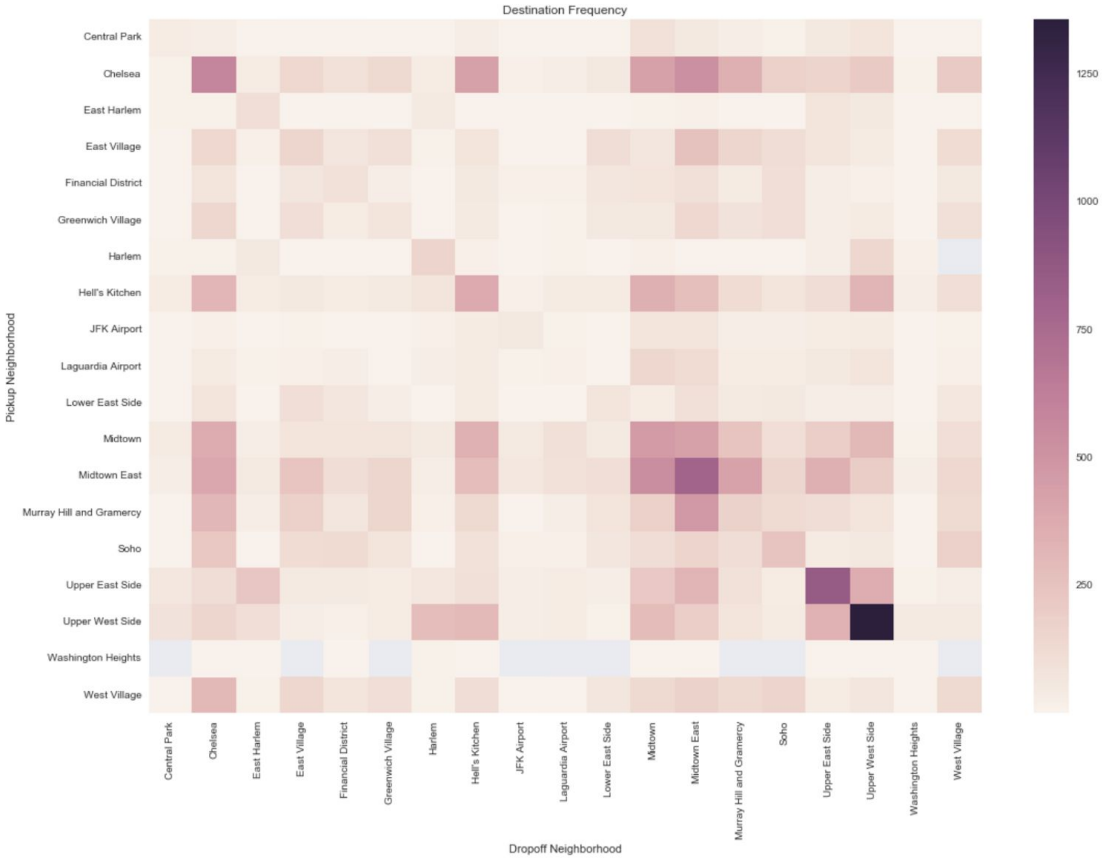
In addition, we could potentially implement a model free approach for locations with limited data, such as Boston, as a model based approach requires significant amount of data to build an accurate model. However, with model free approaches such as $\epsilon$-Greedy Q-learning, or even approximate Q-learning, we could continuously update our policies based on new experiences. For example, in approximate Q-learning, we could use linear feature functions (Berkley AI), where features could include time, location, traffic, is the day a weekend?, fuel remaining, etc. By constantly adjusting the weights, or value, of each feature we could more quickly obtain policies through a limited number of experiences.

