Multiclass Queueing Networks Scheduling Optimization Using Deep Reinforcement Learning

Mark Gluzman
7/11/2019

1 Introduction

In recent years deep reinforcement learning (RL) has become extremely popular research area in AI community with real-world impact. Decision strategies obtained from RL algorithms outperform human in one-player games [6, 27], two-players games [38], team games [30]; are successful in wide range applications from medicine [20] to cooling datacenters [13]. Following the trend many dynamic resource allocation and sequential decision making problems in communications and networking are solved applying deep RL, see the review [9].

One of the hard problems in stochastic processing networks domain is the optimization scheduling policies for multiclass queueing networks (MQNs). In the scheduling optimization problem the decision maker determines which job class should be precessed first in each node (server) of the system.

One approach to optimize scheduling policies is to assume that the multiclass queueing networks is heavily loaded. Then the original scheduling problem can be approximated by a reflected Brownian motion (RBM) [14, 15, 43]. However, determining the stationary distribution of the reflected Brownian motion in higher than two dimensions is hard and, except the cases with state-space collapse [44, 21], this approach is untractable for higher dimensions.

Another approach is to consider fluid model approximations. The strong connection between stability of multiclass queuing network and the associated fluid model has been shown in [10]. There are several papers where authors develop methods to translate a policy derived from fluid optimal control problem into a policy for the original stochastic processing network, for example affine shift policies [25], discrete-review policies [24], and tracking policies [4].

While fluid models are more tractable than RBM, they ignore the "variance" of the associated original stochastic processing network. In [8] the authors propose to assume that realization of the arrival rates and service times can deviate from their mean values in some pre-set interval. The method allows to inject uncertainty in the fluid model. The optimization problem based on a new robust fluid model remains trackable for large multiclass networks. Each decision epoch the decision maker formulates and solves a robust fluid optimization problem for a current system state. It requires knowledge of nominal mean arrival rates and service times, as well as routing probabilities.

For the queuing networks with Poisson arrival and exponential service time assumptions the control problem can be modeled within the framework of Markov decision processes

(MDPs) via uniformization. Based on information that is available for the system manager we distinguish two approaches of solving MDPs: approximate dynamic programming and reinforcement learning.

In dynamic programming paradigm the decision maker has full knowledge about the MDP problem (transition probabilities, cost-to-go function, etc.). A fundamental difficulty in solving the MDP problem directly is the curse of dimensionality: the buffers capacity is unlimited and the corresponding optimization problem has infinite number of constrains and unknowns. Even if the number of waiting jobs in each buffer is limited, the complexity of the problem grows exponentially with the number of job classes. Approximate dynamic programming (ADP) aims to approximate the original MDP to make the problem computationally trackable and yet obtain near optimal control policies. Usually additional assumptions on the value function structure are imposed, see [11, 2, 42].

In practice the system manager may not have full information about arrival, service processing rates and routing path of each job class. The goal of reinforcement learning algorithm is to find near-optimal policies when a probabilistic model of the system is not provided, and only the current state of the queuing network is known.

The following RL methods have been applied for queuing scheduling control problem: look-up table Q-learning [32], SARSA(λ) with linear Q-functions representation [33], actor-critic method [34], policy gradient [31], MCTS [7]. Except the last paper, only small networks with no more than six job classes have been considered. In [7] the MCTS method has been compared with fluid network approach [8] and shown similar or worse results.

In this paper we describe an iterative deep RL algorithm that can be classified as a conservative policy iteration algorithm [17]. Conservative policy iteration algorithms find a gradient direction that guarantees monotonic improvement of a current policy, but constrain the magnitude of policy update to omit performance collapse caused by large changes in the policy. In [35] the authors prove that minimizing a certain surrogate objective function guarantees decreasing of expected discounted cost. Unfortunately, the theoretically-justified step-sizes of policy updates cannot be computed from available information for the RL algorithm. Trust Region Policy Optimization (TRPO) [35] has been proposed as a practical method to restrict policy updates. Proximal policy optimization (PPO) is an alternative way of adjusting step-sizes based on clipped surrogate objective [37].

In Section 3.1 we provide theoretical justification that minimizing surrogate objective function with cost discounting can guarantee monotonic decrease of the long-run average cost. In Section 3.2 we describe in details practical modifications to the algorithm, including PPO step-size control.

We mostly compare performance of our RL policies with the performance of robust fluid policies reported in [8]. Robust fluid policies yield performance that is near-optimal for small-size networks, and have better performance for moderate and large-size networks in comparison with the best other heuristic policies. We want to note several differences in the problem formulation for a RL algorithm and robust fluid optimization problem:

- robust fluid problem requires knowledge of all parameters of the system including mean arrival rates, mean service times, routing probabilities and cost of each transition. In the RL framework no information about transition probabilities and cost-to-go function is assumed to be available.
- RL algorithms rely on simulations of multiple policies. We assume that a simulation environment is available for the decision maker. Such simulation environment can be a real system or a virtual/generative model.

We summarize the major conclusions from our study:

1. We propose a reinforcement learning method based on PPO [37] algorithm for the multiclass processing network control problem. To the best of our knowledge, this is the first paper which proposes an algorithm for average number of jobs minimization that is trackable and does not require first-order system data (arrival rates, mean processing times, and routing probabilities).
2. We provide an iterative procedure for minimizing long-run average cost based on advantage function with discounted future costs, with guaranteed monotonic improvement. We discuss several approximations to the theoretically-justified procedure to get a practical algorithm.
3. We provide extensive computational experiments and claim the following:
   (a) The performance of RL policies is comparable with model-based fluid network approach [8], [7] and better than baseline heuristics.
   (b) The choice of hyperparametrs (such as number of episodes, their duration, discount factor, clipping parameter, etc.) can affect convergence of the algorithm and performance of RL policies. We provide guidance how to choose hyperparametres.
   (c) We propose a neural network architecture that effectively parameterizes scheduling policies.

2 Control of multiclass queueing networks

In this section we formulate the scheduling control problem for multiclass processing networks. We first give the optimal control problem formulation for the criss-cross network and then describe the formulation of the problem for a general multiclass queueing network.

2.1 The criss-cross network

The criss-cross network has been considered in [16] and depicted in Figure 1. It consists of two stations that process three classes of jobs. Each job class has its own affiliated buffer where jobs wait to be served.

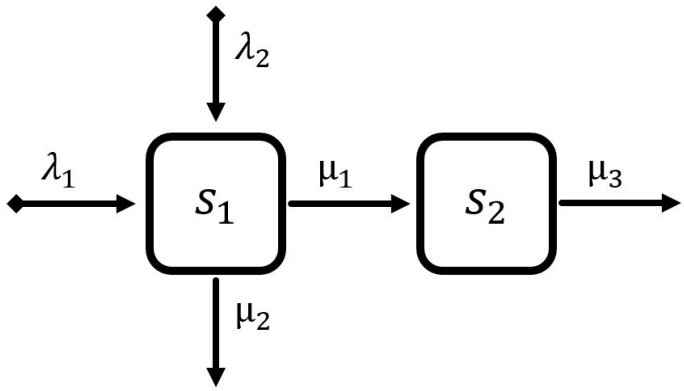


Figure 1: The criss-cross network

We assume that the jobs of class 1 and class 2 arrive to the system following Poisson process with rate λ_1 and with rate λ_2 correspondingly. Both classes are served in the 1st server. After being served jobs of class 1 become jobs of class 3 and wait for 2nd server service. Jobs of the 2nd and 3rd classes leave the system after their service completion. Service times are i.i.d. having exponential distribution with mean $m_j,\ j = 1, 2, 3$.

We assume that both servers employ an non-idling service policy, which means that each server must be busy whenever there is a job available for processing. The only decision that the system manager has to make is to choose whether to process a job of class 1 or a job of class 3 if both buffers 1 and 3 are non-empty.

