Uber driver strategic behavior

Harshal Chaudhari, Evimaria Terzi, John Byers
Department of Computer Science
Boston University

April 20, 2017

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 Problem Formulation

Consider a set of n discrete zones of a city denoted by $\mathcal{X}$. During our data collection we sample rides for each pair of zones to collect information for the route suggested by Uber every 15 minutes. Since the optimal route suggested by Uber to its drivers can change based on the traffic conditions prevalent at the given time of the day, it is essential to model each of the variables for every time step. Hence, there is an implied time superscript for each of the variables described in the table below.

To simplify the notation, let us denote the net reward of a trip from zone i to zone j with $r_{ij} = s_i e_{ij} - c_{ij}$. However, the driver earnings in case of an empty ride (without a passenger) are zero resulting into a negative net reward. Our algorithms described in subsequent sections assume non-negativity of rewards. A slight modification allows us bypass this problem. Let $C = \max_{i,j}\{c_{ij}\}$[1]. Our modified net reward $r'_{ij} = r_{ij} + C$ ensures the non-negativity of the net reward for every ride. Since every reward is offset by equal amount, our modification does not affect the optimal decision at any step in the Markov Decision Process. However, we take this modification into account while calculating overall driver earnings in each of the strategies. Here onwards, we drop the prime notation while refering to this modified net reward.

[1] We take maximum of c_{ij} over all time intervals.

Table 1: Description of Notation

		Definition
$\mathcal{X}_0$	:	Home zone of the Uber driver
N	:	Maximum work time units for Uber driver
B	:	Overall budget time units within which to consume N work time units.
d_{ij}	:	Distance between zones i and j
c_{ij}	:	Driver cost for trip from zone i to zone j
e_{ij}	:	Driver earnings from a passenger trip from zone i to zone j
τ_{ij}	:	Travel time from zone i to zone j
s_i	:	Surge multiplier active at zone i
λ_i	:	Poisson passenger arrival rate at zone i
μ_i	:	Poisson driver arrival rate at zone i
f_{ij}	:	Fraction of passengers at zone i destined to zone j

2.1 Interpreting empirically observed transition frequencies matrix F

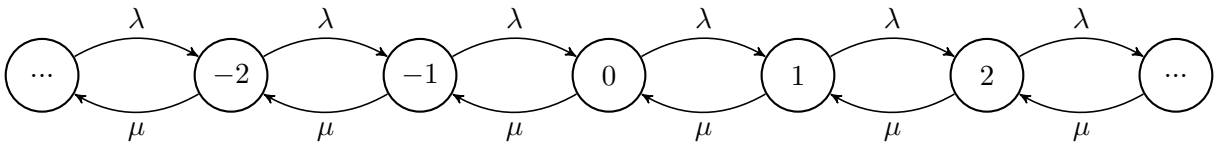
The empirically observed transition matrix is denoted by F, each of its entries being f_{ij} as described in the table above. By adjusting the size of city zones such that the diameter of the city zone is maximum walking distance, we can ensure that there are no within-zone passenger rides i.e., $f_{ii} = 0$. We modify the matrix F, denoting it by F', such that the entry f'_{ii} indicates the probability that an Uber driver unsuccessfully 'busy-waits' for a passenger within zone i, i.e., driver does not find a passenger in a unit time interval. Furthermore, for every row of the matrix we re-normalize the entries $f'_{ij}, i \neq j$ such that they denote conditional probabilities of a driver from node i taking a ride to node j, given that he finds a passenger.

2.2 Modeling probability of successful busy-wait

For brevity, again, we drop the time superscript in this section. Let λ and μ denote the poisson arrival rates of passengers and drivers within a given zone. Let the number of passenger and driver arrivals in a unit time interval be denoted by $N(\lambda)$ and $N(\mu)$ respectively. Assuming that passenger and driver arrivals follow independent Poisson processes, the random variable $K = N(\lambda) - N(\mu)$ follows a Skellam distribution.

$$\Pr[K = k] \quad = \quad e^{-(\lambda+\mu)} \left(\frac{\lambda}{\mu}\right) I_k\left(2\sqrt{\lambda\mu}\right) \tag{1}$$

where $I_k(z)$ is the modified Bessel function of the first kind. We can depict the values taken by the random variable K as states of a Markov Chain. Using this, we can model the entry f'_{ii} of the modified transition frequency matrix. [**Harshal's Note: Although for simplicity, we assume the independence of the passenger and driver arrival processes, we can accomodate correlated processes as well with slight modification.**]



