Uber driver strategic behavior

October 5, 2016

1 Uber Driver strategies

In this study, we compare four kinds of Uber drivers,

1. Naive Uber driver with no strategy.
2. Uber driver with re-location strategy.
3. Uber driver with flexible work schedule strategy.
4. Uber driver with both flexible work schedule and re-location strategy.

In the following sections, we formulate each of the above four problems.

2 Uber Driver with re-location strategy

Note: Need to re-formulate to include the last home trip of Uber driver at the end of the day.

Consider a set of n nodes as vertices of a graph $G = (V, E)$ defined over a geographical area. We are interested in creating an optimal contingency plan of trips, for a particular driver who starts his day at node $v_0 \in V$. We assume that the driver decides the maximum units of service time T that he would spend ferrying passengers at the beginning of the day. Let time t denote the number of time units left for a driver before the end of his pre-decided service time.

- For $i, j \in V, d_{ij}$ is the Euclidean distance between the two nodes. The costs to the driver in terms of gas and vehicle depreciation value in traversing this edge is c_{ij} and his earnings in ferrying a passenger from i to j is represented by p_{ij}. Time required to traverse the edge is denoted by τ_{ij}. The costs, the earnings and the time required to make this ride are all proportional to the distance travelled.
- Let $S^{(t)}_{n\times 1}$ be the surge multiplier vector over the nodes of the graph. Each entry $s^{(t)}_i$ denotes the surge multiplier in effect at node i at time t. When $s^{(t)}_i > 1$, the driver earning from ferrying a passenger from node i to j of the graph is $s^{(t)}_i \times p_{ij}$.
- The arrival process of passengers at node i at time t is a Poisson process with rate $\lambda^{(t)}_i$. Let vector $\Lambda^{(t)}_{n\times 1}$ denote the passenger arrival rates across all the vertices of G at time t. While some passengers may arrive simultaneously (compound Poisson process), we assume that this effect is negligible. Furthermore, we assume that if the driver is present at node i when a

passenger arrives, he picks up the passenger with no further delay. As the inter-arrival times of passengers are exponentially distributed, the expected value of idle time for a driver waiting at node i is $1/\lambda_i^{(t)}$. We implicitly assume that the passenger arrival process is stationary for a finite time slice around t.

- The matrix $M_{n\times n}^{(t)}$ represents the transition probabilites between the nodes of the graph such that the fraction of passengers at node i whose destination is node j at time t is represented by entry $m_{ij}^{(t)}$, where $\forall t, m_{ij}^{(t)} \in \mathbb{R}_{\geq 0}, m_{ii}^{(t)} = 0$ and $\sum_j m_{ij}^{(t)} = 1$.

- H(.) denotes the Heaviside step function.

$$H(x) = \max \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{if } x \geq 0 \end{cases}$$

Table 1: Description of Notation

		Definition
d_{ij}	:	Distance between nodes i and j
c_{ij}	:	Driver cost for trip from node i to node j
p_{ij}	:	Driver earnings from passenger trip from node i to node j
τ_{ij}	:	Travel time from node i to node j
$s_i^{(t)}$	:	Surge multiplier active at node i at time t
$\lambda_i^{(t)}$	:	Poisson passenger arrival rate at node i at time t
$m_{ij}^{(t)}$	:	Fraction of passengers at node i destined to node j at time t
$H(.)$	:	Heaviside function

We formulate a dynamic program to find the contingency plan to maximise the expected earnings of a driver in his service time T. Each entry $OPT(t,i)$ of the matrix OPT represents the maximum expected earnings in the remaining t units of service time when the driver is stationed at node i. The maximum expected revenue of the driver at the beginning of the day is thus $OPT(T, v_0)$.

- Initialization $\forall i, OPT(0,i) = 0$.

- While calculating each $OPT(t,i)$, our algorithm is faced with two possible choices.

  1. Keep waiting at node i for a passenger.
  2. Traverse to some other node j in search of a passenger.

- It chooses whichever choice maximises the expected earnings in the remaining time units t.

$$OPT(t,i) = \max \begin{cases} \sum_{j\neq i} m_{ij}^{(t')} \Big[\Big(s_i^{(t')} p_{ij} - c_{ij} \Big) + OPT(t' - \tau_{ij}, j) \Big], & \text{if } \mathbf{E}[\tau_i | M] \leq t', \\ \max_j \Big[\Big(OPT(t - \tau_{ij}, j) - c_{ij} \Big) \times H(t - \tau_{ij}) \Big], & \text{otherwise} \end{cases}$$

where $t' = t - \frac{1}{\lambda_i^{(t)}}$, denoting the expected time units left after waiting for a passenger at node i.

- While solving the above dynamic program, we maintain an output vector corresponding to a series of driver choices between waiting for a passenger at his current node i or taking an empty ride to some other node j.

3 Uber Driver with flexible work schedule strategy

In this section, we formulate a dynamic program to find optimal work schedule for an Uber driver with a fixed budget of B units of service time. We asssume that the driver uses up his budget before the end of maximum T units of time. For example, a driver who works for a maximum of 40 hours per week shall have $B = 40$ and $T = 168$ hours. Each entry $OPT(t, i, b)$ of the matrix OPT represents the maximum expected earnings with b time units of budget and t time units of week left. The maximum expected revenue of the driver at the beginning of the week is thus $OPT(T, v_0, B)$.