Let $x_j(t)$ be the number of class i jobs (including possibly the one in service) in the system at time $t,\ j = 1, 2, 3$. We use $x(t) = [x_1(t), x_2(t), x_3(t)]$ to denote the system state at time t. We assume that the system manager executes a stationary Markovian policy π i.e. the policy π dictates a preemption-resume priority for jobs of either class 1 or class 3 solely based on the system state $x(t)$ at time t.

The objective is to find a stationary policy that minimizes the long-run average number of jobs in the network:

$$\inf_{\pi} \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\pi} \int_0^T \Big[x_1(t) + x_2(t) + x_3(t) \Big] dt \tag{2.1}$$

Under classical assumptions on arrival and service processes we can convert the original continuous-time problem to an equivalent continuous-time Markov process where the average time between transitions is constant. The problem becomes probabilistically equivalent to discrete-time control problem [22].

We let $\nu = \lambda_1 + \lambda_2 + \sum_{i=1}^{3} \mu_i$ denote uniform transition rate, where $\mu_i = \frac{1}{m_i}$, $i = 1, 2, 3$.

We can consider the equivalent discrete-time control problem, where depending on control, $a = 1$ (class 1 has preemption-resume high priority) or $a = 2$, the transition probabilities are given by:

$$
\begin{aligned}
P_{(x_1,x_2,x_3),(x_1-1,x_2,x_3)}(a=1) &= \frac{\mu_1}{\nu} 1_{x_1>0} \\
P_{(x_1,x_2,x_3),(x_1,x_2-1,x_3)}(a=1) &= \frac{\mu_2}{\nu} 1_{x_2>0,x_1=0} \\
P_{(x_1,x_2,x_3),(x_1-1,x_2,x_3)}(a=2) &= \frac{\mu_1}{\nu} 1_{x_1>0,x_2=0} \\
P_{(x_1,x_2,x_3),(x_1,x_2-1,x_3)}(a=2) &= \frac{\mu_2}{\nu} 1_{x_2>0} \\
P_{(x_1,x_2,x_3),(x_1+1,x_2,x_3)} &= \frac{\lambda_1}{\nu} \\
P_{(x_1,x_2,x_3),(x_1,x_2+1,x_3)} &= \frac{\lambda_2}{\nu} \\
P_{(x_1,x_2,x_3),(x_1,x_2,x_3-1)} &= \frac{\mu_3}{\nu}
\end{aligned} \tag{2.2}
$$

The probability of having a fictitious transition is

$$
\begin{aligned}
P_{(x_1,x_2,x_3),(x_1,x_2,x_3)} = 1 - P_{(x_1,x_2,x_3),(x_1+1,x_2,x_3)} - \\
P_{(x_1,x_2,x_3),(x_1,x_2+1,x_3)} - P_{(x_1,x_2,x_3),(x_1-1,x_2,x_3)} - P_{(x_1,x_2,x_3),(x_1,x_2-1,x_3)} - P_{(x_1,x_2,x_3),(x_1,x_2,x_3-1)}
\end{aligned}
$$

The "fake" event does not represent any real jump in the original CTMC, but allows to unify average time between real transitions.

We abuse the notation and denote a state as $x(k) = [x_1(k), x_2(k), x_3(k)]$ after k transitions of the DTMC.

The objective (2.1) is equivalent to:

$$
\inf_{\pi} \sum_{N\to\infty} \frac{1}{N} \mathbb{E}_{\pi} \left[\sum_{k=0}^{N-1} (x_1(k) + x_2(k) + x_3(k)) \right]. \tag{2.3}
$$

2.2 General formulation

Let L be a total number of services and I be the total number of buffers. We let $x(t) = [x_1(t), x_2(t), ..., x_J(t)]$ denote the number of jobs of each class i at time t in the system, including a job that might be in service. We interpret $s(i)$ as the server associated with job class i. We define $B_l = \{j, j = 1, .., J : \ s(j) = l\}$ as a set of classes that are processed by server l. We assume that routing in the network can be probabilistic: after being served job of class j becomes job of class k with probability u_{jk} and leave the network with probability $1 - \sum_{k=1}^{K} u_{jk}$. We let $J \times J$ matrix $U = (u_{mn})_{m,n=1,...,J}$ denote the routing probabilities. External arrival processes are Poisson with rate λ_j for jobs of class j, $j = 1, .., J$. If no

external arrivals enter jth buffer we take $\lambda_j = 0$. Class j service times are i.i.d. having exponential distribution with mean $1/\mu_j$, $j = 1, ..., J$.

For each class $j = 1, .., J$, let α_j be the total arrival rate into buffer j, that besides external arrivals involves transitions of jobs into class j from other classes. The vector $\alpha = [\alpha_1, \alpha_2, ..., \alpha_J]$ satisfies the following system of linear equations, usually called as traffic equations:

$$\alpha = \lambda + U^T \alpha. \tag{2.4}$$

We define the utilization of server l as

$$\rho_l = \sum_{j \in B_l} \frac{\alpha_j}{\mu_j}. \tag{2.5}$$

We assume that $\rho_l < 1$ for all services $l = 1, ..., L$.

The objective is to find a stationary Markovian policy π that minimize the expected long-run average number of jobs in the system:

$$\inf_{\pi} \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{\pi} \Big[\int_0^{T-1} \sum_{j=1}^{J} x_j(t) dt \Big] \tag{2.6}$$

We use uniformization to formulate MDP problem with state space $X = \mathbb{Z}_+^J$.

We let $a_j(k)$ denote a control at kth decision epoch that provides priority to jth class, i.e. $a_j(k) = 1$ means that class j has the priority over other associated to server $s(j)$ classes to be served right after kth decision epoch. We note that $\sum_{j \in B_l} a_j(k) = 1$ for each server $l = 1, .., L$ at each time-step $k = 0, 1, 2....$ We denote the action space of the MDP problem as A.

The objective of the MDP problem is to find a stationary Markovian policy $\pi : X \to A$ that minimizes long-run average number of jobs in the system, namely:

$$\inf_{\pi} \lim_{N \to \infty} \frac{1}{N} \mathbb{E}_{\pi} \Big[\sum_{k=0}^{N-1} \sum_{j=1}^{J} x_j(k) \Big] \tag{2.7}$$

3 Reinforcement learning approach for queuing network control

The undeniable success of RL has been reached for games [27], [6], and physics engines [12], [41]. These problems are episodic in nature and have long but finite horizon. In the game domain each new episode (game) may require a different amount of time for the player to finish. To compare the effect of different actions the objective is normalized by discounting future costs. Many algorithms, including TRPO and PPO, have been designed to optimize the infinite horizon discounted total cost.

In some domains, for example stochastic processing networks, it is more appropriate to optimize the long-run average cost. As has been noted in [40] many actor-critic algorithms, that claim to solve discounted cost minimization problem, can be more appropriate to solve average cost problems. Once initial states for the episodes are sampled from a stationary distribution of a current policy, the discounted total cost is proportional to the average performance of the the policy, see Theorem 3 in Section 6.2.

In this section we argue that our version of Proximal Policy Optimization algorithm is appropriate to optimize the long-run average performance, even with discounting future costs.

3.1 Monotonic improvement guarantee for average cost objective

Consider a MDP problem with finite state space X, action state A, one-step cost function $g(x,a)$ and transition kernel $P(\cdot|x,a)$.

Definition 1. We denote the long-run average cost of policy π as

$$\eta_\pi = \mathop{\mathbb{E}}_{\substack{x\sim\mu_\pi \\ a\sim\pi(\cdot|x)}} g(x,a), \tag{3.1}$$

where $g(x,a)$ is a one-step cost, μ_π is a stationary distribution induced by policy π.

Definition 2. Let β be the discount factor. We denote the infinite horizon discounted state-action value function of policy π by

$$Q_\pi^\beta(x,a) = \mathop{\mathbb{E}}_{\Gamma\sim\pi} \sum_{t=0}^{\infty} \Big[\beta^t g(x_t,a_t)\Big], \quad \text{for each state-action pair } (x,a)\in X\times A \tag{3.2}$$

where $\Gamma = (x_0,a_0,x_1,a_1,....)$ is a trajectory that follows policy π: $x_0 = x$, $a_0 = a$, $x_{t+1}\sim P(\cdot|x_t,a_t)$, $a_t\sim\pi(\cdot|x_t)$.