Whenever the above Markov Chain is in non-positive state, there are more drivers than passengers in a zone. We assume the worst case scenario in which our strategic driver is at the last position in a FIFO queue. Hence, for a state $k \leq 0$ of the Markov Chain, the driver has to wait for $(|k|+1)$ passenger arrivals in a unit time interval for a successful busy-wait resulting in a passenger ride in the same time interval. Since the passenger arrival process is a Poisson process with rate λ, the probability of k passengers arriving in a unit time interval is given by,

$$\Pr[N(\lambda) = k] \quad = \quad \frac{\lambda^k e^{-\lambda}}{k!}$$

Hence, probability of a driver finding a passenger in a unit time interval within zone i is given by,

$$\begin{aligned} 1 - f'_{ii} &= \sum_{k \leq 0} \Pr[K = k] \times \Pr\big[N(\lambda) = |k| + 1\big] \\ f'_{ii} &= 1 - \sum_{k \leq 0} \Pr[K = k] \times \Pr\big[N(\lambda) = |k| + 1\big] \quad (2) \\ f'_{ij} &= (1 - f'_{ii}) \times f_{ij} \quad (3) \end{aligned}$$

The above algebraic transformation maintains the row stochasticity of the matrix F'. Furthermore, $\forall i, j : f'_{ij} > 0$ (assuming that there is at least one passenger ride between each pair of zones in any time interval). Here onwards, for simplicity of notations, we always refer to this modified empirically observed transition matrix as F.

3 Naive Strategy

In the naive strategy, an Uber driver essentially performs a random walk over the city zones. At the end of every ride, the driver waits in the same zone for his next passenger pickup. We refer to the action of waiting for a passenger pickup in current zone as 'busy-waiting' and denote it by a_{bw}. Hence, in the naive strategy, the set of available actions for a driver at any stage is a singleton set, $\mathcal{A} = \{a_{bw}\}$. Let $R(i, a)$ denote the expected immediate reward of taking action a while within zone i. In the naive strategy, for the only action a_{bw}, we have $R(i, a_{bw}) = \vec{f_i}.\vec{r_i}$ where $\vec{f_i} = \{f_{i1}, ..., f_{in}\}$ is the row vector of the transition matrix F, and $\vec{r_i} = \{r_{i1}, ..., r_{in}\}$ is the net reward vector associated with each transition, such that $r_{ii} = 0$.

In the dynamic program modeling of the naive strategy, the expected overall earnings i.e., value function at time t, for a zone i, $v_t(i)$ is given by,

$$v_t(i) \quad = \quad R(i, a_{bw}) + \vec{f_i}.\vec{v}_{t'}$$

where $\vec{v}_{t'} = \{v_{t+\tau_{i1}}(1), ..., v_{t+\tau_{in}}(n)\}$ is the value function value at time $t + \tau_{ij}$ corresponding to each transition from zone i to zone j. Note that $v_{t+\tau_{ii}}(i) = v_{t+1}(i)$ as it simply implies unsuccessful busy-wait.

4 Relocation Strategy

In the relation strategy, an idle Uber driver in zone i can perform one of the two actions:

1. Busy wait for a passenger ride request in the current zone, denoted by a_{bw}.
2. Take an empty ride to a different zone j in the city, denoted by a_j, where $j \neq i$.

Hence, in the relocation strategy, the set of available actions for a driver at any stage, $\mathcal{A}$ is a set of n elements. Like the previous strategy, let $R(i, a)$ denote the expected immediate reward of taking action a while within the zone i. Hence, we have,

1. $R(i, a_{bw}) = \vec{f_i}.\vec{r_i}$ where $\vec{f_i}$ and $\vec{r_i}$ are as defined for the naive strategy.
2. $R(i, a_j)_{j \neq i} = r_{ij}$

In the dynamic program modeling of the relocation strategy, the value function at time t, for a zone i, is given by,

$$v_t(i) = \max \begin{cases} R(i, a_{bw}) + \vec{f_i}.\vec{v}_{t'}, \\ \max_{a_j} \left\{ R(i, a_j) + v_{t+\tau_{ij}}(j) \right\}, j \neq i, t + \tau_{ij} \leq N \end{cases}$$

where $\vec{v}_{t'}$ is same as defined for the naive strategy.

