

# Reinforcement Learning of Taxi Driver Passenger-seeking Strategies

Haochen Tang

## Introduction

A decision-making system, such as a robot, takes input information from sensors (location, velocity, environments and etc.) and select certain action among multiple alternatives based on its own policy and its current states. After a taxi driver drops

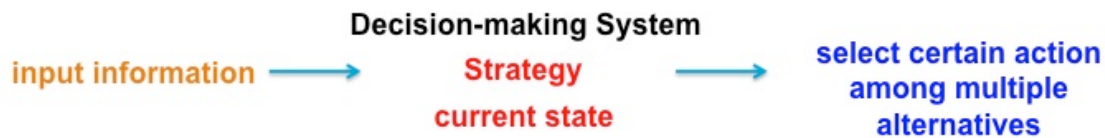


Figure 1. Decision Making System

the previous passenger, he/she needs to decide how to find the next passenger in the most efficient way: take a taxi driver in Beijing as an example, given the drop-off location of the last passenger which is at Beijing University, and all available information such as location, time, traffic, weather and etc., the driver may drive to location A which is another university, or to location B which is a popular tourism place(the summer palace), or to location C which is a shopping center in downtown

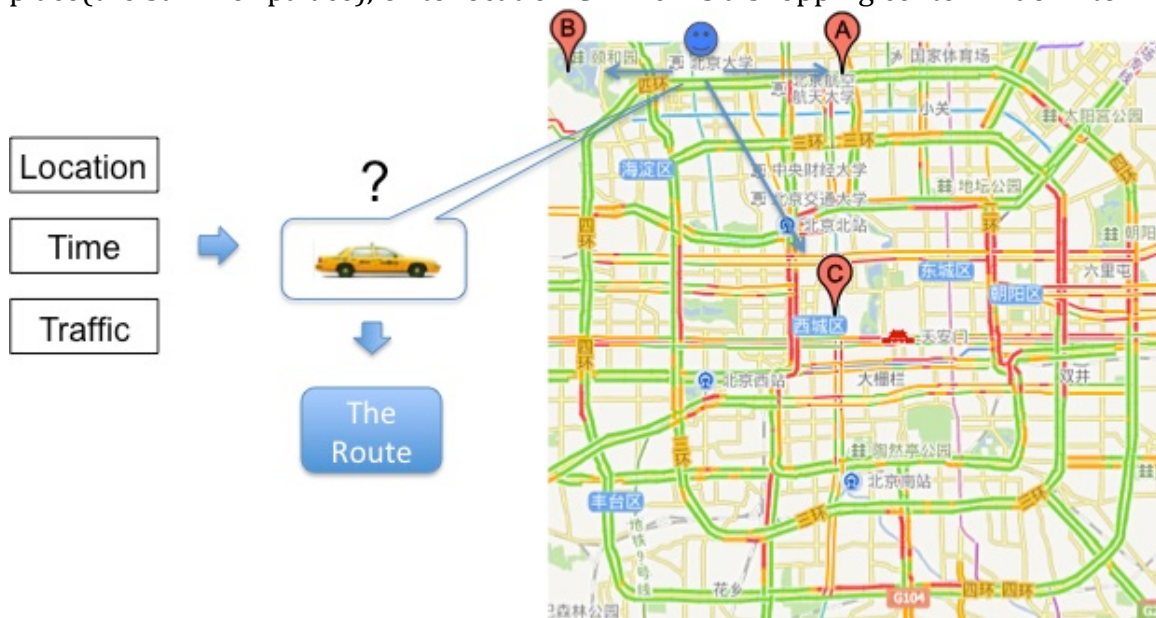


Figure 2. Taxi Driver Decision Making Illustration

Beijing. Therefore in this project, I consider the taxi driver as a special and sophisticated decision-making system when he/she is looking for the next passenger. And I am going to discover the optimal policy when the taxi driver is searching for the next passenger. This is defined as the optimal passenger-seeking strategy. The result will be very helpful to guide taxi drivers to achieve better

earnings as well as reduce fuel cost and hazardous emissions. Moreover, since the self-driving cars have become available, we could expect self-driving taxis in the near future. This result can be used to efficiently manage self-driving taxis systems.