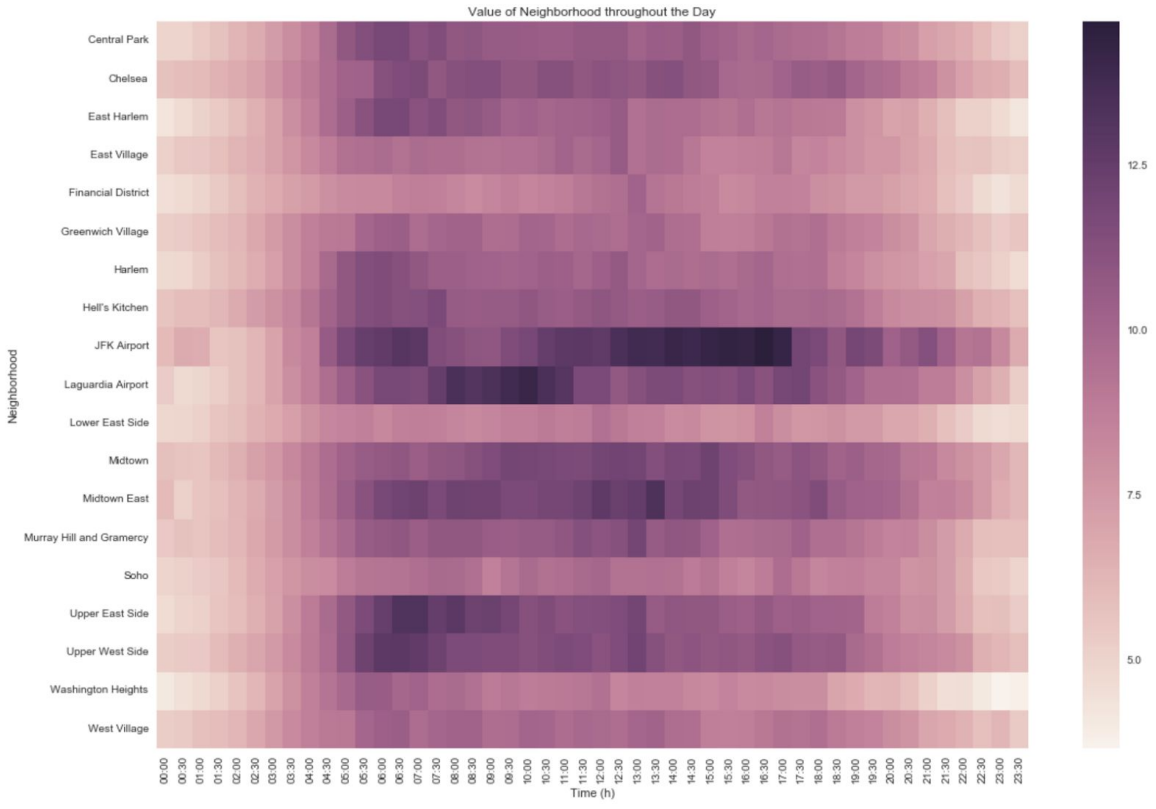
**Appendix 1**

**interface.py** is a program that takes in user input, in the way of Neighborhood and Time, and finds the action that has the highest value. The values are a dictionary that our value iteration has generated. The *italicized* text below is the user input.

```
$ python interface.py
Creating MDP
Running Value iteration
What neighborhood are you currently in: Upper East Side
Current time (HH:MM):   4:30
West
What neighborhood are you currently in: Upper West Side
Current time (HH:MM):   5:21
North
What neighborhood are you currently in: Upper West Side
Current time (HH:MM):   15:21
Commit
What neighborhood are you currently in: East Harlem
Current time (HH:MM):   16:12
South
What neighborhood are you currently in: Harlem
Current time (HH:MM):   19:46
South
What neighborhood are you currently in: Harlem
Current time (HH:MM):   4:06
East
What neighborhood are you currently in:
```

In **data.ipynb** we have created code to generate visualizations for drivers to see the value of neighborhoods at different times as well as a table that drivers can conveniently use to determine where to go based on the current time and they neighborhood they are currently in; This is essentially finding the value for every neighborhood for every time slot.