5 Flexible Schedule Strategy

In the flexible schedule strategy, every Uber driver has an allocated overall budget of B time units within which he works for a maximum of N time units. The motivation of this strategy is as follows. A typical Uber driver desires to work for a maximum of 40 hours per week. However, it is a non-trivial problem to determine which are the optimal 40 hours due to varying demand and surge rates all through the week. The flexible schedule strategy aims to figure out the optimal in expectation work schedule for 40 hours in a week of a total 168 hours. In this strategy, we assume that a driver can log out of the system any time to return to his home zone as a part of his flexible schedule. Hence depending on the current zone of the driver, he has following set of actions available to him.

- **If driver is in home zone**
  1. Idle wait in the home zone, a_{iw}. Idle waiting involves logging out of the system till a more optimal in expectation time to re-enter the system occurs.
  2. Log into the system and busy wait for a passenger ride request in the home zone, denoted by a_{bw}.
- **If driver is not in home zone**
  1. Busy wait for a passenger ride request in the current zone, a_{bw}.
  2. Log out of the system, drive back to home zone to idle wait over there, a_{iw}.

However, we can merge the above to situations into a single set of actions by accounting for the extra cost of an empty ride to home zone that is imposed on a driver not in the home zone, while calculating the net rewards vector. Hence, in the flexible schedule strategy, the set of available actions for a driver at any stage is $\mathcal{A} = \{a_{iw}, a_{bw}\}$. The expected immediate rewards of taking a particular action while within zone i are as follows:

1. $R(i, a_{iw}) = r_{ij}$, where $r_{ii} = 0$.
2. $R(i, a_{bw}) = \vec{f_i}.\vec{r_i}$ where $\vec{f_i}$ and $\vec{r_i}$ are as defined for the naive strategy.

Now, in the dynamic program modeling of the flexible schedule strategy, the value function is defined at time t and budget b, for a zone i. It is given by,

$$v_{t,b}(i) \quad = \quad \max \begin{cases} R(i, a_{bw}) + \vec{f_i}.\vec{v}_{t',b'} \\ \\ R(i, a_{iw}) + v_{t,b'}(\mathcal{X}_0) \end{cases}$$

where in case of busy-waiting action, $\vec{v}_{t',b'} = \{v_{t+\tau_{i1},b+\tau_{i1}}(1), ..., v_{t+\tau_{in},b+\tau_{in}}(n)\}$ is the value function value at time $t + \tau_{ij}$ and budget $b + \tau_{ij}$, corresponding to each transition from zone i to zone j. Note that, $v_{t+\tau_{ii},b+\tau_{ii}}(i) = v_{t+1,b+1}(i)$ as it simply implies unsuccessful busy-wait. In case of idle waiting action, $v_{t,b+\tau_{i\mathcal{X}_0}}(\mathcal{X}_0) = v_{t,b+1}(\mathcal{X}_0)$ if $i = \mathcal{X}_0$. This takes into account that while idle-waiting, only the budget time units are consumed.

6 Relocation + Flexible Schedule Strategy

This is the most general strategy formed by combining the relocation as well as flexible schedule strategies described in the previous sections. In this strategy, a driver has complete freedom for decisions regarding work schedule as well one regarding taking empty rides to different zones of the city in order to maximise total expected reward. Just like the flexible schedule strategy, a driver is allocated overall budget of B time units within which he works for a maximum of N time units. An idle driver in zone i has following set of actions available to him at any stage in the strategy.

1. Busy wait for a passenger ride request in the current zone, a_{bw}.
2. Idle wait in the home zone, a_{iw}.
3. Empty ride to a different zone j in the city, denoted by a_j, where $j \neq i$.

Hence, in this strategy, the set of available actions for an idle driver at any stage $\mathcal{A}$ is a set of $n+1$ elements. The expected immediate rewards of taking particular action while within zone i are as follows:

1. $R(i, a_{bw}) = \vec{f_i}.\vec{r_i}$.
2. $R(i, a_{iw}) = r_{ij}$ where $r_{ii} = 0$.
3. $R(i, a_j)_{j \neq i} = r_{ij}$.

In the dynamic program modeling of this strategy, the value function is again defined at time t and budget b, for a zone i. It is given by,

$$v_{t,b}(i) \quad = \quad \max \begin{cases} R(i, a_{bw}) + \vec{f}_i.\vec{v}_{t',b'} \\ \\ R(i, a_{iw}) + v_{t,b+\tau_i x_0}(\mathcal{X}_0) \\ \\ \max_{a_j} \Big\{ R(i, a_j) + v_{t+\tau_{ij}, b+\tau_{ij}}(j) \Big\}, j \neq i, t + \tau_{ij} \leq N, b + \tau_{ij} \leq B \end{cases}$$

where all notations are as defined in the previous sections.