The data used in this project are GPS trajectories from 1000 taxis in the city of Beijing, China from May 1<sup>st</sup>, 2009 – May 31<sup>st</sup>, 2009. The data structure is shown in the right.

```
struct trajectory {
    int id;
    double time;
    double x;
    double y;
    bool isHeavy;
};
```

## Related Works and My Method

Figure 3. Data Structure

A few works have already been conducted to analyze the taxi GPS trajectories. J. Yuan finds all the taxi waiting areas from the historical GPS data and provides the statistics for the waiting areas [1]. Y. Ge discovers the preferred pick-up locations of those high profitable taxi drivers and uses a sequence of the preferred pick-up locations as the recommending driving route [2]. X. Zheng uses an Non-homogeneous Poisson Process to model the characteristics of vacant taxis and estimate passenger waiting time [3].

My method focuses on applying reinforcement learning algorithm on taxi GPS data to reveal the optimal passenger-seeking strategy at different states. Shown in the figure below, when the driver starts a cruise trip (trip that the taxi is vacant) from

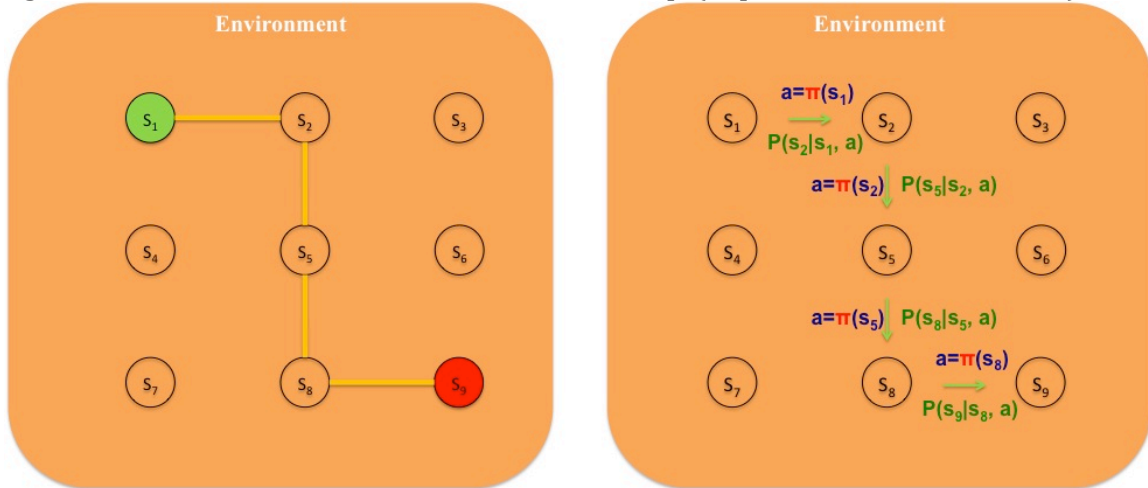


Figure 4. MDP Model

a starting state, he chooses certain action, and with some probability, he will end up in certain state.

The work has two major part: the first part is processing the GPS trajectory data and generate all the parameters needed for the reinforcement learning; the second part is applying the reinforcement learning algorithm to find out the optimal strategy for the taxi drivers.

Figure 6. Occupied and Cruise Trips

Given the trips and the grid, I calculate the number of pick-up events within each of the grid in the one-month period, the average fare income and fuel cost generated from these pick-ups; I also calculate the total cruise time in minutes within each of the grid. Now I would like to build a grid data structure containing all the values I need for the reinforcement learning. The data type for each of the 30×30 grids is shown in Figure 7. The *dropState* data contains the number of drop-off to all 30×30 states from pick-up in this grid, as well as the corresponding expected fare and fuel cost.

```
struct stateGrid {
    int numPickUp;
    double totalCruiseTime;
    double meanFare;
    double meanFuel;
    Grid<Vector<double>> dropState;
};
```

Some parameter values are visualized in Figure 8.

