Nomadic Trucker ADP

Arash Dehghan

January 5, 2022

1 Nomadic Trucker Problem

1.1 Problem Description

In this explicit variation of the *Nomadic Trucker Problem*, there exists a single trucker whose goal it is to optimize the long-term rewards gained from transporting randomly-appearing loads from location to location in a 16×16 grid. There exists both a multi-attribute and single-attribute version of this problem, though we will be discussing the single-attribute variation.

In the single-attribute case, the trucker is defined solely by his current location $l \in \mathcal{L}$. At the beginning of each day, the trucker observes a new set of loads (demands) which probabilistically appear at each of the given set of locations. The trucker then makes his decision as to whether he wants to visit a given location j from his current location i. If there is no load demand at location j (i.e: there is no demand to move a load from location i to location j, then the move is classified as an 'empty' move, otherwise, it is defined as a loaded move. Once a decision is made, all other loads which are not selected are lost, and the trucker then must repeat this process with a new set of loads from his new location j on the next day.

As previously mentioned, the problem setting takes place on a square 1000×1000 mile 16×16 Euclidean grid in which each location $i \in \mathcal{L}$ is described via its (x_i, y_i)-coordinates with the minimum distance between two adjacent locations being $1000/15$ miles. The probability that a load from location i to location j will appear on any given day is $p_{ij} = b_i(1 - b_j)$ where $0 \leq b_i \leq 1$ represents the probability of a load originating at location i will appear for all $i \in \mathcal{L}$. The origin probabilities are given by:

$$b_i = \rho(1 - \frac{f(x_i, y_i) - f^{min}}{f^{max} - f^{min}})$$

in which $\rho = 1$ and represents the arrival intensity of loads and $f(x_i, y_i)$ is given by the Six-hump camel back function $f(x_i, y_i) = 4{x_i}^2 - 2.1{x_i}^4 + \frac{1}{3}{x_i}^6 + x_i y_i - 4{y_i}^2 + 4{y_i}^4$ on the domain $(x_i, y_i) \in [-1.5, 2] \times [-1, 1]$. The function is

smoothened and scaled to the domain $(x_i, y_i) \in [0, 1000] \times [0, 1000]$. The results are shown in Figure 1

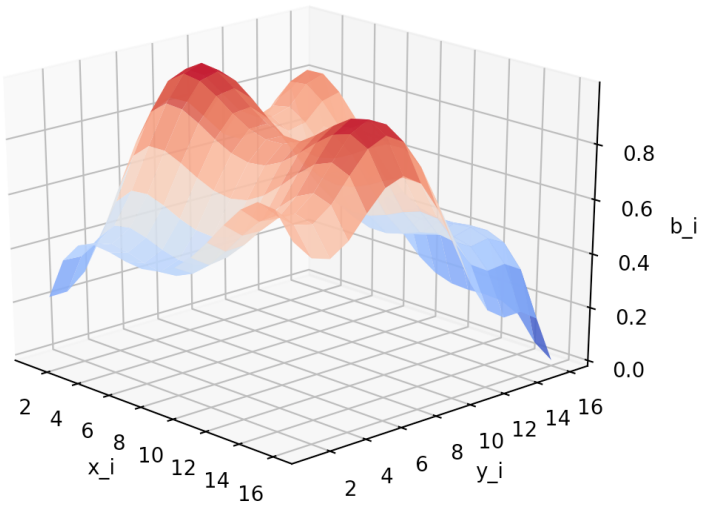

Figure 1: Origin Probabilities for the 256 Locations

1.2 MDP Model

We will consider only the single attribute, infinite horizon case of the problem.

1.2.1 States

A state is defined by $S_t = (l_t, D_t)$ in which l_t defines the location of the trucker at time t and D_t is a vector which represents the load availability from the truckers current location l_t to location i in which $D_{t,i} = 1$ if a load is available and 0 otherwise. Thus, the size of the state space becomes $(256) \cdot 2^{256}$ where the trucker can be located at any of the 256 locations on the grid, and each grid location can have either a value of 0 or 1 in the demand vector.

1.2.2 Actions

The action a_t takes the trucker from its current location l_t to location j. If a load is available from l_t to j then the move will automatically be a loaded move with corresponding rewards/costs, whereas if no load is available, it will be an empty move. Thus, since the trucker may move to any given location at time t (or stay at his current location), the size of the decision space $\mathcal{A}$ is 256.