7 Problem Setup

Consider a finite horizon Markov decision process with finite decision horizon $T = \{0, 1, 2, ..., N-1\}$. At each stage, the driver occupies a state $i \in \mathcal{X}$, where $n = |\mathcal{X}|$ is finite, and a driver is allowed to choose an action a from finite set of allowable actions $\mathcal{A} = \{a_1, a_2, ..., a_m\}$. The system starts in a given initial state i_0. The states make Markov transitions according to a collection of time-dependent transition matrices $\tau := (P_t^a)_{a \in \mathcal{A}, t \in \mathcal{T}}$, the $n \times n$ transition matrix P_t^a contains the probabilities of transitioning under action a at stage t. Let $\pi = (\mathbf{a}_0, ..., \mathbf{a}_{N-1})$ denote a policy, where $\mathbf{a}_t(i)$ denotes the action of the driver while at state i at time t. Define by $r_t(i, a)$, the reward corresponding to state i and action a at time t, and by r_N the reward at the final stage (end of time). We assume that $r_t(i, a)$ is non-negative and finite (negative cost of take empty ride action can be accounted for by considering all the rewards relative to the highest negative cost of empty ride i.e., moving the zero to the left.)

For a given set of transition matrices τ, if $R_N(\pi, \tau)$ denotes the *expected total rewards* under the policy π and transition matrices τ, then,

$$R_N(\pi, \tau) \quad := \quad \mathbf{E}\Big(\sum_{t=0}^{N-1} r_t(i_t, \mathbf{a}_t(i)) + r_N(i_N) \Big)$$

When the set of transition matrices τ is known, $R_N(\pi, \tau)$ maximising policy π can be found via dynamic programming algorithm, with a complexity of $O(mnN)$. The *nominal* problem giving the corresponding maximum expected total reward is,

$$\phi_N(\Pi, \tau) \quad := \quad \max_{\pi \in \Pi} R_N(\pi, \tau),$$

where $\Pi = \mathcal{A}^{nN}$ is the entire policy space.

In our case, the transition matrix under each action a and time t is known to lie in some given subset $\mathcal{P}_t^a$, where $\mathcal{P}_t^a$ can be thought of as *sets of confidence* for the transition matrices. Given this uncertainty in the transition matrices, we are interested in maximising the lower bound on the expected total rewards for the driver. This leads to a game between the driver and *nature*, where the driver seeks to maximise the minimum expected reward, with nature being the minimizing player. Formally, *policy of nature* refers to specific collection of time-dependent transition matrices $\tau = (P_t^a)_{a \in \mathcal{A}, t \in \mathcal{T}}$ chosen by nature, and the set of admissible policies of nature is $\mathcal{T} := (\otimes_{a \in \mathcal{A}} \mathcal{P}_t^a)^N$. The maximum lower bound can be given by,

$$\phi_N(\Pi, \mathcal{T}) \quad := \quad \max_{\pi \in \Pi} \min_{\tau \in \mathcal{T}} R_N(\pi, \tau) \tag{4}$$

8 Robust Finite Horizon Dynamic programming

For a given state $i \in \mathcal{X}$, action $a \in \mathcal{A}$ and $P_t^a \in \mathcal{P}_t^a$, we denote by $p_{i,t}^a$ the next-state distribution drawn from P_t^a corresponding to state $i \in \mathcal{X}$, the i-th row of the matrix. Furthermore, we define $\mathcal{P}_{i,t}^a$ as the projection of the set $\mathcal{P}_t^a$ onto the set of $p_{i,t}^a$ variables. Using the robust dynamic programming theorem from **?**, we can show that perfect duality holds, i.e.,

$$\phi_N(\Pi, \mathcal{T}) = \max_{\pi \in \Pi} \min_{\tau \in \mathcal{T}} R_N(\pi, \tau) = \min_{\tau \in \mathcal{T}} \max_{\pi \in \Pi} R_N(\pi, \tau) = \psi_N(\Pi, \mathcal{T}) \tag{5}$$

The problem can be solved via recursion. The worst case optimal value function in state i at time t, $v_t(i)$ is given by,

$$v_t(i) \quad = \quad \max_{a \in \mathcal{A}} \left(r_t(i,a) + \sigma_{\mathcal{P}_{i,t}^a}(v_{t+1}) \right), i \in \mathcal{X}, t \in T \tag{6}$$

where, $\sigma_{\mathcal{P}(v)} = \inf\{p^T v : p \in \mathcal{P}\}$. It should be noted in that solving $\sigma_{\mathcal{P}(v)} = \inf\{p^T v : p \in \mathcal{P}\}$ is the inner problem in every step of the above recursion. A corresponding optimal driver policy $\pi^* = (\mathbf{a}_0^*, ..., \mathbf{a}_{N-1}^*)$ is obtained by,

$$\mathbf{a}_t^*(i) \quad = \quad \arg\max_{a \in \mathcal{A}} \left\{ r_t(i,a) + \sigma_{\mathcal{P}_{i,t}^a}(v_{t+1}) \right\}, i \in \mathcal{X} \tag{7}$$

