1 Problem statement

We want to build a model that helps users purchase flight tickets at the lowest price possible.

Suppose Alice is planning to travel from Montreal to New York 20 days from now. Flight ticket prices vary everyday depending on many variables, such as, how many places are still available, remaining days before the flight, government regularization ... . Alice wants to know when is the best time to buy her ticket to get the best deal. Tickets can only be bought once, i.e. she is only willing to buy non refundable economy tickets. If Alice reaches the D-Day without purchasing any tickets before, she can travel in business class which is more expensive.

2 Model formulation

Let's denote by N the initial remaining days to travel. For each day $k \leq N$ (starting from day $k = 0$), the flight ticket costs $c(k) = \omega_k$ where ω_k is a random variable that follows a uniform distribution $\mathcal{U}(a_k, b)$. We take $a_k = a \cdot q^k$ with $q > 1$. At day $k + 1$, Alice can decide whether she wants to purchase the flight ticket at the price of $c(k)$ or not. If Alice still hasn't purchased her ticket the day of the flight $k = N$, she has to buy a business class ticket at an expensive price of C. We want to determine the minimum expected cost policy.

We formulate the problem as a dynamic programming problem, where we have:

- the N stages are exactly the initial remaining days to the flight, $k = 0, 1, \ldots, N - 1$[1].
- the state at stage k is represented as $x_k = (\tilde{x}_k, F_k)$ where,

$$x_{k+1} = f(x_k, u_k, \omega_k) = \begin{cases} \omega_k & \text{if } u_0 = \cdots = u_k = 0 \\ \Delta & \text{otherwise} \end{cases} \tag{1}$$

- the action is $u_k = 0$ if we decide not to buy and $u_k = 1$ if we decide to buy the ticket. If $u_k = 1$ for some $0 \leq k \leq N - 1$, then $u_l = 0$ for $l \neq k$. For $x_k = \Delta$ we have $u_k = 0$.
- we denote by $J_k^*(x_k)$ the optimal cost-to-go being in state x_k where we have,

$$J_k^*(x_k) = \begin{cases} 0 & \text{if } x_k = \Delta \\ C & \text{if } k = N \\ \min\left[c(k), \mathbb{E}\left[J_{k+1}^*(\omega_k)\right]\right] & \text{otherwise.} \end{cases} \tag{2}$$

[1] $k = 0$ means there are N days remaining, $k = 1$ means there are $N - 1$ days remaining ...

3 Algorithms

3.1 Exact method

For each stage, we define α_k as[1],

$$\begin{aligned}\alpha_k &= \mathbb{E}\left[J^*_{k+1}(\omega_k)\right] \\ &= \begin{cases} C & \text{if } k = N \\ \mathbb{E}_{\omega_k}\left[\min\{\omega_k, \alpha_{k+1}\}\right] & \text{otherwise.} \end{cases}\end{aligned} \tag{3}$$

The optimal policy will be to buy a ticket if $\omega_k < \alpha_{k+1}$. Note that α_k can be computed recursively starting from $\alpha_N = C$.

The computation of the equations above are done through the backward chaining algorithm (2).

Algorithm 1 Backward Chaining for Stochastic case[2]

Require: $N \geq 0$
  for $x \in X_N$ **do**
    $J_N(x) \leftarrow g_N(x)$
  end for
  for $k = N-1, \ldots, 0$ **do**
    $J_k(x) \leftarrow \min_{u \in U_k(x)} \mathbb{E}\left[g(x_k, u_k, \omega_k) + J_{k+1}(f(x_k, u_k, \omega_k))\right]$
    $\mu^*_k(x) \leftarrow \arg\min_{u \in U_k(x)} \mathbb{E}\left[g(x_k, u_k, \omega_k) + J_{k+1}(f(x_k, u_k, \omega_k))\right]$ ▷ Optimal policy
  end for