We denote the discounted value function of policy π by

$$V_\pi^\beta(x) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|x)} Q_\pi(x,a), \quad \text{for each } x\in X \tag{3.3}$$

We denote the discounted advantage function of policy π by

$$A_\pi^\beta(x,a) = Q_\pi^\beta(x,a) - V_\pi^\beta(x), \quad \text{for each } (x,a)\in X\times A \tag{3.4}$$

The proof of the theorem can be found in Section 6.2.

Theorem 1. *For any function $f: X\to\mathbb{R}$ and any policies π' and π, define*

- $$\delta_f(x,a,x') = g(x) + \beta f(x') - f(x) \tag{3.5}$$

- *its max-bound:*

$$\epsilon_f^\pi = \max_{x \in X} \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|x) \\ x' \sim P(\cdot|x,a)}} [\delta_f(x, a, x')] \tag{3.6}$$

- *for transition matrix $P_{\pi'}$:*

$$a_{\pi'} = \kappa_2(S_{\pi'}) \frac{1-\beta}{1-\beta|\lambda_2(P_{\pi'})|} ||D_{\pi'}||_1, \tag{3.7}$$

where

  * $\kappa_2(S_{\pi'}) = ||S_{\pi'}||_2 ||S_{\pi'}^{-1}||_2$ *is a spectral conditional number of the matrix* $S_{\pi'} = (s_1, s_2, .., s_n)$ *where* s_i *are right eigenvectors of* $P_{\pi'}$*;*
  * $\lambda_2(P_{\pi'})$ *is a second largest eigenvalue of* $P_{\pi'}$*;*
  * $\Omega_{\pi'}$ *is a matrix with equal rows* $\mu_{\pi'}$*;*
  * $D = (I - P_{\pi'} + \Omega_{\pi'})^{-1} - \Omega$ *is a deviation matrix of* $P_{\pi'}$*,* $||D||_1$ *is known to be finite if* $P_{\pi'}$ *is uniformly ergodic [18];*

- $$L_{\pi,f}(\pi') = \mathop{\mathbb{E}}_{\substack{x \sim \mu_\pi \\ a \sim \pi(\cdot|x) \\ x' \sim P(\cdot|x,a)}} \left[\left(\frac{\pi'(a|x)}{\pi(a|x)} - 1 \right) \delta_f(x, a, x') \right] \tag{3.8}$$

- $$D_{\pi,f}^{\pm}(\pi') = L_{\pi,f}(\pi') \pm 2 \left(\frac{\beta}{1-\beta} \epsilon_f^{\pi'} + ||g||_2 a_{\pi'} \right) \mathbb{E}_{x \sim \mu_\pi}[D_{TV}(\pi'||\pi)[x]], \tag{3.9}$$

  where $\mathbb{E}_{x \sim \mu_\pi} D_{TV}(\pi'||\pi)[x] = \frac{1}{2} \mathbb{E}_{x \sim \mu_\pi} ||\pi(\cdot|x) - \pi'(\cdot|x)||_1 = \frac{1}{2} \sum\limits_{x \in X} \mu_\pi(x) \sum\limits_{a \in A} |\pi(a|x) - \pi'(a|x)|$.

The following bound holds for long-run average cost:

$$D_{\pi,f}^-(\pi') \geq \eta_{\pi'} - \eta_\pi \geq D_{\pi,f}^+(\pi') \tag{3.10}$$

In particular, if we take $f(x) = V_\pi^\beta(x)$ *we get* $\delta_V^\beta(x, a, x') = A_\pi^\beta(x, a)$ *and*

$$\eta_{\pi'} - \eta_\pi \leq \mathop{\mathbb{E}}_{\substack{x \sim \mu_\pi \\ a \sim \pi(\cdot|x)}} \left[\left(\frac{\pi'(a|x)}{\pi(a|x)} - 1 \right) A_\pi^\beta(x, a) \right] + 2 \left(\frac{\beta}{1-\beta} \epsilon_V^{\pi'} + ||g||_2 a_{\pi'} \right) \mathbb{E}_{x \sim \mu_\pi}[D_{TV}(\pi'||\pi)[x]] \tag{3.11}$$

Now we need to interpret the result of Theorem 1. Let define the coefficient near D_{TV} as

$$C_{\pi',\beta} = 2\frac{\beta}{1-\beta}\epsilon_V^{\pi'} + ||g||_2 a_{\pi'} \tag{3.12}$$

Then if follows from Equation 3.11 that we are able to generate a monotonically improving sequence of policies

$$\eta_{\pi_0} \geq \eta_{\pi_1} \geq ...$$

To see this, let

$$M_i(\pi_i) = \eta_{\pi_i} + \mathop{\mathbb{E}}_{\substack{x\sim\mu_{\pi_i} \\ a\sim\pi_i(\cdot|x)}} \left[\left(\frac{\pi_{i+1}(a|x)}{\pi_i(a|x)} - 1\right) A^{\beta}_{\pi_i}(x,a)\right] + C_{\pi_{i+1},\beta}[D_{TV}(\pi_{i+1}||\pi_i)[x]] \tag{3.13}$$

Then by Equation 3.11

$$\eta_{\pi_{i+1}} \leq M_i(\pi_{i+1})$$

and

$$\eta_{\pi_i} = M_i(\pi_i)$$

Subtracting we get

$$\eta_{\pi_{i+1}} - \eta_{\pi_i} \leq M_i(\pi_{i+1}) - M_i(\pi_i) \tag{3.14}$$

Thus, by minimizing M_i at each iteration, we guarantee that the objective η is non-increasing.

3.2 Practical algorithm

We consider parameterized policies $\pi_\theta(a|x)$ with parameter vector θ. In the previous section we showed that minimizing RHS of equation (3.11) one can yield a monotonic improvement of policy performance. In practice, we can simulate only finite trajectories and the penalty coefficient $C_{\pi',\beta}$ can be hard to evaluate. Following the idea of trust region optimization we use the following heuristic: we minimize (3.8) w.r.t. parameters θ

$$L_{\pi,V^{\beta}_{\pi}}(\pi'_\theta) = \mathop{\mathbb{E}}_{\substack{x\sim\mu_{\pi} \\ a\sim\pi(\cdot|x)}} \left[\left(\frac{\pi'_\theta(a|x)}{\pi(a|x)} - 1\right) A^{\beta}_{\pi}(x,a)\right] \tag{3.15}$$

constraining a divergence between the new policy π'_θ and the old policy π.

We note that the gradient $\nabla_\theta L_{\pi,V^{\beta}_{\pi}}(\pi'_\theta)$ is a biased gradient estimate of $\nabla_\theta \eta_{\pi_\theta}$ that has been analysed in [5]. It has been proved that the bias converges to 0 as β approaches 1.

We use Proximal Policy Optimization [37] that add clipping into surrogate objective (3.15):

$$L_{\pi,V_\pi^\beta}(\pi'_\theta) = \mathop{\mathbb{E}}_{\substack{x\sim\mu_\pi \\ a\sim\pi(\cdot|x)}} \left[\max \left\{ \frac{\pi'_\theta(a|x)}{\pi(a|x)} A_\pi^\beta(x,a), clip\left(\frac{\pi'_\theta(a|x)}{\pi(a|x)}, 1-\epsilon, 1+\epsilon \right) A_\pi^\beta(x,a) \right\} \right], \tag{3.16}$$

where ϵ is a hyperparameter.

In the second term of (3.16), namely $clip\left(\frac{\pi'_\theta(a|x)}{\pi(a|x)}, 1-\epsilon, 1+\epsilon\right) A_\pi^\beta(x,a)$, the clipping of the probability ratio eliminates the advantage of moving the changing ratio $\frac{\pi'_\theta(a|x)}{\pi(a|x)}$ outside of the interval $[1-\epsilon, 1+\epsilon]$.