Furthermore, the effect of uncertainty on some other policy $\pi = (\mathbf{a}_0, ..., \mathbf{a}_{N-1})$ can be evaluated as follows,

$$v_t^\pi(i) \quad = \quad r_t(i, \mathbf{a}_t(i)) + \sigma_{\mathcal{P}_{i,t}^{\mathbf{a}_t(i)}}(v_{t+1}^\pi), i \in \mathcal{X} \tag{8}$$

9 Likelihood Model

We use a likelihood constraint to describe the uncertainty in the transition matrix for the action a (when driver takes a passenger ride). Let F_t^a denote the matrix of observed frequencies of transitions in our data (in time slice centered around time t), and let $f_{i,t}^a$ be its i-th row. We have $F_t^a \geq 0$ and $F_t^a \mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ denotes the vector of ones. For simplicity, let us assume that $F_t^a > 0$, which is same as assuming $f_{i,t}^a > 0$, $f_{i,t}^a \in \mathbf{R}_+^n$ and $(f_{i,t}^a)^T \mathbf{1} = 1$. To simplify the notation in this section, we shall drop the subscript t denoting the time slice as we estimate the likelihood models independently for each time slice. Furthermore, we shall also drop the superscript a denoting action, because there is uncertainty associated with only one of the actions in our model i.e, driver taking a passenger ride. [**Harshal's Note: The other category of actions is 'taking empty ride to some other state'. There is no uncertainty involved in the transition matrix coresponding to this action, as exactly one value in every row is exactly 1. Another type of action in flexible work schedule involves driver going offline in the system. Again, there is either no transition in this action, or taking a definite transition to home node of the driver.**]Plug-in estimates for the transition matrix P with uncertainty level β is given by,

$$\left\{ P \in \mathbf{R}^{n \times n} : P \geq 0, P\mathbf{1} = \mathbf{1}, \sum_{i,j} F(i,j) \log P(i,j) \geq \beta \right\}$$

where $\beta < \beta_{max} = \sum_{i,j} F(i,j) \log F(i,j)$. From **?**, for a $(1 - U_L)$ level of confidence, β is solution of the equation,

$$(1 - U_L) \quad = \quad F_{\mathcal{X}^2_{n(n-1)}}(2(\beta_{max} - \beta))$$

where $F_{\mathcal{X}^2_d}$ is the cumulative density function of the $\mathcal{X}^2$-distribution with d degrees of freedom. In the inner problem, we only need to work with uncertainty on each row p_i. Due to separable nature of the log-likelihood function, the projection of the above set onto the p_i variables of the matrix P can be given as,

$$\mathcal{P}_i(\beta_i) = \left\{ p \in \Delta^n : \sum_j f_i(j) \log p_i(j) \geq \beta_i \right\}$$

where,

$$\begin{aligned} \beta_i &= \beta - \sum_{k \neq i} \sum_j F(k,j) \log F(k,j). \\ \Delta^n &= \{ p \in \mathbf{R}^n_+ : p^T \mathbf{1} = 1 \} \end{aligned}$$

In the following section, as we deal with each row of the matrix separately, we drop the subscript i in the observed frequencies f_i and in the lower bound β_i. Hence $\beta_{max} = \sum_j f(j) \log f(j)$. We assume that $\beta < \beta_{max}$ along with $f > 0$. [**Harshal's Note: Unsure what happens if we relax this constraint to $f \geq 0$, which would be more appropriate in our cases, as during some time slices there no transitions to/from airport zone of the city.**] Without loss of generality, we can also assume that $v \in \mathbf{R}^n_+$.