Destination Frequency
Pickup Neighborhood
Central Park
Chelsea
East Harlem
East Village
Financial District
Greenwich Village
Harlem
Hell's Kitchen
JFK Airport
Laguardia Airport
Lower East Side
Midtown
Midtown East
Murray Hill and Gramercy
Soho
Upper East Side
Upper West Side
Washington Heights
West Village
Dropoff Neighborhood
1250
1000
750
500
250


Value of Neighborhood throughout the Day
Neighborhood
Central Park
Chelsea
East Harlem
East Village
Financial District
Greenwich Village
Harlem
Hell's Kitchen
JFK Airport
Laguardia Airport
Lower East Side
Midtown
Midtown East
Murray Hill and Gramercy
Soho
Upper East Side
Upper West Side
Washington Heights
West Village
Time (h)
12.5
10.0
7.5
5.0

# Current Neighborhood

| Time | Central Park | Chelsea | East Harlem | East Village | Financial District | Greenwich Village | Harlem | Hell's Kitchen | JFK Airport | Laguardia Airport | Lower East Side | Midtown | Midtown East | Murray Hill and Gramercy | Soho | Upper East Side | Upper West Side | Washington Heights | West Village |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00:00 | South | North | South | North | West | North | South | South | Commit | Commit | North | West | Commit | West | North | South | South | East | North |
| 00:30 | South | North | West | North | West | North | South | South | Commit | West | North | West | South | West | North | South | South | East | North |
| 01:00 | South | North | West | North | East | North | South | South | Commit | West | North | West | West | West | North | South | South | East | North |
| 01:30 | South | North | West | North | West | North | South | South | West | West | North | West | West | West | North | South | South | East | North |
| 02:00 | South | North | West | North | West | North | South | East | West | West | West | West | West | West | North | South | South | East | North |
| 02:30 | South | North | West | North | West | North | South | South | West | West | West | West | West | West | North | South | South | East | North |
| 03:00 | South | North | West | North | West | North | South | South | West | West | North | West | West | West | North | South | South | East | North |
| 03:30 | South | North | South | North | West | North | South | North | West | West | North | West | West | West | North | South | South | East | North |
| 04:00 | North | North | West | North | East | North | East | North | West | West | North | West | North | West | North | West | South | East | North |
| 04:30 | West | North | West | North | West | North | South | North | West | West | North | North | North | North | North | West | North | East | North |
| 05:00 | West | North | South | North | West | North | South | North | Commit | West | North | North | North | North | North | West | North | East | North |
| 05:30 | West | North | South | North | West | North | South | North | West | West | North | North | North | North | North | North | Commit | East | North |
| 06:00 | East | North | South | North | West | North | South | North | West | West | North | East | North | North | North | Commit | Commit | East | North |
| 06:30 | East | Commit | South | North | West | North | South | North | West | West | North | East | North | North | North | Commit | Commit | East | North |
| 07:00 | East | North | South | North | West | North | South | North | Commit | West | West | East | North | North | North | Commit | Commit | East | North |
| 07:30 | East | Commit | South | North | West | North | South | North | West | West | West | East | North | North | North | South | Commit | East | North |
| 08:00 | East | Commit | South | North | West | North | South | North | West | Commit | West | East | North | North | North | South | Commit | East | North |
| 08:30 | East | Commit | South | North | West | North | South | East | West | Commit | North | East | North | North | Commit | South | Commit | East | North |
| 09:00 | South | Commit | South | North | West | North | South | East | West | Commit | North | East | West | North | North | South | Commit | East | North |
| 09:30 | South | North | South | North | West | North | South | East | Commit | Commit | West | East | West | North | North | South | Commit | East | North |
| 10:00 | South | Commit | South | North | West | North | South | East | Commit | Commit | North | East | West | North | Commit | South | Commit | East | North |
| 10:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | North | East | West | North | North | South | Commit | East | North |
| 11:00 | South | East | South | North | West | North | South | East | Commit | Commit | West | East | West | North | North | South | Commit | East | North |
| 11:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | North | East | West | North | Commit | South | Commit | East | North |
| 12:00 | South | East | South | North | Commit | North | South | East | Commit | Commit | North | East | West | North | Commit | South | Commit | East | North |
| 12:30 | West | East | South | North | Commit | North | South | North | Commit | West | South | East | North | North | South | South | Commit | East | North |
| 13:00 | South | Commit | South | North | Commit | North | South | East | Commit | Commit | North | East | Commit | North | North | South | Commit | East | North |
| 13:30 | South | Commit | South | North | Commit | North | South | East | Commit | Commit | North | East | Commit | North | Commit | South | Commit | East | North |
| 14:00 | South | Commit | South | North | Commit | North | South | East | Commit | Commit | North | East | West | North | Commit | South | East | East | North |
| 14:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 15:00 | South | Commit | South | North | Commit | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 15:30 | South | North | South | North | West | North | South | East | Commit | Commit | North | Commit | West | North | Commit | South | Commit | East | North |
| 16:00 | West | North | South | North | West | North | South | North | Commit | Commit | West | East | West | North | North | South | Commit | East | North |
| 16:30 | West | Commit | South | North | West | North | South | North | Commit | Commit | West | East | Commit | North | North | South | Commit | East | North |
| 17:00 | South | East | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 17:30 | South | Commit | South | North | Commit | North | South | East | Commit | Commit | North | East | West | North | North | South | Commit | East | North |
| 18:00 | West | Commit | South | North | West | North | South | North | Commit | Commit | West | East | North | West | North | South | Commit | East | North |
| 18:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 19:00 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 19:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 20:00 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 20:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | West | Commit | South | Commit | East | North |
| 21:00 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 21:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | Commit | South | Commit | East | North |
| 22:00 | South | Commit | South | North | Commit | North | South | East | Commit | Commit | North | East | West | North | Commit | South | Commit | East | North |
| 22:30 | South | Commit | South | North | West | North | South | East | Commit | Commit | West | East | West | North | North | South | South | East | North |
| 23:00 | South | Commit | South | North | West | North | South | East | Commit | Commit | North | Commit | West | North | Commit | South | Commit | East | Commit |
| 23:30 | South | Commit | West | North | West | North | South | East | Commit | Commit | North | East | Commit | North | North | South | South | East | North |