- Initialization $\forall it, OPT(t, i, 0) = 0$.
- While calculating each $OPT(t, i, b)$, our algorithm is faced with distinct possibilities depending on i.

  If $i = v_0$ (driver is at home node),

  1. Stay out of system, waiting for a better time to enter the system.
  2. Enter the system, wait for a passenger at node v_0.

  If $i \neq v_0$ (driver is not at home node),

  1. Stay in the system, wait for a passenger at node i.
  2. Exit the system, drive back to home node v_0.

- The driver chooses whichever choice maximises the expected earnings in remaining b units of his budget.

  if $i = v_0$ and $t > b$,

$$OPT(t, i, b) = \max \begin{cases} OPT(t-1, i, b) \\ \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t' - \tau_{ij}, j, b' - \tau_{ij}) \right] \end{cases}$$

  if $i = v_0$ and $t = b$,

$$OPT(t, i, b) = \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t' - \tau_{ij}, j, b' - \tau_{ij}) \right]$$

if $i \neq v_0$ and $t > b$,

$$OPT(t,i,b) = \max \begin{cases} \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t' - \tau_{ij}, j, b' - \tau_{ij}) \right] \\ \left(OPT(t - \tau_{iv_0}, v_0, b - \tau_{iv_0}) - c_{iv_0} \right) \times H(b - \tau_{iv_0}) \end{cases}$$

if $i = v_0$ and $t = b$,

$$OPT(t,i,b) = \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t' - \tau_{ij}, j, b' - \tau_{ij}) \right]$$

where $b' = b - \frac{1}{\lambda_i^{(t)}}$, denoting the expected budget time units left after waiting for a passenger at node i.

- While solving the above dynamic program, we maintain an output vector corresponding to a series of driver choices between waiting at home node, waiting for a passenger at his current node or taking an empty ride back home.

4 Uber Driver with both re-location and flexible work schedule strategy

Here, we formulate a dynamic program to find optimal work schedule and re-location strategy for an Uber driver with fixed budget of B units of service time. Like the previous section, we assume that the driver uses up his budget before the end of maximum T units of time. We maintain a matrix OPT where each entry, $OPT(t,i,b)$ represents the maximum expected earnings with b time units of budget and t time units of the week left. Similar to previous section, the maximum expected revenue of the driver at the beginning of the week is thus $OPT(T, v_0, B)$.

- Initialization $\forall it, OPT(t,i,0) = 0$.
- While calculating each $OPT(t,i,b)$, our algorithm is faced with distinct possibilities depending on i.

  If $i = v_0$ (driver is at home node),

  1. Stay out of system, waiting for a better time to enter the system.
  2. Enter the system, wait for a passenger at node v_0.
  3. Enter the system, traverse to some other node j in search of a passenger.

  If $i \neq v_0$ (driver is not at home node),

  1. Stay in the system, wait for a passenger at node i.
  2. Stay in the system, traverse to some other node j in search of a passenger.
  3. Exit the system, drive back to home node v_0.

- The driver chooses whichever choice maximises the expected earnings in remaining b units of his budget.

if $i = v_0$ and $t > b$,

$$OPT(t,i,b) = \max \begin{cases} OPT(t-1,i,b) \\ \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t' - \tau_{ij}, j, b' - \tau_{ij}) \right] \\ \max_{j \neq i} \left[\left(OPT(t - \tau_{ij}, j, b - \tau_{ij}) - c_{ij} \right) \times H(b - \tau_{ij}) \right] \end{cases}$$

if $i = v_0$ and $t = b$,

$$OPT(t,i,b) = \max \begin{cases} \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t - \tau_{ij}, j, b' - \tau_{ij}) \right] \\ \max_{j \neq i} \left[\left(OPT(t - \tau_{ij}, j, b - \tau_{ij}) - c_{ij} \right) \times H(b - \tau_{ij}) \right] \end{cases}$$

if $i \neq v_0$ and $t > b$,

$$OPT(t,i,b) = \max \begin{cases} \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t - \tau_{ij}, j, b' - \tau_{ij}) \right] \\ \max_{j \neq i} \left[\left(OPT(t - \tau_{ij}, j, b - \tau_{ij}) - c_{ij} \right) \times H(b - \tau_{ij}) \right] \\ \left(OPT(t - \tau_{iv_0}, v_0, b - \tau_{iv_0}) - c_{iv_0} \right) \times H(b - \tau_{iv_0}) \end{cases}$$

if $i = v_0$ and $t = b$,

$$OPT(t,i,b) = \max \begin{cases} \sum_{j \neq i} m_{ij}^{(t')} \left[\left(s_i^{(t')} p_{ij} - c_{ij} \right) + OPT(t - \tau_{ij}, j, b' - \tau_{ij}) \right] \\ \max_{j \neq i} \left[\left(OPT(t - \tau_{ij}, j, b - \tau_{ij}) - c_{ij} \right) \times H(b - \tau_{ij}) \right] \end{cases}$$

- While solving the above dynamic program, we maintain an output vector corresponding to a series of driver choices between waiting at home node, waiting for a passenger at his current node , taking an empty ride to some other node in search of a passenger or taking an empty ride back home.