Figure 7. Grid Data type

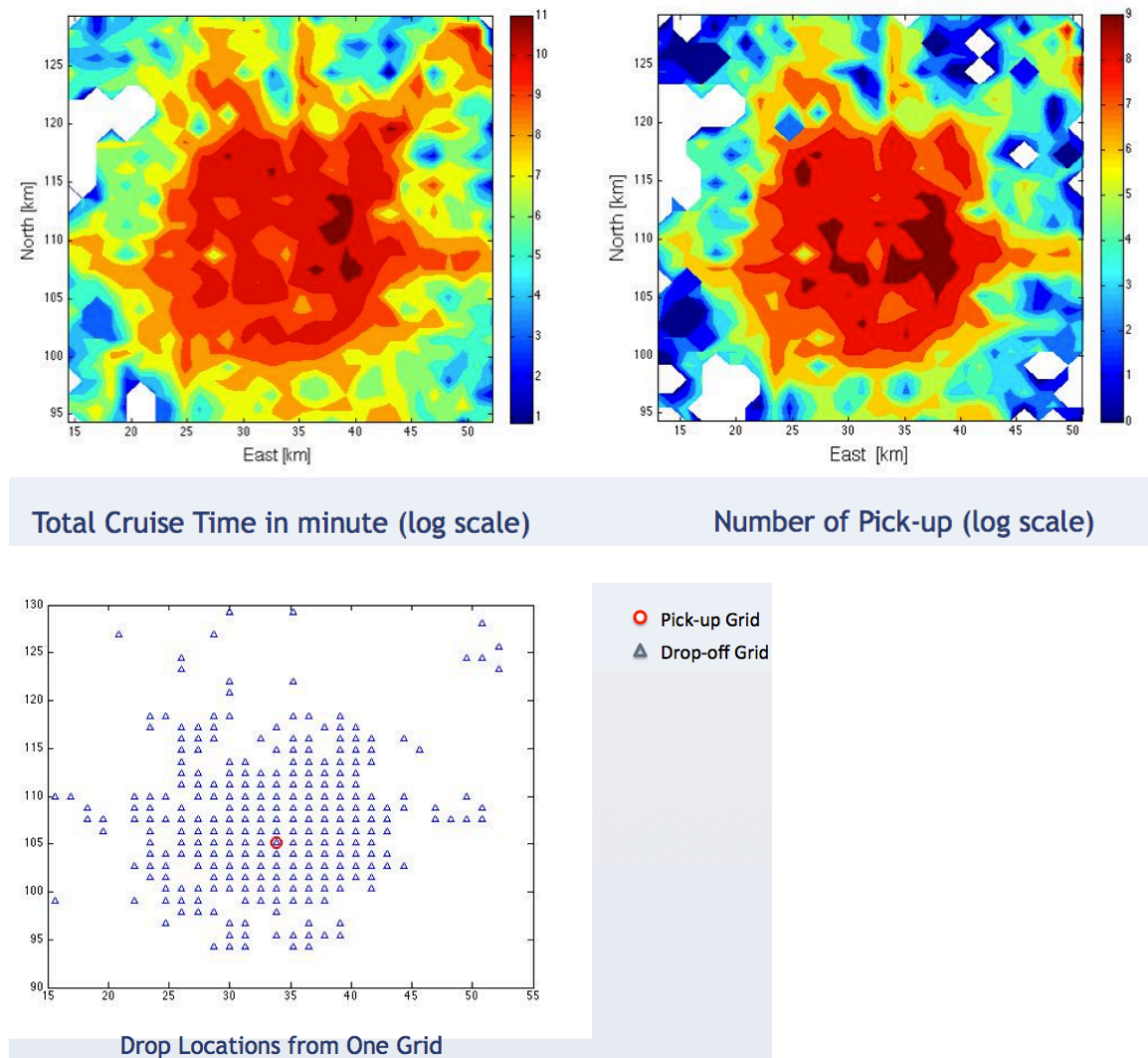


Figure 8. Data Visualization

## MDP and Reinforcement Learning

Now we will use the values we obtained from the data processing to implement the MDP model for reinforcement learning. In my MDP model, the time step is set as 5 minutes. The actions can be made at each grid are driving north, east, south, west to the adjacent grid or stay in the same grid. Every time the driver makes any decision, for the first half time step, he/she will be traveling in that grid, if failed to pick up any passenger, the driver will travel in the adjacent grid or the same grid, depending on the action he/she chooses. Therefore, the pick up probability at grid(i,j) within 1 minute cruise will be:

$$P_{pu\_min}(i,j) = \frac{\#Pickup(i,j)}{TotalCruiseTime(i,j)}$$

And the pick-up probability of every half time step (2.5 minutes) will be (assuming iid):

$$P_{pu}(i,j) = 1 - (1 - P_{pu\_min}(i,j))^{2.5}$$

The probability that after a taxi picks up passengers in grid(i,j), it goes to grid(m,n) is:

$$P_{ij}(m,n) = \frac{\#Drop@Grid < m,n > fromGrid < i,j >}{\#TotalPickup@Grid < i,j >}$$