The maximum of the clipped and unclipped objective is taken and, as a result, the final objective is a upper bound on the unclipped objective.

Each iteration the algorithm approximates the value function and then use it in generalized advantage estimation (GAE) [36] to reduce the variance in advantage function estimates, namely for $\lambda \in (0,1)$

$$\hat{A}_t^\pi = \delta_t + (\beta\lambda)\delta_{t+1} + (\beta\lambda)^2\delta_{t+2} + .. + (\beta\lambda)^{T-t+1}\delta_{T-1}, \tag{3.17}$$

where $\delta_t = g_t + \beta V_\pi(x_{t+1}) - V_\pi(x_t)$.

Each iteration N (parallel) actors simulate the current policy for T timesteps. After finishing simulation the algorithm estimates the surrogate loss based on collected $N \times T$ data points. Then the algorithm optimizes the surrogate obejctive using Adam algorithm [19], for E epochs. We use fully-connected deep neural networks to represent policy and value function. We use π_θ to specify policy parameters θ of the neural network. Similarly we define V_ω as a value function that is also approximated by a neural network with parameters ω. We do not share parameters between the policy and value function and do

not use any entropy bonus or other regularization term.

Algorithm 1: PPO, simulation routine

Result: policy π_{θ_I}

Initialize policy π_{θ_0};

for *policy iteration* $= 1, 2, ..., I$ **do**

  for *actor* $= 1, 2, ..., N$ **do**

    Define initial states of episodes;

    Run policy π_{i-1} for T time-steps;

    Compute advantage functions $\hat{A}_t^{\pi}$, $t = 1, .., T$.

  end

  Optimize surrogate objective function w.r.t. θ with E epochs:

$$L(\theta, D^i_{0:NT}) = \frac{1}{NT} \sum_{t=0}^{NT} \max \left[\frac{\pi_\theta(a_t|x_t)}{\pi_{\theta_i}(a_t|x_t)} \hat{A}_t^{\pi_{\theta_i}}, clip(\frac{\pi_\theta(a_t|x_t)}{\pi_{\theta_i}(a_t|x_t)}, 1-\epsilon, 1+\epsilon) \hat{A}_t^{\pi_{\theta_i}} \right]$$

  Update $\theta_{i+1} := \theta$

end

3.3 Choosing initial states for simulation

In the Algorithm 1 we use finite episodes to estimate the expectation of the advantage function over the stationary distribution of the current policy. To run the simulations one needs to specify how to choose initial states for the episodes.

We propose sampling the initiate states from the set of states visited during preceding policy iteration, see Algorithm 2. Consider the ith policy iteration of the algorithm. We need to choose initial states to simulate policy π_i. Policy π_{i-1} has been simulated in the $(i-1)$th iteration of the algorithm and a subset of states from the simulated trajectories can be saved in memory. In ith iteration initial states are sampled uniformly at random from this subset. For policy π_0 an episode starts from state $x = (0, .., 0)$.

Since the policy updates are restricted two policies π_i and π_{i-1} should be close. The state-visitation frequencies obtained simulating policy π_{i-1} should be close to the stationary distribution of policy π_i. The disadvantage of this method is that the policy π_{i-1} simulated in the preceding iteration can be unstable and does not have a stationary distribution. In our experiments we start from a random policy π_0 that often is unstable. To overcome this problem we introduce a limit Λ for an initial state x on the total number of jobs. That is, if a sampled state x is s.t. $\sum_{j=1}^{J} x_j > \Lambda$ the corresponding episode starts from state $x = (0, ..., 0)$.

Algorithm 2: Episodes are initialized from the states sampled from simulations conducted in the preceding policy iteration

Result: policy π_I

Initialization: policy π_0, buffer $\mathbb{B} = \emptyset$;
Run policy π_0 for T timesteps;
Sample uniformly at random $100 \times N$ states from the simulation of policy π_0;
Add $100 \times N$ states to $\mathbb{B}$;
for *policy iteration* $= 1, 2, ..., I$ **do**
    Sample uniformly at random N states $\{x^1, x^2, ..., x^N\}$ from $\mathbb{B}$;
    for $s = 1, 2, ..., N$ **do**
        if $\sum_{j=1}^{J} x_j^s > \Lambda$ **then**
            $x^s = (0, ..., 0)$;
        end
    end
    for *actor* $s = 1, 2, ..., N$ **do**
        Initialize episode at state x^s;
    end
    Sample uniformly at random $100 \times N$ states from policy π_i simulation;
    Overwrite buffer $\mathbb{B}$ with the sampled states;
end

4 Experimental results

In this section we test the policy π_I obtained after I policy iterations of our algorithm. We call this policy a *RL policy*.

To estimate the long-run average performance of the RL policy we average number of jobs in the network over $\left\lfloor \nu \Big(\sum_{j=1}^{J} \lambda_j \Big)^{-1} \right\rfloor \times 1,000,000$ decision epochs starting from an empty state $x = (0, .., 0)$, where $\lfloor y \rfloor$ is an integer part of y.

4.1 Criss-cross network

We study the algorithm performance for the system with various load intensity regimes, including I.L. (imbalanced light), B.L. (balanced light), I.M. (imbalanced medium), B.M. (balanced medium), I.H. (imbalanced heavy), and B.H. (balanced heavy). The corresponding arrival and service rates are presented in Table 1.

For the criss-cross network the optimal stationary policy can be found fairy accurately using truncation and relative value iteration. Each decision epoch we can check whether

an action recommended by the RL policy is optimal or not. Therefore, we can use the percentage of optimal decisions as an additional performance measure for the criss-cross network. The corresponding learning curves are provided in Figure 1. We note that if either buffer 1 or buffer 2 is empty the transition probabilities do not depend on action in (2.2). In this situation we classify both actions as optimal.

As benchmarks we provide performance of policies that have been proposed in the literature in Table 2. In the first column, we indicate the load regime. In the second column, we report the optimal performance obtained via dynamic programming, denoted by DP. In the third column, we report the performance of a target-pursuing policy proposed in [31] and denoted by OTP in the table. In the fourth column, we list the performance of a threshold policy proposed in [16]. In the fifth and sixth columns, we list the performance of fluid (FP) and robust fluid (RFP) policies from [8].

In the last column of Table 2 we provide simulation results of RL policies. The RL algorithm has run for 200 iterations. The parameters used in the experiments are provided in Appendix 6.3.

Load regime	λ_1	λ_2	μ_1	μ_2	μ_3	ρ_1	ρ_2
I.L.	0.3	0.3	2	2	1.5	0.3	0.2
B.L.	0.3	0.3	2	2	1	0.3	0.3
I.M.	0.6	0.6	2	2	1.5	0.6	0.4
B.M.	0.6	0.6	2	2	1	0.6	0.6
I.H.	0.9	0.9	2	2	1.5	0.9	0.6
B.H.	0.9	0.9	2	2	1	0.9	0.9

Table 1: Load parameters for the criss-cross network of Figure 1

Load regime	DP (optimal)	TP	threshold	FP	RFP	RL
I.L.	0.671	0.678	0.679	0.678	0.677	0.673
B.L.	0.843	0.856	0.857	0.857	0.855	0.847
I.M.	2.084	2.117	2.129	2.162	2.133	2.092
B.M.	2.829	2.895	2.895	2.965	2.920	2.855
I.H.	9.970	10.13	10.15	10.398	10.096	9.987
B.H.	15.228	15.5	15.5	18.430	15.585	15.361

Table 2: Simulation results for the criss-cross network under different polices.

4.2 Extended six-class queuing network

In this experiment we consider the family of extended six-class networks from [8]. The structure of the networks is shown in Figure 2.