**Appendix 2**

**interface.py:**

To run this program simply navigate to the directory where it is located and run `python interface.py`. The program will take a few seconds to build the MDP model and run value iteration. Then it will ask for your current neighborhood, you may input any of the following: Washington Heights, Chelsea, Hell's Kitchen, Midtown, Midtown East, Murray Hill and Gramercy, East Village, West Village, Greenwich Village, Financial District, Lower East Side, Soho, Central Park, Laguardia Airport, JFK Airport, Upper East Side, Upper West Side, Upper West Side, East Harlem, Harlem. Then it will ask you for the time, which should be in HH:MM format example (5:30, 6:34, 15:45). The program will complain if it is unable to parse your time or can't find recognize neighborhood inputted. It will then spit out the direction where you should head which is one of the following: South, North, West, East, or Commit.

**data.ipynb:**

To run this program, one must navigate to the directory where data.ipynb located and run `ipython notebook data.ipynb`. An internet browser will open, click on the first cell and then above on the tab that says "Cell" and choose the option to "Run All Below". There are also instructions on the file itself.

* You must have the correct python libraries installed

## Appendix 3 –

Mitchel Cole: Built nondeterministic transition using double dictionary, incorporated time (hours of the day) into the model/state space, worked on reward function and debugging value iteration

Chris Xu: Built deterministic transition matrix, wrote project proposal and update, worked on reward function and initial value iteration implementation, helped Luciano with data cleaning

Luciano Arango: Lead role on data cleaning (pulling the data and creating “dropoff neighborhood” and “pickup neighborhood” fields), worked on user interface and data visualization

References:

Lucas Agussurja , Hoong Chuin Lau, A POMDP model for guiding taxi cruising in a congested urban city, Proceedings of the 10th Mexican international conference on Advances in Artificial Intelligence, November 26-December 04, 2011, Puebla, Mexico

http://ares.lids.mit.edu/fm/documents/pomdp.pdf

New York City Taxi Cab Data

http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

Dietterich, Thomas G. "Hierarchical reinforcement learning with the MAXQ value function decomposition." J. Artif. Intell. Res.(JAIR) 13 (2000): 227-303.

https://inst.eecs.berkeley.edu/~cs188/fa10/slides/FA10%20cs188%20lecture%2012%20--%20reinforcement%20learning%20II%20(6PP).pdf

## Link to Poster

Taxi MDP Poster