Queueing Networks

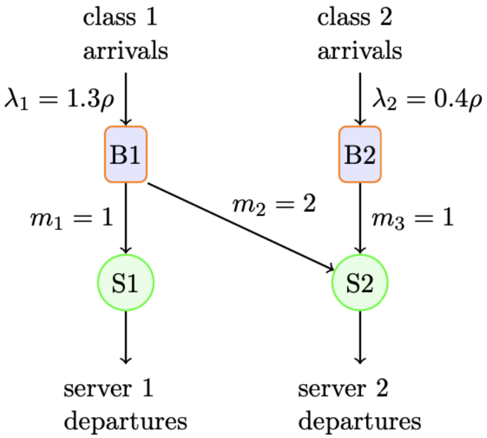
Trang H. Tran and Tanishq Aggarwal

Cornell University

May 29, 2021

N-Queue

One of the queueing system from Homework 2: the N-queue model



Usually consist of some customer lines and service stations

Queueing Networks

- Are an important set of problems in operation research. The question is to find the optimal way to serve the customers.
- When the arrival times of customers and service times of servers are exponentially distributed, we can model queueing networks as MDPs via uniformization. Hence the model is known when we know the input parameters (arrival rates, service rate, workload)

Queueing Networks

- Are an important set of problems in operation research. The question is to find the optimal way to serve the customers.
- When the arrival times of customers and service times of servers are exponentially distributed, we can model queueing networks as MDPs via uniformization. Hence the model is known when we know the input parameters (arrival rates, service rate, workload)
- Existing work include TRPO and PPO for discounted setting ([4] and [5]), however we are interested in the long-run average cost

$$\inf_{\pi} \lim_{N\to\infty} \frac{1}{N} \mathbb{E}_{\pi} \left[\sum_{k=0}^{N} c^{\top} x^{(k)} \right],$$

  where c is a cost vector.
- Approximate linear programming (ALP) [6] and Dai & Gluzman's work on average PPO [2] explores this cost setting.

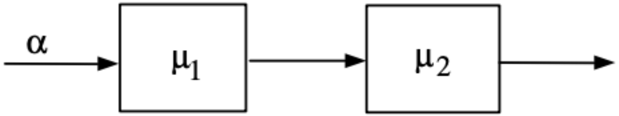
General environment for Queueing Networks

We create a common Gym environment:

- State space: finite subsets of $\mathbb{N}^n$, where n is the number of queues in the network
- Action space: maybe idle or work on different classes of jobs
- Transitions: occur due to arrivals of jobs or completions of jobs. As discussed previously (and done on HW) these are modeled as Poisson processes

The N-queues, as well as other models, are the subclasses of our environment.

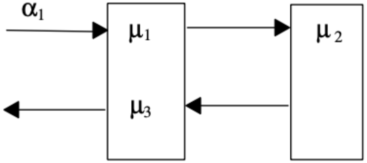
The Two-Series Queue



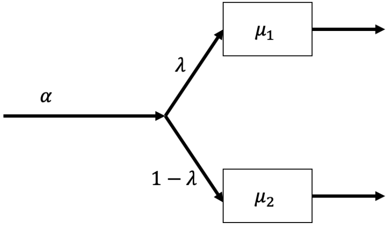
- Simple example of queueing network
- Two queues, two servers, each server can choose between Work or Idle
- *Non-idling policy is optimal*, so it's useful as a toy problem for our implementation of algorithms

Other Networks

Three-class Reentrant

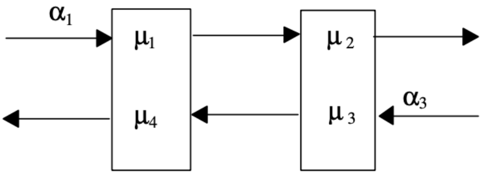


Arrival Routing

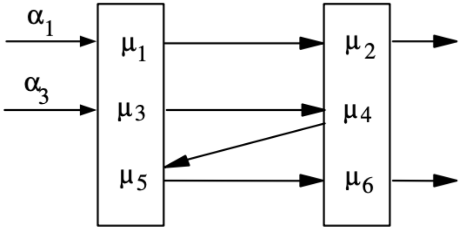


Other Networks

Rybko-Stoylar



Six-class Network



Parameters for models are given in [6].

The Standard PPO algorithm (Source: [1])

Algorithm 1 PPO-Clip

```
1: Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
2: for $k = 0, 1, 2, \ldots$ do
3:     Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4:     Compute rewards-to-go $\hat{R}_t$.
5:     Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based
       on the current value function $V_{\phi_k}$.
6:     Update the policy by maximizing the PPO-Clip objective:
           $\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$
       typically via stochastic gradient ascent with Adam.
7:     Fit value function by regression on mean-squared error:
           $\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2,$
       typically via some gradient descent algorithm.
8: end for
```