Job classes 1 and 3 arrive externally to the network according to a Poisson process with a rate λ_1 and λ_2, respectively. Then jobs from class 1 and class 3 follow two separate routes. Jobs from class 3 are sequentially processed in each of L servers and then leave the network. Jobs from class 1 after being processed in each server fed back into 2nd buffer associated with the first server. After the jobs of class 2 are served in each server again and then leave the system. The experiments have been conducted for the network with the following traffic parameters: $\lambda_1 = \lambda_2 = 9/140$, the service times are exponentially distributed with service rates determined by the modulus after division the class index by 6. That is, classes associated with server 1 are served with rates $\mu_1 = 1/8$, $\mu_2 = 1/2$, $\mu_3 = 1/4$ and classes associated with server 2 are processed with service rates $\mu_4 = 1/6$, $\mu_5 = 1/7$, $\mu_6 = 1$. The service rates for the odd servers $S_1, ..., S_{\lfloor L/2 \rfloor + 1}$ are the same as the service rates for server 1, while the service rates for the even servers $S_2, ..., S_{\lfloor L/2 \rfloor}$ are the same as the service rates for server 2.

Table 3 provides the performance of the RL policy and compares it with other heuristic methods for the extended six-class queuing networks. In the experiments we change the size of the network to test the robustness of the RL policies. The size of the networks varies from 6 to 21 classes. In the table FP and RFP refer to fluid policy and robust fluid policy correspondingly. These policies are proposed in [8]. LBFS defines the last-buffer first-serve policy, where a priority at a server is given to jobs with highest index. FCFS refers to the first-come first-serve policy, where a priority at a server is given to jobs with the longest waiting time for service.

In our experiments we have noticed that action selection that involves execution a neural network and sampling from the output distribution is computationally intensive for large queuing networks. To reduce simulation time we repeated action in 4 consecutive timesteps before a new action is selected. This is similar to the "frame skipping" conducted for Atari games in [26]. We apply this technique to simulate policies in the extended six-class networks and the reentrant networks in the next section.

Learning curves showing the average number of jobs for each network are shown in Figure 8.

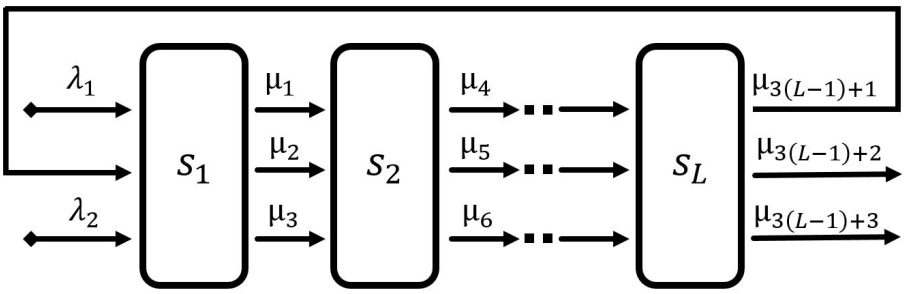


Figure 2: The extended six-class network

Num. of classes $3L$	LBFS	FCFS	FP	RFP	RL
6	15.749	40.173	15.422	15.286	14.180
9	25.257	71.518	26.140	24.917	23.166
12	34.660	114.860	38.085	36.857	32.420
15	45.110	157.556	45.962	43.628	40.508
18	55.724	203.418	56.857	52.980	50.034
21	65.980	251.657	64.713	59.051	56.947

Table 3: Numerical results for the extended six-class network of Figure 2.

4.3 Extended re-entrant queuing network

Another network that has been proposed in [8] is a reentrant line with L servers. Figure 3 displays the topology of this queueing network. Only buffer 1 is fed by external arrivals according to a Poisson process with a rate λ_1. The network consists of L services. To leave the system each job has to re-enter each server three times, thus each server processes 3 classes of jobs. The arrival rate λ_1 and all service rates of the job classes are the same as for the extended six-class network considered in Section 4.2.

Learning curves showing the average number of jobs for each network are shown in Figure 9.

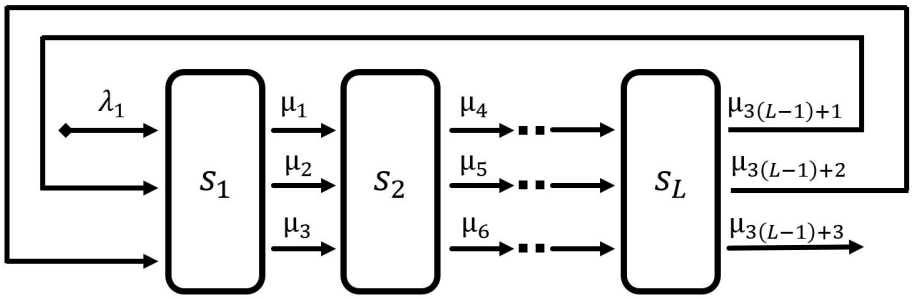


Figure 3: The extended re-entrant network

Num. of classes $3L$	LBFS	FCFS	FP	RFP	RL
6	15.911	16.441	15.422	15.663	14.317
9	30.001	27.154	25.955	24.015	22.359
12	40.167	35.633	32.014	29.925	28.856
15	57.056	47.969	40.113	36.901	36.003
18	66.042	56.996	48.781	44.261	46.670
21	87.136	71.002	54.711	48.418	51.512

Table 4: Numerical results for the reentrant network of Figure 3.

5 Conclusion

In this study we have presented application of deep reinforcement learning to the scheduling problem in multiclass queuing networks.

We have proposed a theoretically-justified modification of the Proximal Policy Optimization algorithm that can be applied in the long-run average setting. The algorithm does not require any knowledge of topological routing structure within the network and any information about traffic intensity. Admittedly, the learning process of the RL algorithm requires intensive simulations of many different policies that may require a generative model. The proposed approach scales well and can handle networks with multiple job classes and servers if advanced enough computational infrastructure is available.

Our numerical results show that RL policies yield almost-optimal performance for the criss-cross network and accomplish long-run average performance within 1% from the optimal according to Table 2. In large networks our approach leads to effective scheduling policies that outperform or perform comparable to known alternatives. In the extended six-class queuing network RL polices outperform the robust fluid policies on average by more than 5%. In the extended reentrant queuing network RL polices reach comparable performance with the robust fluid policies outperforming them on the 6-, 9-, 12-, 15- classes networks and performing within 5% of RFP on 18-, 21-classes networks. Note that displayed performance of robust fluid policy in Table 4 is the performance of the best robust fluid policy which corresponds to the best choice of policy parameters that can be different for each network. In all our experiments we have used fixed set of hyperparameters. Nevertheless, the results for the extended re-entrant line indicate that optimization of scheduling policies for large MQNs is a challenging problem for RL methods that use discounting to approximate the policy gradient. Designing of a stable deep RL algorithm that does not depend on the discount factor and directly optimizes long-run average performance is an open problem.

Based on our simulation study we provide a recommended set of hyperparameters for the algorithm in Section 6.4 and neural network structures for policy parametrization and approximation of the value function in Section 6.3. The recommended parameters of the algorithm yield stable learning process for large range of networks from the criss-cross network to the re-entrant network with 21 job classes.

Complexity of the scheduling optimization problem highly depends not only on the network topology, but on the traffic intensity. For the small criss-cross network in low traffic regime the RL policy almost coincides with the optimal policy: up to 99% of actions suggested by RL policy are optimal. While only around 85% of actions are optimal in the heavy traffic regime, see Figure 7. Therefore, the complexity of the problem can be easily adjusted. We believe that this feature makes multiclass queuing networks potentially good benchmark domain to test RL methods, for example cross-task generalization (transfer learning) [29].

In the paper we impose classical assumptions that external arrivals are Poisson and

service times are assumed to be exponentially distributed with class-dependent rates. More realistic scheduling policy optimization problem is to design a learning algorithm for a time-dependent distribution of inter-arrival times. We hope that the current paper a small step in this direction.

6 Appendix

6.1 Maximal Stability of Random Proportional Policy

In this section we consider random proportional policy.

Associated with each decision is a random proportional vector for $\gamma = (\gamma_1, ..., \gamma_J) \geq 0$ and satisfies $\sum_{i \in B_l} \gamma_i = 1$ and $\max_{i \in B_l} \gamma_i = 1$ for each server l.

