Determine Bikes Allocations Between Stations using Policy Iteration

Chengyu Deng (cdeng@bu.edu)
Department of Computer Science, Boston University

CS 504
Data Mechanics
Spring 2019

Introduction

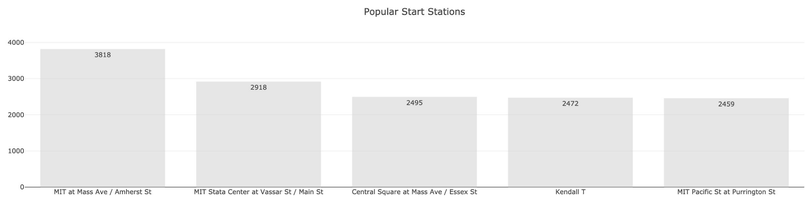
- Bike riding is very popular in the Boston area.
- Biking sharing services in the Boston area like BLUE Bikes handle thousands of bike trips each day.
- Problems of using dock stations for bike sharing:
  - Bike trips begin/end locations are fixed.
  - The number of docks per station is limited. (The main focus of this project).
- Simulate a bike moving solution between stations to solve “docks full” and “no bike” problems and maximize bike utilization.

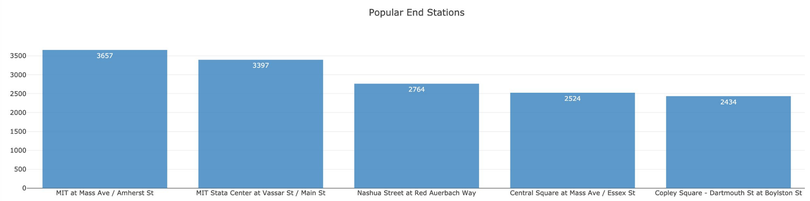
Data Sources

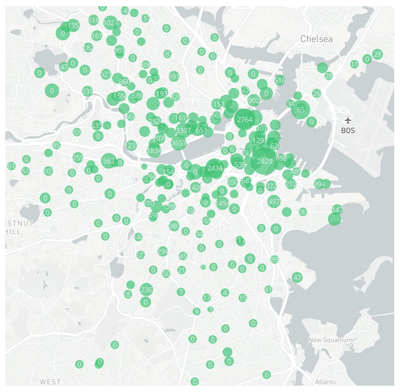
- BLUE Bikes dock station information datasets in Boston, Brookline, Cambridge, and Somerville. The datasets contain the station street names, coordinates, and the number of docks.
- BLUE Bikes trip history datasets from January, February 2018 in those four cities above. The datasets contain every begin/end trip information of each user in that time period.

Data Visualization

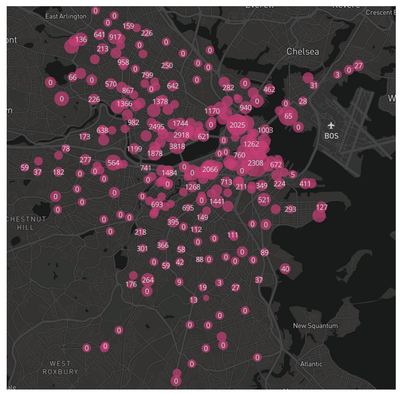
- Find the number of trips for each station based on the datasets.
- The numbers shown in the charts and maps below are calculated through relational model building blocks using MongoDB and results are stored in the database.







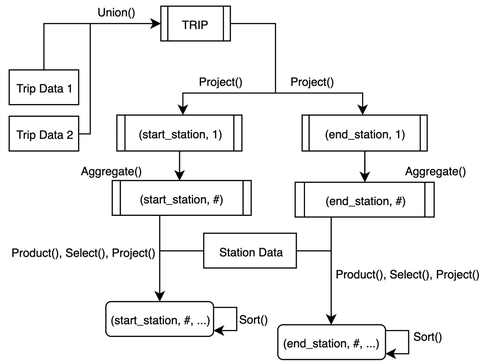
Incoming trips map



Outgoing trips map

Methodologies

- ***Data Extraction***
  - Use the building relational building blocks to process data and sort the data to find out popular stations.
  - Store the results into MongoDB collections for further usages such as optimization problems and statistical analysis.