10 Bisection Algorithm

The inner problem of the robust dynamic program is as follows:

$$\sigma^* := \min_p p^T v : p \in \Delta^n, \sum_j f(j) \log p(j) \geq \beta$$

The Lagrangian $\mathbf{L} : \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ associated with the inner optimization problem is

$$\mathbf{L}(v, \zeta, \mu, \lambda) = p^T v - \zeta^T p + \mu(1 - p^T \mathbf{1}) + \lambda(\beta - f^T \log p)$$

where ζ, μ, and λ are Lagrange multipliers. The dual function $d : \mathbf{R}^n \times \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ is the minimum value of the Lagrangian over p for $\zeta \in \mathbf{R}^n, \mu \in \mathbf{R}$, and $\lambda \in \mathbf{R}$,

$$\begin{aligned} d(\zeta, \mu, \lambda) &= \inf_p \mathbf{L}(v, \zeta, \mu, \lambda) \\ &= \inf_p (p^T v - \zeta^T p + \mu(1 - p^T \mathbf{1}) + \lambda(\beta - f^T \log p)) \end{aligned}$$

The optimal p^* minimizing the above dual function is obtained by solving $\frac{\partial \mathbf{L}}{\partial p} = 0$, which gives us,

$$p^*(i) = \frac{\lambda f(i)}{v(i) - \zeta(i) - \mu}$$

Plugging in the value of $p^*(i)$ into the dual function d, after some simplification, gives us the dual problem,

$$\bar{\sigma} := \max_{\zeta, \mu, \lambda} \lambda(1 + \beta) + \mu - \lambda \sum_j f(j) \log \left(\frac{\lambda f(j)}{v(j) - \zeta(j) - \mu} \right) : \lambda \geq 0, \zeta \geq 0, v \geq \zeta + \mu \mathbf{1}$$

As the above problem is concave, has a feasible set with non-empty interior (checked by plotting), there is no duality gap, that is $\sigma^* = \bar{\sigma}$. [**Harshal's Note: Previous statement needs clarification. The problem is upper convex, i.e., concave, not convex. Not sure how to argue the no duality gap**]. Moreover, by a monotonicity argument (ζ increasing causes the function value to decrease monotonically), we can conclude that ζ is zero. Thus, the last constraint of the dual problem can be expressed as $\mu \leq v_{min} := \min_j v(j)$. Hence, we get,

$$\begin{aligned} \sigma^* &= \max_{\lambda,\mu} h(\lambda,\mu) \\ \text{where,} & \\ h(\lambda,\mu) &:= \begin{cases} \lambda(1+\beta) + \mu - \lambda \sum_j f(j) \log\left(\frac{\lambda f(j)}{v(j)-\mu}\right), & \text{if } \lambda > 0, \mu < v_{min}, \\ -\infty, & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

The gradient of h is given by,

$$\frac{\partial h(\lambda,\mu)}{\partial \lambda} = \beta - \sum_j f(j) \log\left(\frac{\lambda f(j)}{v(j)-\mu}\right) \tag{10}$$

$$\frac{\partial h(\lambda,\mu)}{\partial \mu} = 1 - \lambda \sum_j \left(\frac{f(j)}{v(j)-\mu}\right) \tag{11}$$

From (3), we obtain the optimal value of λ for a fixed value of μ, $\lambda(\mu)$ given by,

$$\lambda(\mu) = \left(\sum_j \frac{f(j)}{v(j)-\mu}\right)^{-1}$$

which further reduces the problem to a 1-dimensional optimization problem,

$$\begin{aligned} \sigma^* &= \max_{\mu < v_{min}} \sigma(\mu) \\ \text{where,} & \\ \sigma(\mu) &= h(\lambda(\mu),\mu) \end{aligned}$$

$\sigma(\mu)$ is a concave function. Now, we may use a bisection algorithm to maximise this function.

To initialize the bisection algorithm, we need upper and lower bounds μ_+ and μ_- on a minimizer of σ. When $\mu \to v_{min}, \sigma(\mu) \to v_{min}$ and $\sigma'(\mu) \to -\infty$ [**Harshal's Note: To be proved analytically, verified by graphing**]. Thus, we may set upper bound $\mu_+ = v_{min}$. The lower bound μ_- must be chosen such that $\sigma'(\mu_-) > 0$.

$$\sigma'(\mu) = \frac{\partial h}{\partial \mu}(\lambda(\mu),\mu) + \frac{\partial h}{\partial \lambda}(\lambda(\mu),\mu)\frac{\partial \lambda(\mu)}{\partial \mu}$$