For a job class $i \in B_l$ the fraction of server k the probability to get full service capacity is proportional to population size at the decision epoch.

$$\mathbb{P}(\gamma_i = 1) = \frac{x_i}{\sum\limits_{j \in B_l}}$$

Non-negative $J-$vector β of class-level service rates going forward from a decision time t,

$$\beta_i(t) = \begin{cases} \gamma_i \text{ if } z_i(t) > 0, \\ 0, \text{ otherwise} \end{cases} \tag{6.1}$$

Define a non-decreaseing $J-$dimentional process $T = \{T(t),\ t \geq 0\}$ whose jth component $T_j(t)$ represents the cumulative amount of service effort devoted to jth class over the time interval $(0, t]$. It can be expressed mathematically:

$$T_j(t) := \int_0^t \beta_i(u)du, \quad \text{for each class } j \text{ and } t \geq 0. \tag{6.2}$$

In the fluid model, for all $t \geq 0$ we have

$$\dot{Z}(t) = \lambda - (I - P')M\beta(t) \tag{6.3}$$

$$C\beta(t) \leq e \tag{6.4}$$

$$Z(t) \geq 0 \tag{6.5}$$

$$\beta(t) \geq 0 \tag{6.6}$$

The functions $Z_i(t)$ and $T_i(t)$ are absolutely continuous, and thus, differentiable almost everywhere. The equations 6.3 hold for all times t at which $Z_i(t)$ and $T_i(t)$ are differentiable.

For a job class $i \in B_l$ the fraction of server define "an expected" service allocation given queue lenghts at time t $\hat{\beta}_i(t) := \begin{cases} \frac{z_i(t)}{\sum_{j \in B_l} z_j(t)} \end{cases}$.

For each initial state $z \in \mathbb{Z}_+^L$ and $t > 0$,

$$\xi_i^z(t) := \int_0^t (\beta_i^z(u) - \hat{\beta}_i^z(u))du \tag{6.7}$$

Let us explicitely write $Z^z(t, \omega)$ for a particular sample path ω of $Z^z(t)$.

Lemma 1. *Let $\{z^r : r \geq 1\} \subset \mathbb{Z}_+^r$ be a sequence of initial states satisfying $|z^r| \geq r^2$ for each $r \geq 1$, then with probability 1:*

$$\lim_{r \to \infty} \frac{\xi^{z^r}(|z^r, \omega|)}{|z^r|} = 0 \tag{6.8}$$

Proof. By Chebyshev's inequality:

$$\mathbb{P}\left\{\omega : \frac{|\xi_i^{z^r}(|z^r, \omega|)|}{|z^r|} > \epsilon\right\} \leq \frac{\mathbb{E}\left[\xi_i^{z^r}(|z^r|)\right]^2}{\epsilon^2 |z^r|^2} \tag{6.9}$$

Let $\{\tau_k\}_{k=0}^K$ are the decision epochs before time and $\tau_{K+1} = |z^r|$, where $\tau_0 = 0$ then

$$\mathbb{E}\left[\xi_i^{z^r}(|z^r|)\right]^2 = \mathbb{E}\left[\int_0^{|z^r|} (\beta(u) - \hat{\beta}(u))du\right]^2 = \mathbb{E}\left[\sum_{k=0}^K \int_{\tau_k}^{\tau_{k+1}} (\beta(u) - \hat{\beta}(u))du\right]^2 = \tag{6.10}$$

$$\mathbb{E}\sum_{k=0}^K \left(\int_{\tau_k}^{\tau_{k+1}} (\beta(u) - \hat{\beta}(u))du\right)^2 + 2\sum_{k<k'}^K \mathbb{E}\left(\int_{\tau_k}^{\tau_{k+1}} (\beta(u) - \hat{\beta}(u))du\right)\left(\int_{\tau_k'}^{\tau_{k'+1}} (\beta(u) - \hat{\beta}(u))du\right) = \tag{6.11}$$

$$\mathbb{E}\sum_{k=0}^K \left(\int_{\tau_k}^{\tau_{k+1}} (\beta(u) - \hat{\beta}(u))du\right)^2 \leq \mathbb{E}\sum_{k=0}^K \left(\int_{\tau_k}^{\tau_{k+1}} 1du\right)^2 \leq \mathbb{E}\sum_{\tau' \in \text{arrivals}} (\tau'_{k+1} - \tau'_k)^2 \leq O(|z^r|) \tag{6.12}$$

where the second inequality follows from the fact that

$$\mathbb{E}\sum_{k<k'}^{K}\mathbb{E}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)\left(\int_{\tau_k'}^{\tau_{k'+1}}(\beta(u)-\hat{\beta}(u))du\right)= \tag{6.13}$$

$$\mathbb{E}\left[\sum_{k<k'}^{K}\mathbb{E}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)\mathbb{E}\left[\left(\int_{\tau_k'}^{\tau_{k'+1}}(\beta(u)-\hat{\beta}(u))du\right)|Z(\tau_k')\right]\right]=0 \tag{6.14}$$

Hence $\mathbb{P}\left\{\omega:\frac{|\xi_i^{z^r}(|z^r,\omega|)|}{|z^r|}>\epsilon\right\}\leq\frac{1}{\epsilon^2 r^2}$ the lemma follows from the Borel-Cantelli lemma.

□

Theorem 2. *Consider a multiclass queueing network operating under the random proportional policy. Let $\{z^r : r\geq 1\}\subset\mathbb{Z}_+^r$ be a sequence of initial states satisfying $|z^r|\geq r^2$ for each $r\geq 1$, and let Ω be as in Lemma **??**. For each $\omega\in\Omega_1\sup\Omega_2$, all fluid limits $(\hat{D},\hat{T},\hat{Z})$ satisfying the following: for each packet class $i\in I$*

$$\frac{d}{dt}\hat{T}_i(t)=\frac{\hat{Z}_i(t)}{\sum_{j\in B_l}\hat{Z}_j(t)}\ \textit{when}\ \sum_{j\in B_l}\hat{Z}_j(t)>0. \tag{6.15}$$

Proof. Assume that $\{z^r : r\geq 1\}\subset\mathbb{Z}_+^r$ is a sequence of initial states satisfying $|z^r|\geq r^2$. Fix an $\omega\in\Omega_1\sup\Omega_2$. Let $(\hat{D}(\cdot),\hat{T}(\cdot),\hat{Z}(\cdot))$ be a fluid limit path. Fix a buffer i and $t>0$, and let $l\in L$ be an assotiated server to buffer i. Assume that $\hat{Z}_i(t)>0$. By the continuity of $\hat{Z}_i(\cdot)$, there exists a $\delta\in(0,t)$ s.t.

$$\epsilon:=\min_{u\in[t-\delta,t+\delta]}\hat{Z}_i(u)>0. \tag{6.16}$$

Thus, there exists $Z_i^{z_n}(u)\geq|z_n|\epsilon/2\geq 1$ for $u\in(|z_n|(t-\delta),|z_n|(t+\delta))$ and $n\geq L_0$. Next,

$$T_i^{z_r}(|z_r|u_2)-T_i^{z_r}(|z_r|u_1)=\int_{|z_r|u_1}^{|z_r|u_2}(\beta_i^z(u)du=\int_{|z_r|u_1}^{|z_r|u_2}\hat{\beta}_i^z(u)du+\xi_i^{z_r}(|z_r|u_2)-\xi_i^{z_r}(|z_r|u_1). \tag{6.17}$$

We have proved that $\lim_{r\to\infty}\frac{1}{||z^r||}\left[\xi_i^{z_r}(|z_r|u_2)-\xi_i^{z_r}(|z_r|u_1)\right]=0$.