3.2 Approximate method

Regarding the approximate method, we use a linear feature-based architecture. We will approximate the true cost-to-go function J^*_k defined in equation (2) by a class of functions $\tilde{J}_k(x_k, r_k)$ where $\boldsymbol{r}_k = (r_{k,1}, \ldots, r_{k,m})$ are a set of parameters that will be learned sequentially starting from $\boldsymbol{r}_N$ and propagating backwards towards $k = 0$. This is done by sampling the q elements x^s_k from the state for each k. The parameters r are learned over the training data (x^s_k, β^s_k) with $s = 1, \ldots, q$, using the mean squared error as our loss function[3],

$$\hat{\boldsymbol{r}}_k = \arg\min_{r_k} \sum_{s=1}^{q} \left(\tilde{J}_k(x^s_k, r_k) - \beta^s_k\right)^2, \tag{4}$$

$$\beta^s_k = \mathbb{E}\left[g(x^s_k, u, \omega_k) + \tilde{J}_{k+1}\left(f_k(x^s_k, u, \omega_k), \hat{\boldsymbol{r}}_{k+1}\right)\right]. \tag{5}$$

One good approximation to our original cost-to-go is through a linear weighting of the states with feature extraction, that is

$$\tilde{J}_k(x_k, r_k) = \hat{J}_k(\phi_k(x_k), r_k) = \sum_{l=1}^{m} r_{k,l}\phi_{k,l}(x_k). \tag{6}$$

The pseudo code of the algorithm used is given in (2).

Algorithm 2 Approximate method[4]

Data: Architecture of $\tilde{J}_k(x_k, \hat{\boldsymbol{r}}_k)$ $k = 1, \ldots, N-1$
Result: Vector $\hat{\boldsymbol{r}}_k$ (i.e. $\tilde{J}_k(x_k, \hat{\boldsymbol{r}}_k)$) $k = 1, \ldots, N-1$

$q \leftarrow Q$ ▷ number of samples per step k
$\tilde{J}_N(x_N) \leftarrow C$
$\hat{\boldsymbol{r}}_N \leftarrow [C, \mathbf{0}]$
$k \leftarrow N-1$
while $k \geq 0$ **do**
  $s \leftarrow 1$
  while $s \leq q$ **do**
    sample a state x_k^s ▷ Monte Carlo method
    $\beta_k^s \leftarrow \min_{u \in U_k(x_k^s)} \mathbb{E}\left[g(x_k^s, u, \omega_k) + \tilde{J}_{k+1}\left(f_k(x_k^s, u, \omega_k), \hat{\boldsymbol{r}}_{k+1}\right)\right]$
    save the example $\left(x_k^s, \beta_k^s\right)$
    $s \leftarrow s+1$
  end while
  $\hat{\boldsymbol{r}}_k \leftarrow \arg\min_{r_k} \sum_{s=1}^{q} \left(\tilde{J}_k(x_k^s, r_k) - \beta_k^s\right)^2$ ▷ Learning problem on training data $\left(x_k^s, \beta_k^s\right)$
  $k \leftarrow k-1$
end while

As mentioned in the beginning of this section, the linear weighting of the states with feature extraction is a good approximation to our original cost-to-go. Using the least squared method as our loss function gives a simple solution to our linear architecture,

$$\begin{aligned}\hat{\boldsymbol{r}}_k &= \arg\min_{r_k} \sum_{s=1}^{q} \left(\tilde{J}_k(x_k^s, r_k) - \beta_k^s\right)^2 \\ &= \left(\sum_{s=1}^{q} \phi_k(x_k^s, u_k^s)\phi_k(x_k^s, u_k^s)^\top\right)^{-1} \sum_{s=1}^{q} \phi_k(x_k^s, u_k^s)\beta_k^s \end{aligned} \tag{7}$$

4 Results and analysis

We create three instances to solve with different models, one small $N = 2$, the second being medium $N = 10$ and the third larger $N = 100$.

Parameters: we chose the following values for the parameters,

$$a = 200, \qquad b = C, \qquad q = 2^{1/(N-1)}, \qquad \text{and } C = 600.$$

4.1 Exact model

- **Small instance:** We put $N = 2$ and generate ω_k using the uniform distribution $\mathcal{U}(a_k, b)$. We find

  $$\omega_0 = 276.61, \qquad \text{and} \qquad \omega_1 = 524.42.$$

  Then we compute α_k using equation (3). By definition $\alpha_{N=2} = 600$. Now we solve recursively for $k = 1$,

  $$\begin{aligned} \alpha_1 &= \mathbb{E}_{\omega_1}\left[\min\{\omega_1, \alpha_2\}\right] \\ &= 500. \end{aligned}$$

  The optimal policy is then,

  - for $k = 1$, we have $\omega_0 = 276.61 \leq \alpha_1 = 500$ and therefore $u_1 = 1$,
  - for $k = 2$, since $u_1 = 1$ then $u_2 = 0$.

