A Queuing Problem: 4-Queues

Consider a system with four queues, each with capacity $L = 9$, shown in Figure 5. This problem has been studied in several papers (e.g., Chen and Meyn 1999; Kumar and Seidman 1990; de Farias and Van Roy 2003). There are two servers in the system: Server 1 serves queue 1 and 4 with rates r_1 and r_4, and Server 2 serves queue 2 and 3 with rates r_2 and r_3. Each server serves only one of its associated queues at each time. Jobs arrive at queue 1 and 3 with rate λ. A job leaves the systems after being served at either queue 1 and 2 or queue 3 and 4. The state of the system is denoted by a 4-dimensional vector $[x_1, x_2, x_3, x_4]$, where x_i represents the number of jobs in queue i at each time. A controller can choose a 4-dimensional action from $\{0, 1\}^4$ such that $a_1 + a_4 \leq 1$ and $a_2 + a_3 \leq 1$. The cost at each time is equal to the total number of jobs in the system: $c(x) = \sum_{i=1}^{4} x_i$.

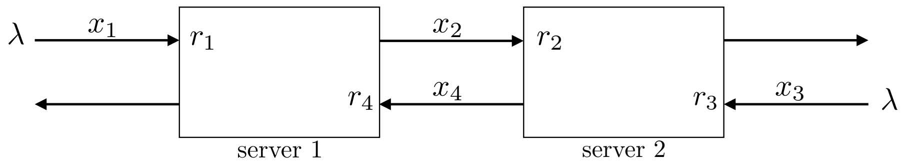

Figure 5: A system with four queues and two servers

Assume $r_1 = r_2 = 0.12$, $r_3 = r_4 = 0.28$, and $\lambda = 0.08$. We choose our base policies from a family of policies for which server i serves its longer queue w.p. p_i and its shorter queue w.p. $1 - p_i$, $i = \{1, 2\}$. Five base policies and their associated costs are shown in Table 2.

Table 2: Base Policies for the 4-Queue problem.

	p_1	p_2	cost
1	0.9	0.9	16.2950
2	0.9	0.7	17.3926
3	0.8	0.8	15.1535
4	0.7	0.9	13.6525
5	0.7	0.7	14.3266

Solving this problem in the space of policies using policy gradient will result in the optimal weight vector $w^* = [0, 0, 0, 1, 0]$, i.e., giving all the weight to the policy with the lowest cost. Therefore, the cost of the mixture policy will be $J(\pi^*) = 13.6525$. Interestingly, if we solve the problem in the dual space (space of stationary distributions) using policy gradient and Eq. 3, after 200 iterations, the optimal resulting policy will have cost $J(\pi_\theta) = 12.84$ with $\theta = [-0.32, -0.68, -0.4, 2.16, 0.24]$ (note that based on the definition of Θ, this vector can have negative values). In either of the spaces, in order to approximate the gradient surface for the policy gradient method, we need to compute the cost of each mixed policy a couple of times (depending on the number of base policies) in each iteration. Computing the cost involves an eigen-decomposition of the transition matrix, which makes the whole process computationally costly. Using our stochastic sub-gradient method, we can approximate the cost very fast and efficiently. Our method needs only one eigen-decomposition for the final mixing weights. Therefore, the computational cost is much lower (in this particular example $1/(8 \times 200)$ of the policy gradient method, where 8 is the number of points we use for approximating the gradient surface and 200 is the number of iterations). The policy resulted from our method will have cost $J(\pi_\theta) = 13.00$ with $\theta = [-0.23, -1.28, 0.09, 1.54, 0.88]$. Figure 6 shows the difference between the costs of policy gradient and the stochastic sub-gradient method at each iteration.

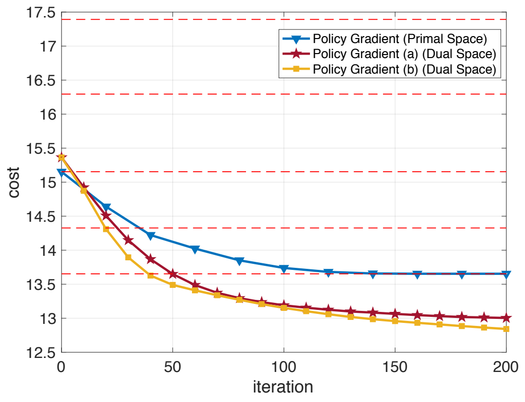

Figure 6: Cost per iteration for the primal and dual spaces. The policy gradient (b) for the dual space is the method described in Algorithm 1. Horizontal dashed lines are the costs of the base policies.