Thus

$$\hat{T}_i^{z_n}(u_2) - \hat{T}_i^{z_n}(u_1) = \int_{u_1}^{u_2} \hat{\beta}(\hat{Z}^{z_r}(u))du = \int_{u_1}^{u_2} \tag{6.18}$$

□

6.2 Proofs

Definition 3. We denote the discounted future state distribution starting at state s by

$$d_\pi^s(x) = (1-\beta)\sum_{t=0}^{\infty} \beta^t P_\pi(x_t = x | x_0 = s), \tag{6.19}$$

where $P_\pi(x_t = x|x_0 = s) = P_\pi^t(x_1 = x|x_0 = s)$ is a probability od visiting state x at time t if we start at state s and follow policy π.

Theorem 3.

$$\eta_\pi = (1-\beta)\mathbb{E}_{x\sim\mu_\pi} V_\pi^\beta(x) \tag{6.20}$$

Proof. The proof is given in [39, Section 10.4]. □

Theorem 4. *For any policies π and π' we have*

$$\eta_{\pi'} - \eta_\pi \leq (1-\beta)\mathbb{E}_{x\sim\mu_\pi}\Big[V_{\pi'}^\beta(x) - V_\pi^\beta(x)\Big] + 2\kappa_2(S_{\pi'})||g||_2 \frac{1-\beta}{1-\beta|\lambda_2(P_{\pi'})|}||D||_1 \mathbb{E}_{x\sim\mu_\pi} D_{TV}(\pi'||\pi)[x], \tag{6.21}$$

where $\kappa_2(S_{\pi'}) = ||S_{\pi'}||_2||S_{\pi'}^{-1}||_2$ is a spectral conditional number of the matrix $S_{\pi'} = (s_1, s_2, .., s_n)$ where s_i are right eigenvectors of $P_{\pi'}$;

$\lambda_2(P_{\pi'})$ is a second largest eigenvalue of $P_{\pi'}$;

$\Omega_{\pi'}$ is a matrix with equal rows $\mu_{\pi'}$;

$D = (I - P_{\pi'} + \Omega_{\pi'})^{-1} - \Omega$ is a deviation matrix of $P_{\pi'}$, $||D||_1$ is known to be finite if $P_{\pi'}$ is uniformly ergodic [18];

$\mathbb{E}_{x\sim\mu_\pi} D_{TV}(\pi'||\pi)[x] = \frac{1}{2}\mathbb{E}_{x\sim\mu_\pi}||\pi(\cdot|x) - \pi'(\cdot|x)||_1 = \frac{1}{2}\sum\limits_{x\in X}\mu_\pi(x)\sum\limits_{a\in A}|\pi(a|x) - \pi'(a|x)|.$

Proof. From [5, Theorem 3.] we have the following equality:

$$(1-\beta)V_{\pi'}^\beta = \eta_{\pi'}e + S(\mu'_\pi)\text{diag}(0, \frac{1-\beta}{1-\beta|\lambda_2(P'_\pi)|}, ..., \frac{1-\beta}{1-\beta|\lambda_n(P'_\pi)|})S^{-1}(\mu'_\pi)g, \tag{6.22}$$

where $V^{\beta}_{\pi'}$ is a vector form of $[V^{\beta}_{\pi'}(x)]_{x\in X}$, $e=(1,1,...,1)^T$ $\lambda_1(P'_\pi)=1>|\lambda_2(P'_\pi)|\geq\lambda_n(P'_\pi)$ are eigenvalues of P'_π (we assume that P'_π has n district eigenvalues.)

Now we consider a difference between expectation of equation 6.22 taken over stationary distribution μ_π and expectation of this equation taken over $\mu_{\pi'}$.

$$(1-\beta)\mathbb{E}_{x\sim\mu_\pi}V_{\pi'}(x)=\eta'_\pi+(\mu_\pi-\mu'_\pi)^TS(\mu'_\pi)\text{diag}(0,\frac{1-\beta}{1-\beta|\lambda_2(\mu'_\pi)|},...,\frac{1-\beta}{1-\beta|\lambda_n(\mu'_\pi)|})S^{-1}(\mu'_\pi)g, \tag{6.23}$$

where we use the fact from Theorem 3 that $\mathbb{E}_{x\sim\mu_\pi}V_\pi(x)=\eta_\pi$.

The last term can be bounded using Cauchy-Schwarz inequality:

$$\Big|(\mu_\pi-\mu'_\pi)^TS(\mu'_\pi)\text{diag}(0,\frac{1-\beta}{1-\beta|\lambda_2(\mu'_\pi)|},...,\frac{1-\beta}{1-\beta|\lambda_n(\mu'_\pi)|})S^{-1}(\mu'_\pi)g\Big|\geq \tag{6.24}$$

$$-||\mu_\pi-\mu'_\pi||_2||S(\mu'_\pi)\Big|\Big|\text{diag}(0,\frac{1-\beta}{1-\beta|\lambda_2(\mu'_\pi)|},...,\frac{1-\beta}{1-\beta|\lambda_n(\mu'_\pi)|})\Big|\Big|_2||S^{-1}(\mu'_\pi)g||_2\geq \tag{6.25}$$

$$\kappa_2(S)||g||_2\frac{1-\beta}{1-\beta|\lambda_2(\mu'_\pi)|}||\mu_{\pi'}-\mu_\pi||_2, \tag{6.26}$$

where $||diag(d_1,d_2,..,d_n)||_2\leq\max_i d_i$ and $\kappa_2(S)=||S||_2||S^{-1}||_2$.

We can get a bound on the difference between stationary distributions :

$$||\mu_{\pi'}-\mu_\pi||_2\leq||\mu_{\pi'}-\mu_\pi||_1\leq||\mu_\pi(P_{\pi'}-P_\pi)||_1||D_{\pi'}||_1\leq 2\mathbb{E}_{x\sim\mu_{\pi'}}D_{TV}(\pi'||\pi)[x]||D_{\pi'}||_1, \tag{6.27}$$

where the second inequality follows from equality $\mu_\pi-\mu'_\pi=\mu_\pi(P_\pi-P_{\pi'})D$, see [23, Section 2] and the last inequality follows from [3, Lemma 3]. □

Theorem 5.

$$\mathop{\mathbb{E}}_{s\sim\mu_\pi}\left[\mathop{\mathbb{E}}_{\substack{x\sim d_\pi(s)\\a\sim\pi(\cdot|x)\\x'\sim P(\cdot|x,a)}}f(x,a,x')\right]=\mathop{\mathbb{E}}_{\substack{x\sim\mu_\pi\\a\sim\pi(\cdot|x)\\x'\sim P(\cdot|x,a)}}f(x,a,x') \tag{6.28}$$

Proof. Consider the same MDP problem but with cost-to-go function $f(x)=\mathop{\mathbb{E}}_{\substack{a\sim\pi(\cdot|x)\\x'\sim P(\cdot|x,a)}}f(x,a,x')$. Then $\frac{1}{1-\beta}E_{x\sim d_\pi(s)}f(x)$ is the infinite horizon discounted total costs at state s of policy π. The result follows from Theorem 3.