  Therefore the analytical solution is $\boldsymbol{u} = (1, 0)$.

- **Medium instance:** We put $N = 10$ and generate ω_k using the uniform distribution $\mathcal{U}(a_k, b)$. We find,

  $$\begin{aligned} &\omega_0 = 276.61, \quad \omega_1 = 454.89, \quad \omega_2 = 393.82, \quad \omega_3 = 525.3, \quad \omega_4 = 527.87 \\ &\omega_5 = 377.37, \quad \omega_6 = 395.59, \quad \omega_7 = 549.06, \quad \omega_8 = 590.39, \quad \omega_9 = 575.19. \end{aligned}$$

  We then compute α_k recursively starting with $\alpha_{10} = 600$ and we find,

  $$\begin{aligned} &\alpha_1 = 313.38, \quad \alpha_2 = 330.42, \quad \alpha_3 = 348.52, \quad \alpha_4 = 367.79, \quad \alpha_5 = 388.4 \\ &\alpha_6 = 410.65, \quad \alpha_7 = 435.16, \quad \alpha_8 = 463.4, \quad \alpha_9 = 500.00, \quad \alpha_{10} = 600. \end{aligned}$$

  The optimal policy is then found by comparing α_k and $c(k)$ as in (2). We find,

  $$\boldsymbol{u} = (0, 0, 0, 0, 0, 0, 0, 0, 1, 0).$$

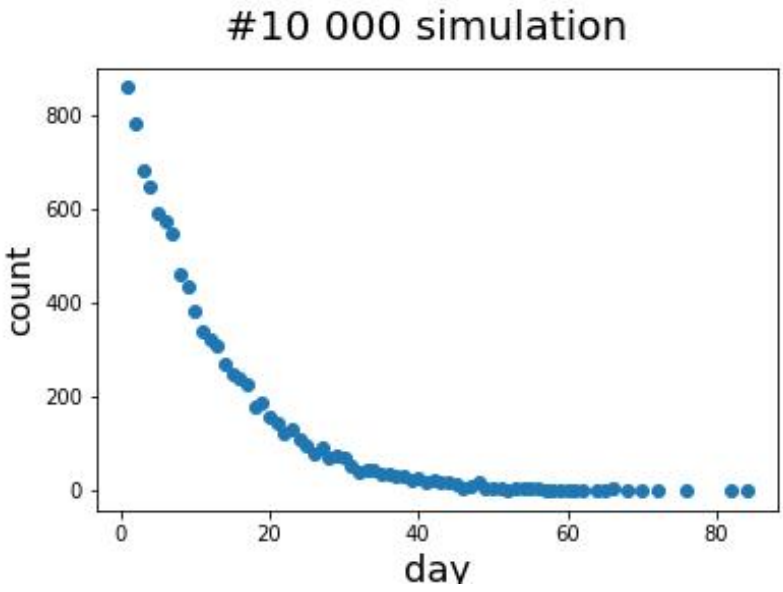

Figure 1: plot of the average optimal policy of 10^4 simulation.

- **Large instance:** We put $N = 100$ and generate ω_k using the uniform distribution $\mathcal{U}(a_k, b)$

  The optimal policy is then found by comparing α_k and $c(k)$ as in (2). We find, $\boldsymbol{u}_{15} = 1$.

  In order to get a better idea for best optimal policy, we run a simulation over 10000 samples and take the average exit time. We present a plot of the simulation result in figure 1. The average optimal policy is to purchase the ticket at day 11.2291.

  The average ticket price is: $\pi = \$\,232.54$.