Then my MDP model is:

- States: 30×30 locations
- Actions: at each grid, driver can go to the adjacent grids in north, east, south, west directions or stay in the same grid
- State Transition Probability from grid(i,j) to grid(m,n):
  - Go North:  $P_{pu}(i,j) \times P_{ij}(m,n) + (1 - P_{pu}(i,j)) \times P_{pu}(i+1,j) \times P_{i+1j}(m,n)$
  - Go East:  $P_{pu}(i,j) \times P_{ij}(m,n) + (1 - P_{pu}(i,j)) \times P_{pu}(i,j+1) \times P_{ij+1}(m,n)$
  - Go South:  $P_{pu}(i,j) \times P_{ij}(m,n) + (1 - P_{pu}(i,j)) \times P_{pu}(i-1,j) \times P_{i-1j}(m,n)$
  - Go West:  $P_{pu}(i,j) \times P_{ij}(m,n) + (1 - P_{pu}(i,j)) \times P_{pu}(i,j-1) \times P_{ij-1}(m,n)$
  - Stay:  $P_{pu}(i,j) \times P_{ij}(m,n) + (1 - P_{pu}(i,j)) \times P_{pu}(i,j) \times P_{ij}(m,n)$
- Reward function:
 
$$R(i,j) = P_{pu}(i,j) \times \{E[Fare(i,j)] - E[Fuel\_Cost(i,j)]\}$$
- $\gamma = 0.95$

And I use the value iteration to get the optimal policy:

- For each state s, initialize  $V(s) = 0$ ;
- Repeat until converge {
  - For every state s, update

$$V(s) := R(s) + \max_{a \in A} \sum_{s'} P_{sa}(s') V(s')$$

}

## Result

The value convergence is shown in figure 9. It takes about 90 time steps to converge. And the optimal policy at each state is shown in figure 10.