By construction of $\lambda(\mu)$, the first term of the above derivative is zero. Furthermore, $\frac{\partial \lambda(\mu)}{\partial \mu} < 0$. Hence, we need μ such that $\frac{\partial h}{\partial \lambda}(\lambda(\mu),\mu) < 0$. Using the bounds on $\lambda(\mu)$, $v_{min} - \mu \leq \lambda(\mu) \leq v_{max} - \mu$, we can show that,

$$\begin{aligned} \frac{\partial h}{\partial \lambda}(\lambda(\mu),\mu) &= \beta - \sum_j f(j) \log\left(\frac{\lambda(\mu) f(j)}{v(j)-\mu}\right) \\ &= \beta - \sum_j f(j) \log f(j) - \sum_j f(j) \log\left(\frac{\lambda(\mu)}{v(j)-\mu}\right) \end{aligned}$$

$$
\begin{aligned}
&= \beta - \beta_{max} - \sum_j f(j) \log\left(\frac{\lambda(\mu)}{v(j)-\mu}\right) \text{(Need to verify this step)} \\
&\leq \beta - \beta_{max} - \log\left(\sum_j \frac{f(j)\lambda(\mu)}{v(j)-\mu}\right) \\
&\leq \beta - \beta_{max} - \log\left(\sum_j \frac{\lambda(\mu)}{v(j)-\mu}\right) \\
&\leq \beta - \beta_{max} - \log\left(\sum_j \frac{v_{min}-\mu}{v(j)-\mu}\right) \\
&\leq \beta - \beta_{max} - \log\left(\sum_j \frac{v_{min}-\mu}{v_{max}-\mu}\right) \\
&\leq \beta - \beta_{max} - \log\left(n \times \frac{v_{min}-\mu}{v_{max}-\mu}\right) \\
&< \beta - \beta_{max} - \log\left(\frac{v_{min}-\mu}{v_{max}-\mu}\right) (\text{n} > 1)
\end{aligned}
$$

Therefore, the sufficient condition is,

$$
\begin{aligned}
0 &> \beta - \beta_{max} - \log\left(\frac{v_{min}-\mu}{v_{max}-\mu}\right) \\
\log\left(\frac{v_{min}-\mu}{v_{max}-\mu}\right) &> \beta - \beta_{max} \\
\left(\frac{v_{min}-\mu}{v_{max}-\mu}\right) &> e^{\beta-\beta_{max}} \\
(v_{min}-\mu) &> e^{\beta-\beta_{max}}(v_{max}-\mu) \\
v_{min} - e^{\beta-\beta_{max}} v_{max} &> \mu(1-e^{\beta-\beta_{max}})
\end{aligned}
$$

Hence,

$$
\mu_-^0 := \frac{v_{min} - e^{\beta-\beta_{max}} v_{max}}{1-e^{\beta-\beta_{max}}} > \mu
$$

By construction, the interval $[\mu_-^0, v_{min})$ is guaranteed to contain the global maximiser of σ over $(-\infty, v_{min})$. The bisection algorithm is as follows:

1. Set $\mu_+ = v_{min}$ and $\mu_- = \mu_-^0$. Let $k = 1, \mu_1 = (\mu_+ + \mu_-)/2$.
2. While $k \leq N$:
   (a) If $\sigma^{'}(\mu_1) > 0$, set $\mu_- = \mu_1$, if $\sigma^{'}(\mu_1) < 0$ then $\mu_+ = \mu_1$, else break
   (b) $k = k + 1$,
   (c) $\mu_k = (\mu_+ + \mu_-)/2$
3. $\hat{\mu}^* = \arg\max_i\{\sigma(\mu_i)\}$

Lemma: After $N \approx \log_2(V/\delta)$ where $V = \max\big(\sigma^* - \sigma(\mu_+), \sigma^* - \sigma(\mu_-)\big)$, the bisection algorithm provides optimal solution to the inner problem within an accuracy $\delta > 0$ i.e., $\sigma^* - \sigma(\hat{\mu}^*) \leq \delta$ which we call as the δ-solution.

Proof: Let the interval $[\mu_-, \mu_+]$ from step 1 of the bisection algorithm be denoted by G. At each iteration, the length of the interval that contains the global maximiser of σ is exactly halved. Hence, let $[\mu_{N-}, \mu_{N+}]$ be the corresponding interval after N iterations of the bisection algorithm. Length of the interval G_N is 2^{-N} times the length of G. Using the bisection algorithm, we know that,

$$\forall \mu : \mu \in G \setminus G_N \rightarrow \sigma(\mu) \leq \sigma(\hat{\mu}^*)$$

Let $2^{-N} < \alpha < 1$. α-contraction of G to μ^* is the segment given by points,

$$G^\alpha = (1-\alpha)\mu^* + \alpha G = \big\{(1-\alpha)\mu^* + \alpha z \big| z \in G\big\}$$