4.2 Approximate model

We now use the approximate method described in section 3.2. We set $Q = 10$ and we empirically estimate the expected value over a sample of size 100. We use a polynomial feature mapping of degree $l = 1$,

$$\phi_k(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}. \tag{8}$$

- **Small instance:** We put $N = 2$ and start by computing $\hat{\boldsymbol{r}}$ using equation (7). We put $\hat{\boldsymbol{r}}_{N=2} = (C, 0, 0)^\top$. For $k = 1$ we sample $Q = 10$ x_1^s,

$$\begin{aligned} x_1^1 &= 494.45, & x_1^2 &= 272.99, & x_1^3 &= 363.73, & x_{14} &= 382.43, & x_1^5 &= 545.12, \\ x_1^6 &= 296.44, & x_1^7 &= 227.55, & x_1^8 &= 508.31, & x_{19} &= 560.09, & x_1^{10} &= 244.24. \end{aligned}$$

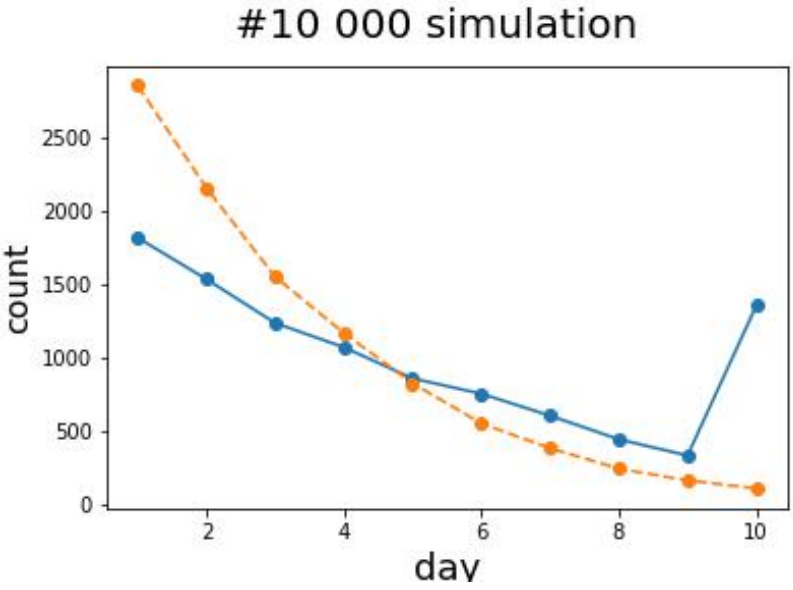

Figure 2: plot of the average optimal policy of 10^4 simulation for $N = 10$. The blue line is for the approximate method and the orange line represents the exact method.

We then compute β_1 using equation (5). We find the same value as x_k^s.

Using equation (7) we find $\hat{\boldsymbol{r}}_1 = (0, \quad 1)$.

We do the same thing for $k = 0$ and find $\hat{\boldsymbol{r}}_0 = \left(1.13 \cdot 10^2, \quad 5.29 \cdot 10^{-1}\right)$.

With the linear parameters $\hat{\boldsymbol{r}}$ computed, we use equation (6) to compute $\tilde{J}$. We find, $\tilde{J}_1 = 486.48$. Comparing $\tilde{J}_1$ with ω_0, we find that it respects the buying condition and therefore break the loop. At the end we have $\boldsymbol{u} = (1, 0)$.

- **Medium instance:** We put $N = 10$ and generate ω_k using the uniform distribution $\mathcal{U}(a_k, b)$. We run a simulation over 10000 samples and take the average exit time. We find for the approximate method an average exit time of 4.5993, while we found 3.1142 for the exact method. We present in figure 2 the results of the simulation.

  Both curves in figure 2 are decreasing with respect to k except for day 10 which seems to increase for the approximate method.

- **Large instance:** We put $N = 100$ and generate ω_k using the uniform distribution $\mathcal{U}(a_k, b)$. We run again a simulation over 10000 samples and take the average exit time. We find for the approximate method an average exit time of 12.1098, while we found 11.2291 for the exact method. We present in figure 2 the results of the simulation.

  Both curves in figure 3 are decreasing with respect to k and follow the same shape. We see again that the approximate method favours the last stage in few cases (only one time). Except for $k = N$, both methods stop at stage 84 at most.