- ***Policy Iteration Algorithm***
  - Assumptions:
    - Select the most 2 popular stations for allocation optimization.
      - MIT at Mass Ave / Amherst St: station 1,
      - MIT Stata Center at Vassar St / Main St: station 2
    - Move bikes from one station to the other every two hours.
    - A reward ***R*** is given for every bike rental.
    - A cost ***C*** is given for moving a single bike.
    - For each time, at most ***m*** bikes can be moved.
    - The average bike incoming/outgoing rates for each station are calculated using the data obtained above. The number of incoming and outgoing bikes every two hours follows a Poisson Distribution.
  - Model Formalism: Markov Decision Process (MDP):
    - **State**: A tuple that contains both stations’ available bikes number.
    - **Action**: Moving of ***n*** bikes ($0 < n \leqq m$) at a state. Actions are in [-n, n].
    - **State Transition Matrix**: Probability of a transition from a state to another state. It is based on the stations’ incoming/outgoing bikes probability distribution.
    - **Reward Function**: Total net rewards for move bikes at a state.
    - **Gamma(γ)**: A discount factor from 0 to 1 for calculating future rewards in the main algorithm.
  - Policy Iteration Algorithm:
    - An iterative optimization algorithm.
    - A policy π: an action for a certain state.
    - **State value *V***: An expected value for future reward at a state.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s') \quad \text{(Bellman Equation)}$$

Target: find the ***optimal state values*** for each state so that the total rewards for bike rental are maximized and the bikes for both stations are fully utilized given the assumptions.

- ***Pseudocode of Policy Algorithm for bike allocations***

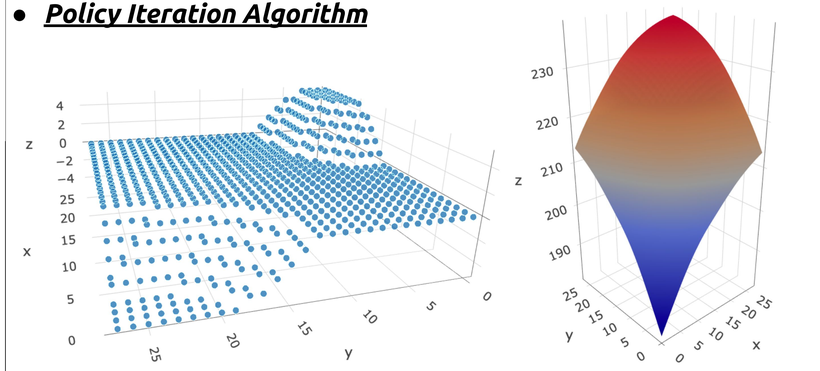
```
Initialize policy(s)=0 and state value v(s)=0 for each state s;
do:
| do:
| | For each possible incoming rate/outgoing rate p, q:
| | | For each state s:
| | | | For each action:
| | | | | Update available bikes for each station to get s’ and v(s’);
| | | | | Based on the action, cost, and incoming rate, get net reward;
| | | | | Update current v(s) using the Bellman Equation of v(s);
| While (total change of state value > ε which is a small number);
| Greedy update of policy for each state;
While (policy is still changing);
```

Methodologies (Continued)

- ***Statistical Analysis***
  Calculate the correlations and p-values between:
  - Number of docks for each station and incoming bike trips.
  - Number of docks for each station and outgoing bike trips.
  - Incoming bike trips and outgoing bike trips for each station.

Results

- ***Policy Iteration Algorithm***



The x and y axis denote the number of bikes at station 1 and station 2. The tuple (x, y) represents a state. The z axis indicates the policy option on the left graph and the state value on the right graph.

- ***Statistical Analysis***

Cases	# of docks vs incoming trips	# of docks vs outgoing trips	Incoming trips vs outgoing trips
Corr Coefficient	0.4234	0.4508	0.9916
P-Value	0.0	0.0	0.0

Conclusion and Future work

- P-values for those three cases are all 0, indicating that they are truly correlated based on the correlation coefficients that we calculated from the data.
- Policy Iteration returns an optimal solution for bike allocations for the two most popular stations.
- It is still an open question that how to find the optimal allocation strategy for all bike stations. It is challenging since the number of stations is quite large. One possible future work is to extend the algorithm to handle all stations. One potential solution is using a neural network to predict a state value instead of searching through all of them in the algorithm.
- The Poisson Distribution assumption can be examined using an advanced statistical method in the future.

Reference

- https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node20.html
- https://en.wikipedia.org/wiki/Markov_decision_process
- https://medium.com/@m.alzantot/deep-reinforcement-learning-demystified-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa
- https://youtu.be/Nd1-UUMVfz4
- https://www.bluebikes.com/system-data
- http://cs-people.bu.edu/lapets/504/s.php#2
- https://plot.ly/javascript/