1.2.3 Costs

The costs associated with action a_t are represented by:

$$C(S_t, a_t) = \begin{cases} -d(l_t, a_t) & if\ D_{t,a_t} = 0 \\ d(l_t, a_t) \cdot b_i & if\ D_{t,a_t} = 1 \end{cases}$$

in which $d(l_t, a_t)$ is the euclidean distance between the truckers current location and the location action a_t will take it to and b_i represents the probability of a load originating at location i.

1.2.4 ADP Solution

We use a forward dynamic programming approach in which we look to approximate the down-stream costs associated with each post-decision state. In particular, post-decision states remove the stochastic information from our state description leaving us with a single deterministic post-decision state value per location of trucker in this problem. In this way, our value function approximations of our post decision state $\bar{V}^n$ in the single-attribute infinite horizon case are represented by a 1×256 vector which is initialized to all values of 0.

In each step, after a new set of demands are introduces, we solve the ADP forward optimality equations shown below using the post-decision state and the approximated next-stage cost:

$$\hat{v}^n = \min_{a^n \in \mathcal{A}} \ (C(S^n, a^n) + \gamma \bar{V}^{n-1}(S^{M,a}(S^n, a^n)))$$

to determine the decision which minimizes $\hat{v}^n$:

$$\tilde{a}^n = \arg\min_{a^n \in \mathcal{A}} \ (C(S^n, a^n) + \gamma \bar{V}^{n-1}(S^{M,a}(S^n, a^n)))$$

where $S^{M,a}(S^n, a^n)$ represents the transition function of our states. Note that we take the value function approximations of our next post-decision state using the previous iterations values in our matrix (consider this our downstream costs) and multiply this with a discount factor, then add our immediate costs.

Given a exploitation-exploration policy, we then select our action to take and update our approximations for our previous post-decision state (the post-decision state of the state we were most recently in). By doing so, we update our approximation value of being in said state. To update this, we use the following equation:

$$\bar{V}^n(S^{a,n}) = (1 - \alpha)\bar{V}^{n-1}(S^{a,n}) + \alpha \cdot \hat{v}^n$$

where α represents our step-size.

Finally, we find the new post-decision state $S^{a,n}$ and move to the pre-decision state S^{n+1}, continuing this process for n iterations and returning the final value approximation matrix as our given policy. A breakdown of this algorithm is provided in Algorithm 1.

Algorithm 1 Approximate Dynamic Programming algorithm

Step 0. Initialization

Step0a. Choose an initial approximation $\bar{V}^0$

Step0b. Set n = 1, and the maximum number of iterations N

Step0c. Set the initial post-decision state to S^1

Step 1.

Step1a. Introduce new set of demands

Step1b. Solve the forward optimality equations for $\hat{v}^n$ and $\tilde{a}^n$

Step1c. Update the step-size (Given Harmonic or BAKF)

Step1d. Update the approximation of the previous post decision state $\bar{V}^n(S^{a,n})$

Step1e. Find the new post-decision state $S^{a,n}$ and pre-decision state S^{n+1}

Step 2. Increment n. If $n \leq N$ go to Step 1

Step 3. Return the value functions $\bar{V}^N$

1.3 Experiments and Results

For each of the following three experiments, two sets of performance metrics were set for the ADP algorithm. In the first case, the estimated value of the initial post-state S_0 is measured for varying iteration numbers of n. In the second case, the discounted rewards are measured from using the estimates given the $\bar{V}^n$ at varying values of n. More specifically, N many iterations are run, and every M iterations a set of side-simulations are run for O many iterations calculating the discounted rewards from using those sets of policies. Both sets of performance metric experiments are repeated K many times for a smoothing of results.

In Figure 2, we begin by comparing three varying approaches with regards to the step-size variable *alpha* and following a pure exploitation based methodology. In the first case, we set α to be fixed (F) to $\alpha = 0.05$, while in the second and third cases we follow a harmonic (H) and BAKF (B) step-size. We see that on the left hand side the value function estimations for the initial post-decision state is quite low and far off from what we know to be the optimal solution of the problem. More explicitly, the value of the fixed step-size returns a significantly worse estimation compared with the other two variations. However, on the right hand side we see that this same step-size performs the best when actually utilizing its generated policy to make decisions.