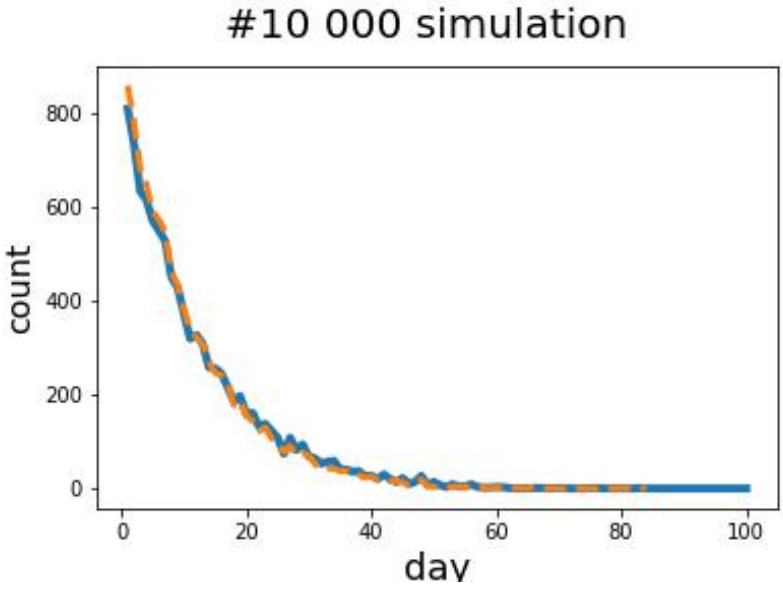

Figure 3: plot of the average optimal policy of 10^4 simulation for $N = 100$. The blue line is for the approximate method and the orange line represents the exact method.

4.3 Performance

Complexity wise, the exact method has a complexity of $O(NI)$ where I is the number of iterations. The complexity for the approximate method is $O(NIQ)$ where the Q comes from the need to generate (x_k^s, β_k^s) data to learn the parameters $\hat{r}_k$.

The average time complexity for 1000 simulation is 1.55 s for the exact method while we find 91.14 s for the approximate method. The results comply with the big oh notation we discussed.

5 Conclusion

All in all, we used two different dynamic programming techniques to solve an optimal stopping problem. The first one being exact and the second one being an approximate method. We see that results become more alike for large instances. This shows that our approximate method can be reliable in cases where we don't know the distribution of the prices. However the approximate method takes longer to run $O(NIQ)$ instead of $O(NI)$ for the exact method. Although the approximate method takes a longer time than the exact one, in case of an unknown distribution of the random variables it is a good solution. The approximate results were almost identical to the exact method with only $Q = 10$ examples to learn from. As for our feature extraction we stopped at a linear mapping, since quadratic terms were found to be in the order of $\sim 10^{-15}$. Therefore a linear mapping was enough.

6 Instructions

We provide a `.ipynb`. It has two main sections: Exact and Approximate method. The stages N and the iteration number as well as the parameters are defined at the beginning of the exact method section.

References

[1] Emma Frejinger, *IFT 6521. Programmation Dynamique, Lecture Slides*, Modèles stochastiques sur horizon fini, page 45.

[2] Emma Frejinger, *IFT 6521. Programmation Dynamique, Lecture Slides*, Introduction, page 41.

[3] Dimitri P. Bertsekas, *Reinforcement Learning and Optimal Control*, Massachusetts Institute of Technology.

[4] Emma Frejinger, *IFT 6521. Programmation Dynamique, Lecture Slides*, Approximation paramétrique dans l'espace des valeurs - horizon fini, pages 13-17.

A Self-assessment

3. Project "advanced" variant	Grade	Motivation
3.1 Problem description	A+	Original problem and consistent with the modeling
3.2 Modelling	A	Clear models, tend to be realistic in some sense
3.3 Algorithms	A	Clearly described and consistent with the results
3.4 Results and analysis	A	Correct solutions. The two methods give similar solutions that are intuitive as well
3.5 Conclusion	A-	We found the desired conclusion. Maybe a comparison with another approximate method could give better conclusions.
3.6 Other parts (instructions, references, self assessment)	A	Complies with instructions
3.7 Code	A	Complies with instructions
3.7 Poster	A-	Reflects all parts of the report