Final Project: Load balancing

Urtzi AYESTA and Matthieu JONCKHEERE

Consider the following simple model for a dispatcher in a data center.

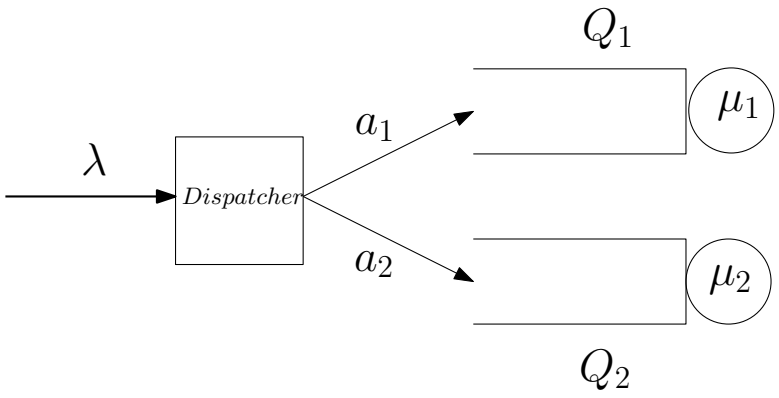

Figure 1: A dispatcher shares the load between two servers

Time is discrete. Let Q_1 and Q_2 denote the total number of jobs in the server 1 and 2, respectively. Every time slot, the dispatcher observes (Q_1, Q_2) and takes the action a_i, $i = 1, 2$, that dispatches a potential new incoming job to server i. The cost in every time slot is $Q_1 + Q_2$, independently of the action.

In every time slot, there is a probability λ of having a new job, which will be dispatched to server a_i. If $Q_i > 0$, with probability μ_i a job from server i will depart. For simplicity, we will assume that in every time slot, only one event can happen, i.e., either an arrival, or a departure from servers 1 or 2.

For example, in the state (2,1), if the dispatcher takes the action 1, we have that with probability λ the next state will be (3,1), with probability μ_1 (1,1), , with probability μ_2 (2,0), and with probability $1 - \mu_1 - \mu_2 - \lambda$ the next state will be (2,1).

We will assume that there is an upper bound for both Q_1 and Q_2 equal to 20. If either $Q_1 = 20$ or $Q_2 = 20$, no new incoming job will arrive to the system.

Let us choose $\mu_1 = 0.2$, $\mu_2 = 0.4$ and $\lambda = 0.3$. Throughout we take that the discouting factor is $\gamma = 0.99$.

1 MDP

1.1 Policy Evaluation

Assume the random policy that dispatches every job with probability 0.5 to either queue 1 and 2.

- Write down the Bellman equation that characterizes the value function for this policy.
- Calculate the value function for this policy using Iterative Policy Evaluation. Due to the contraction principle, the initial vector can be arbitrary, so you can take $V(Q_1, Q_2) = 0$, for all (Q_1, Q_2). To stop iterating, you can take as a criterion that the difference between two iterations must be smaller than some small δ.

  Bonus: The policy could also be evaluated by calculating $V = (I - \gamma P)^{-1} R$, but this would require to translate the 2D state space into vectors. If you do it this way, verify that both solutions coincide.

1.2 Optimal control

In this part you are asked to find the optimal policy to dispatch incoming jobs.

- Write down the Bellman equation that characterizes the optimal policy.
- Solve numerically the optimality value function by Value Iteration Algorithm.
- Represent on the plane the optimal action as a function of the state (Q_1, Q_2).
- Compare the performances obtained with the random policy and the optimal one, how can you conclude that the optimal policy performs better ?
- Carry out a one-step improvement on the random policy, what do you observe ?

2 Tabular Model-Free control

2.1 Policy Evaluation

Assume the random policy that dispatches every job with probability 0.5 to either queue 1 and 2. For the learning parameter you can use $\alpha_n = 1/n$

- Implement TD(0)
- Compare the obtained value function with the results of Section 1.

- Explore other alternatives for α_n, for example constant, to see if convergence improves.

2.2 Optimal control

In this part you are asked to find the optimal policy to dispatch incoming jobs.

- Implement Q-learning
- Represent on the plane the optimal action as a function of the state (Q_1, Q_2).
- Explore other alternatives for α_n, for example constant or $1/n^\gamma$, to see if convergence improves.

3 Model-free control with Value Function/Policy approximation

This section is a bit more open and research oriented. You are expected to propose and implement your own approach. The objective is to devise a scheme that can learn the optimal policy as found in Section 1 more efficiently than in Section 2 using a non-tabular approach . As you have probably found out in Section 2, Q-learning is not very efficient in learning the optimal policy for this problem. Some possible approaches are:

- Approximation for value function: Linear approximation, Neural Networks etc.
- Policy Approximation: softmax parametrization, or other parametrization inspired by the structural results of Section 1.
- Model-based Reinforcement learning using the conclusions from Section 1 on the optimal policy.