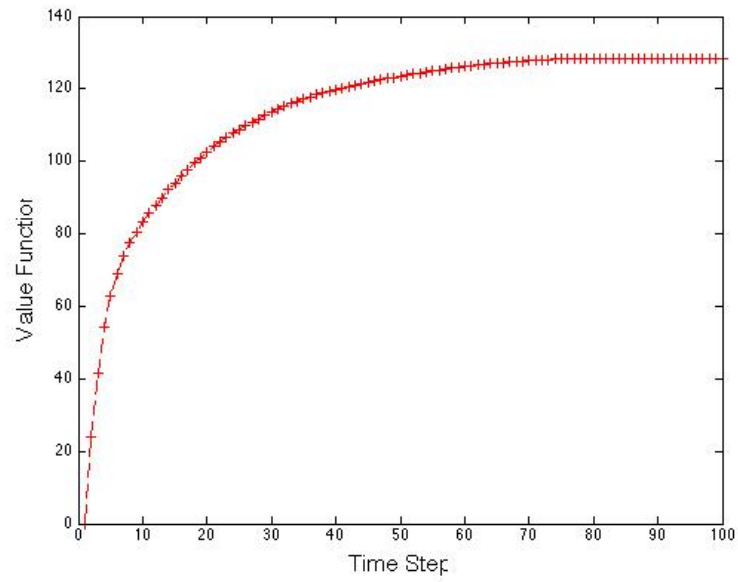


Figure 9. Value Convergence



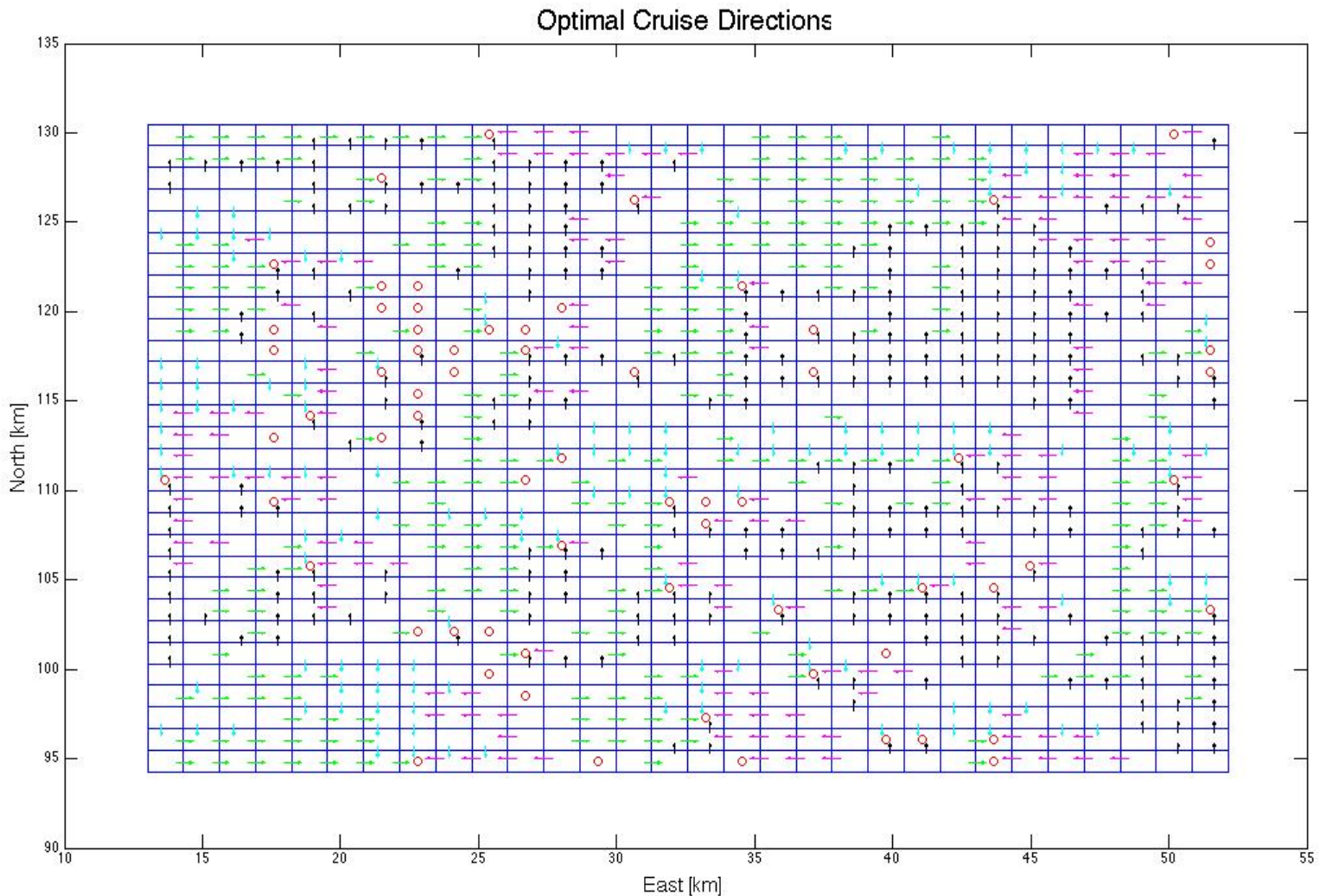


Figure 10. Optimal Strategy at Each Grid

## Future Works

I haven't find a way to validate my result. One method I am considering is to find the high profitable driver from the data and compare their actions at each grid with the optimal policy discovered by the reinforcement learning.

Another interesting question is: if this optimal policy is available to every taxi driver, optimal actions may not be optimal any more since a lot of taxi drivers choose them. I am thinking about applying game theory to solve this problem.

Machine learning is a really amazing technology, thank you so much for introducing it to me, Professor Ng. I enjoyed learning this new knowledge. Please have a nice holiday season!

## References

- [1] J. Yuan and et al., Where to Find My Next Passenger? UbiComp'11, September 17–21, 2011, Beijing, China.
- [2] Y. Ge and et al., A Taxi Business Intelligence System. KDD'11, August 21–24, 2011, San Diego, California, USA.
- [3] X. Zheng and et al., Where to Wait for a Taxi? UrbComp '12, August 12, 2012. Beijing, China.