In Figure 3, we introduce exploration into our learning by using an ϵ-greedy approach in which we take a random action ϵ percentage of the time. By doing this, we are able to visit new states and gather information on them to which

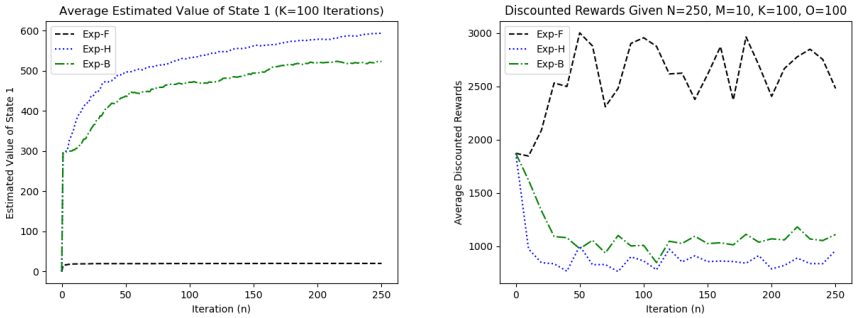

Figure 2: Infinite horizon single-attribute case: resulting estimate $\bar{V}_0^n(S_0)$ and realized rewards

we did not previously do when taking a purely exploitation based approach. We use both the harmonic and the BAKF step-sizes while also experimenting with $\epsilon = 1$ (pure exploration) and $\epsilon = 0.25$ exploration-exploitation policies. We see that though the ϵ-greedy policies improve the discounted rewards, we still do not see a convergence to the true optimal value.

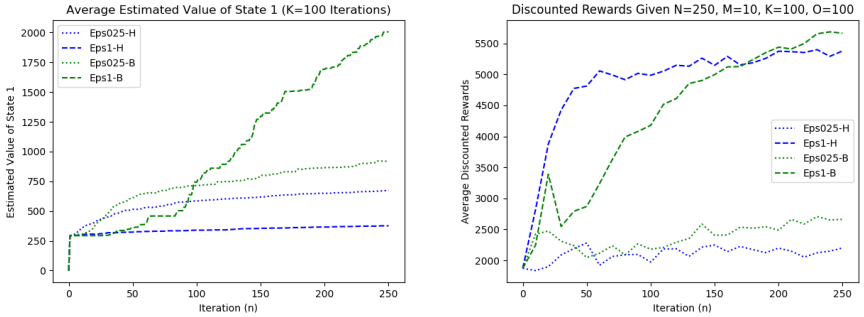

Figure 3: Infinite horizon single-attribute case: resulting estimate $\bar{V}_0^n(S_0)$ and realized rewards

Thus, in the final figure (Figure 4 we run these same experiements for $N = 25,000$ iterations, where we see first that the average estimated value of the first post-decision state converges to near the optimal value for the epsilon-greedy policies via the BAKF step-size. We see this as well for when computing the discounted rewards, thought they take approximately 15,000 iterations to converge.

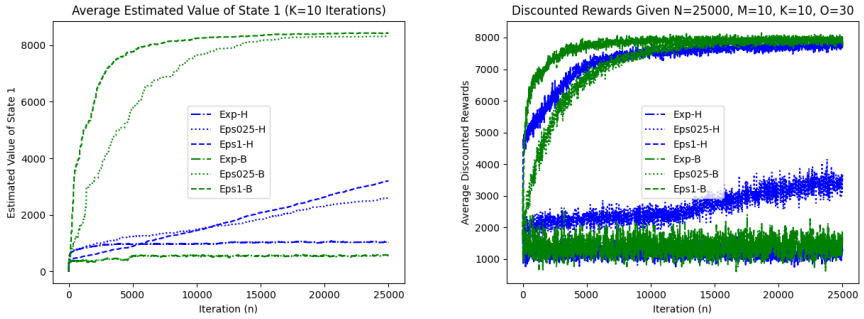

Figure 4: Infinite horizon single-attribute case: resulting estimate $\bar{V}_0^n(S_0)$ and realized rewards