Advantage Estimation for Custom PPO

The difference between standard PPO and our average cost-modified version is the expression for the advantage estimate [3]:

$$\hat{A}_\theta(s_t^k, a_t^k) = \sum_{i=0}^{T_0-1} (r(s_{t+i}^k, a_{t+i}^k) - \mu_\theta(r) + \hat{V}_\theta(s_{t+i+1}^k) - \hat{V}_\theta(s_{t+i}^k))$$

$$(\text{for } 0 \leq t \leq T - T_0)$$

$$\hat{A}_\theta(s_t^k, a_t^k) = \sum_{i=0}^{T-t-1} (r(s_{t+i}^k, a_{t+i}^k) - \mu_\theta(r) + \hat{V}_\theta(s_{t+i+1}^k) - \hat{V}_\theta(s_{t+i}^k))$$

$$(\text{for } T_0 - T \leq t \leq T)$$

Variance Reduction

The variance in the advantage estimate can be reduced if we replace the value function estimate with an expectation, i.e. replace $\hat{V}_\theta(s^k_{t+i+1})$ with $\mathbb{E}_{s\sim P^\theta(\cdot|s^k_{t+i})}[\hat{V}_\theta(s)]$ so that

$$\begin{aligned}\hat{A}_\theta(s^k_t, a^k_t) =& r(s^k_t, a^k_t) - \mu(r) + \mathbb{E}_{s\sim P(\cdot|s^k_t, a^k_t)}[\hat{V}_\theta(s)] - \hat{V}_\theta(s^k_{t+i}) \\ &+ \sum_{i=1}^{T_0-1}(r_\theta(s^k_{t+i}) - \mu_\theta(r) + \mathbb{E}_{s\sim P^\theta(\cdot|s^k_{t+i})}[\hat{V}_\theta(s)] - \hat{V}_\theta(s^k_{t+i})) \\ &(\text{for } 0 \le t \le T - T_0)\end{aligned}$$

and another similar expression for $T_0 - T \le t \le T$. Performing this replacement requires knowing the model transition probabilities, which we do.

Randomized Non-Idling Policy

Average cost taken over $T = 20$k timesteps and 100 episodes.

Networks	Average Cost	Regeneration Cycles
Two-Series Queue ($\mu_1 > \mu_2$)	4.71 ± 0.06	4.91 ± 18.46
Two-Series Queue ($\mu_1 < \mu_2$)	8.00 ± 0.06	9.68 ± 40.10
N-Queue	3.79 ± 0.06	9.29 ± 39.40
Arrival Routing	4.34 ± 0.04	4.91 ± 17.08
Three Class Reentrant	15.02 ± 0.39	206.81 ± 1250.83
Rybko-Stolyar	16.08 ± 0.83	267.84 ± 1876.97
Six-Class Network	9.09 ± 0.54	136.41 ± 1621.67

- Regeneration cycle length is small, validates our use of long horizons ($T = 10000$) in remainder of training.
- Last three models are significantly more complex (see cycle length).
- Two-series queue results are optimal.

Standard PPO

Networks	Cost: $\gamma = 1$	Cost: $\gamma = 0.99999$
Two-Series Queue ($\mu_1 > \mu_2$)	5.50 ± 0.61	364.64 ± 2.12
Two-Series Queue ($\mu_1 < \mu_2$)	368.13 ± 1.31	367.75 ± 1.33
N-Queue	93.72 ± 0.92	94.66 ± 1.11
Arrival Routing	2.81 ± 0.27	3.72 ± 0.23
Three Class Reentrant	38.32 ± 0.13	38.21 ± 0.08
Rybko-Stolyar	279.04 ± 14.34	292.01 ± 11.29
Six-Class Network	149.14 ± 7.31	149.99 ± 7.92

- Performance is similar across both settings of γ
- We get a good cost for the first two-series model?
- PPO-computed policies are all worse than the randomized policy, except for arrival routing—a very simple network. It's hard to pick right hyperparameters even for simple models.

Custom PPO

Networks	Average Cost No VR	Average Cost With VR
Two-Series Queue ($\mu_1 > \mu_2$)	362.91 ± 1.87	29.15 ± 21.03
Two-Series Queue ($\mu_1 < \mu_2$)	367.36 ± 1.59	394.14 ± 6.96
N-Queue	102.47 ± 9.80	102.69 ± 1.58
Arrival Routing	5.18 ± 0.64	3.19 ± 0.28
Three Class Reentrant	44.60 ± 3.09	39.09 ± 1.50
Rybko-Stolyar	573.02 ± 17.66	564.03 ± 14.42
Six-Class Network	154.08 ± 8.17	151.11 ± 9.46

- Results slightly worse than standard PPO.
- Variance reduction seems to work better then no VR version, but with longer training time.

Conclusions

- Implemented a special version of PPO with average-cost advantage estimation + variance reduction
- Had trouble finding good hyper-parameters for the set of models
- Future areas of interest:
  - We have perfect knowledge of the model and yet we don't use it much—can this help inform a better algorithm design?
  - Explore how knowing regeneration cycle length can help with hyper-parameter selection

References I

Proximal policy optimization.
https://spinningup.openai.com/en/latest/algorithms/ppo.html.

J. G. Dai and Mark Gluzman.
Queueing network controls via deep reinforcement learning, 2020.

J. G. Dai and Mark Gluzman.
Average-cost mdp: Foundation and algorithms for trpo and ppo, 2021.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel.
Trust region policy optimization, 2017.

References II

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.
Proximal policy optimization algorithms, 2017.

Michael H. Veatch.
Approximate linear programming for networks: Average cost bounds.
Computers & Operations Research, 63:32–45, 2015.