Since, $\alpha > 2^{-N}$, we know that length of $G^\alpha > G_N$, in fact, length of G^α is $\alpha(\mu_+ - \mu_-)$. Hence, $\exists y \in G^\alpha$ such that $y \notin G_N$. Furthermore, $\exists z \in G$ such that $y = (1-\alpha)\mu^* + \alpha z$. By the concavity of $\sigma(\mu)$ function,

$$\begin{aligned} \sigma(y) &\geq (1-\alpha)\sigma(\mu^*) + \alpha\sigma(z) \\ \sigma(\mu^*) - \sigma(y) &\leq \alpha(\sigma(z) - \sigma(\mu^*)) \\ &\leq \alpha V \end{aligned}$$

Hence, y is an αV-solution to our problem.

$$\begin{aligned} \sigma(\hat{\mu}^*) &\geq \sigma(y) \\ \sigma(\mu^*) - \sigma(\hat{\mu}^*) &\leq \sigma(\mu^*) - \sigma(y) \leq \alpha V \end{aligned}$$

Hence, sufficient condition for obtaining a δ-solution is, $\alpha V \leq \delta$ i.e., $N \geq \log_2(V/\delta)$. However, V is unknown by itself. But, the objective function of the inner problem, $p^T v$, is bounded from above by v_{max}. Therefore, the lemma still holds for $V = \max\big(v_{max} - \sigma(\mu_+), v_{max} - \sigma(\mu_-)\big)$.

11 ϵ-suboptimal Algorithm

Each step of the robust dynamic programming algorithm from Section 8 involves finding the solution of an optimization problem, referred to as the 'inner problem', of the form,

$$\sigma_{\mathcal{P}}(v) := \min_{p \in \mathcal{P}} p^T v,$$

where p corresponds to particular row of a transition matrix, $\mathcal{P} = \mathcal{P}^a_{i,t}$ is the set that describes the uncertainty of this row, and v contains the elements of the value function at this step. Section 9 provides a criterion for the choice of uncertainty model that represents accurately the statistical uncertainty on the transition matrices. Section 10 describes a bisection algorithm that provides an optimal solution to the inner problem within an accuracy $\delta > 0$. [**Harshal's Note: This should be ensured by the stopping criterion of the algorithm**]. Thus, for a given $v \in \mathbf{R}^n_+$ and $\delta > 0$, the bisection algorithm gives output of the form,

$$\hat{\sigma}_{\mathcal{P}}(v) = \sigma_{\mathcal{P}}(v) - \delta_{\mathcal{P}}(v) \text{ where } 0 \leq \delta_{\mathcal{P}}(v) \leq \delta$$

An ϵ-suboptimal policy, $\hat{\pi}$ is a policy such that the worst-case expected total reward under policy $\hat{\pi}$, namely, $\phi_N(\hat{\pi}, \mathcal{T}) = \min_{\tau \in \mathcal{T}} R_N(\hat{\pi}, \tau)$, satisfies

$$\phi_N(\hat{\pi}, \mathcal{T}) \leq \phi_N(\Pi, \mathcal{T}) \leq \phi_N(\hat{\pi}, \mathcal{T}) + \epsilon$$

Here, $\epsilon > 0$ is given. Using the uncertainty model from Section 9, we solve the bisection algorithm with an accuracy $\delta := \epsilon/N$. This gives us the robust finite horizon dynamic programming algorithm as follows,

1. Set $\epsilon > 0$. Initialize the value function to its terminal value $\hat{v}_N = r_N$
2. Repeat until t=0:
   (a) For every state $i \in \mathcal{X}$ and action $a \in \mathcal{A}$, compute, using the bisection algorithm described in the previous section, a value $\hat{\sigma}_i^a$ such that,

$$\hat{\sigma}_i^a \leq \sigma_{\mathcal{P}_{i,t}^a}(v_t) \leq \hat{\sigma}_i^a + \frac{\epsilon}{N}$$

   (b) Update the value function by,

$$v_{t-1}(i) \quad = \quad \max_{a \in \mathcal{A}} \left(r_{t-1}(i,a) + \hat{\sigma}_i^a \right), i \in \mathcal{X}$$

   (c) t = t - 1
3. For every $i \in \mathcal{X}$ and $t \in T$, set $\pi^\epsilon = (\mathbf{a}_0^\epsilon, ..., \mathbf{a}_{N-1}^\epsilon)$, where

$$\mathbf{a}_t^\epsilon \quad = \quad \arg\min_{a \in \mathcal{A}} \left\{ r_{t-1}(i,a) + \hat{\sigma}_i^a \right\}, i \in \mathcal{X}, a \in \mathcal{A}$$