□

Theorem 6. *For any function $f:X\to\mathbb{R}$ and any policies π' and π, define*

- $$\delta_f(s,a,x')=g(x)+\beta f(x')-f(x) \tag{6.29}$$

- *its max-bound:*

$$\epsilon_f^\pi = \max_{x \in X} \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|x) \\ x' \sim P(\cdot|x,a)}} [\delta_f(x, a, x')] \tag{6.30}$$

- *For transition matrix $P_{\pi'}$ and discount factor β we define*

$$a_{\pi'} = \kappa_2(S_{\pi'}) \frac{1-\beta}{1-\beta|\lambda_2(P_{\pi'})|} ||D_{\pi'}||_1, \tag{6.31}$$

  where

  * $\kappa_2(S_{\pi'}) = ||S_{\pi'}||_2 ||S_{\pi'}^{-1}||_2$ *is a spectral conditional number of the matrix* $S_{\pi'} = (s_1, s_2, .., s_n)$ *where* s_i *are right eigenvectors of* $P_{\pi'}$*;*
  * $\lambda_2(P_{\pi'})$ *is a second largest eigenvalue of* $P_{\pi'}$*;*
  * $\Omega_{\pi'}$ *is a matrix with equal rows* $\mu_{\pi'}$*;*
  * $D = (I - P_{\pi'} + \Omega_{\pi'})^{-1} - \Omega$ *is a deviation matrix of* $P_{\pi'}$*,* $||D||_1$ *is known to be finite if* $P_{\pi'}$ *is uniformly ergodic [18];*

- $$L_{\pi,f}(\pi') = \mathop{\mathbb{E}}_{\substack{x \sim \mu_\pi \\ a \sim \pi(\cdot|x) \\ x' \sim P(\cdot|x,a)}} \left[\left(\frac{\pi'(a|x)}{\pi(a|x)} - 1 \right) \delta_f(x, a, x') \right] \tag{6.32}$$

- $D_{\pi,f}^{\pm}(\pi') = L_{\pi,f}(\pi') \pm 2 \left(\frac{\beta}{1-\beta} \epsilon_f^{\pi'} + ||g||_2 a_{\pi'} \right) \mathbb{E}_{x \sim \mu_\pi}[D_{TV}(\pi'||\pi)[x]]$,
  where $\mathbb{E}_{x \sim \mu_\pi} D_{TV}(\pi'||\pi)[x] = \frac{1}{2} \mathbb{E}_{x \sim \mu_\pi} ||\pi(\cdot|x) - \pi'(\cdot|x)||_1 = \frac{1}{2} \sum\limits_{x \in X} \mu_\pi(x) \sum\limits_{a \in A} |\pi(a|x) - \pi'(a|x)|$.

The following bound for long-run average cost holds:

$$D_{\pi,f}^-(\pi') \geq \eta_{\pi'} - \eta_\pi \geq D_{\pi,f}^+(\pi') \tag{6.33}$$

In particular, if we take $f(x) = V_\pi(x)$ *we get* $\delta_V(x, a, x') = A_\pi(x, a)$ *and*

$$\eta_{\pi'} - \eta_\pi \leq \mathop{\mathbb{E}}_{\substack{x \sim \mu_\pi \\ a \sim \pi(\cdot|x)}} \left[\left(\frac{\pi'(a|x)}{\pi(a|x)} - 1 \right) A_\pi(x, a) \right] + 2 \left(\frac{\beta}{1-\beta} \epsilon_V^{\pi'} + ||g||_2 a_{\pi'} \right) \mathbb{E}_{x \sim \mu_\pi}[D_{TV}(\pi'||\pi)[x]] \tag{6.34}$$

Proof. Consider inequality 6.38 from Lemma 2 and show one side of inequality 6.33. Starting from inequality 6.37 one can show opposite side of inequality 6.33.

Assuming that initial states s are sampled from a stationary distribution of policy π we get:

$$(1-\beta)\mathbb{E}_{s\sim\mu_\pi}\left[V_{\pi'}(s)-V_\pi(s)\right]\leq$$

$$\mathbb{E}_{s\sim\mu_\pi}\left[\mathop{\mathbb{E}}_{\substack{x\sim d_\pi^s\\ a\sim\pi(\cdot|x)\\ x'\sim P(\cdot|x,a)}}\left[\left(\frac{\pi'(a|x)}{\pi(a|x)}-1\right)\delta_f(x,a,x')\right]+\frac{2\beta}{1-\beta}||\delta^s_{f,\pi'}||_\infty\mathbb{E}_{x\sim d_\pi^s}[D_{TV}(\pi'||\pi)[x]]\right]$$

Then from Theorem 5 we have:

- $\mathbb{E}_{s\sim\mu_\pi}\left[\mathop{\mathbb{E}}_{\substack{x\sim d_\pi^s\\ a\sim\pi(\cdot|x)\\ x'\sim P(\cdot|x,a)}}\left[\left(\frac{\pi'(a|x)}{\pi(a|x)}-1\right)\delta_f(x,a,x')\right]\right]=\mathop{\mathbb{E}}_{\substack{x\sim\mu_\pi\\ a\sim\pi(\cdot|x)\\ x'\sim P(\cdot|x,a)}}\left[\left(\frac{\pi'(a|x)}{\pi(a|x)}-1\right)\delta_f(x,a,x')\right]$
- $\mathbb{E}_{s\sim\mu_\pi}\left[\frac{2\beta}{1-\beta}||\delta^s_{f,\pi'}||_\infty\mathbb{E}_{x\sim d_\pi^s}[D_{TV}(\pi'||\pi)[x]]\right]=\frac{2\beta}{1-\beta}||\delta_{f,\pi'}||_\infty\mathbb{E}_{x\sim\mu_\pi}[D_{TV}(\pi'||\pi)[x]]$

Hence

$$(1-\beta)\mathbb{E}_{s\sim\mu_\pi}\left[V_{\pi'}(s)-V_\pi(s)\right]\leq$$
$$\mathop{\mathbb{E}}_{\substack{x\sim\mu_\pi\\ a\sim\pi(\cdot|x)\\ x'\sim P(\cdot|x,a)}}\left[\left(\frac{\pi'(a|x)}{\pi(a|x)}-1\right)\delta_f(x,a,x')\right]+\frac{2\beta}{1-\beta}||\delta_{f,\pi'}||_\infty\mathbb{E}_{x\sim\mu_\pi}[D_{TV}(\pi'||\pi)[x]]$$

Combining the last equation with Theorem 4 we get

$$\eta_{\pi'}-\eta_\pi\leq(1-\beta)\mathbb{E}_{x\sim\mu_\pi}\left[V_{\pi'}(x)-V_\pi(x)\right]+2a_{\pi'}||g||_2\mathbb{E}_{x\sim\mu_\pi}D_{TV}(\pi'||\pi)[x]\leq \tag{6.35}$$

$$\mathop{\mathbb{E}}_{\substack{x\sim\mu_\pi\\ a\sim\pi(\cdot|x)\\ x'\sim P(\cdot|x,a)}}\left[\left(\frac{\pi'(a|x)}{\pi(a|x)}-1\right)\delta_f(x,a,x')\right]+\frac{2\beta}{1-\beta}||\delta_{f,\pi'}||_\infty\mathbb{E}_{x\sim\mu_\pi}[D_{TV}(\pi'||\pi)[x]]+2a_{\pi'}||g||_2\mathbb{E}_{x\sim\mu_\pi}D_{TV}(\pi'||\pi)[x] \tag{6.36}$$

□

Lemma 2.

$$V_{\pi'}(s)-V_\pi(s)\geq\frac{1}{1-\beta}\left(\mathop{\mathbb{E}}_{\substack{x\sim d_\pi^s\\ a\sim\pi(\cdot|x)\\ x'\sim P(\cdot|x,a)}}\left[\left(\frac{\pi'(a|x)}{\pi(a|x)}-1\right)\delta_f(x,a,x')\right]-\frac{2\beta}{1-\beta}||\delta^s_{f,\pi'}||_\infty\mathbb{E}_{x\sim d_\pi^s}[D_{TV}(\pi'||\pi)[x]]\right) \tag{6.37}$$

$$V_{\pi'}(s) - V_{\pi}(s) \leq \frac{1}{1-\beta} \left(\mathop{\mathbb{E}}_{\substack{x \sim d_{\pi}^{s} \\ a \sim \pi(\cdot|x) \\ x' \sim P(\cdot|x,a)}} \left[\left(\frac{\pi'(a|x)}{\pi(a|x)} - 1 \right) \delta_f(x, a, x') \right] + \frac{2\beta}{1-\beta} ||\delta_{f,\pi'}^{s}||_{\infty} \mathbb{E}_{x \sim d_{\pi}^{s}} [D_{TV}(\pi' || \pi)[x]] \right) \tag{6.38}$$

Proof. See Lemma 2 and Lemma 3 in [3]. Note that all expectations are taken w.r.t. the discounted future state distribution $d_{\pi}^{s}(\cdot)$ starting at state s. □

6.3 Neural Network structure

In the experiments we parameterized the RL policy with a neural network. The policy neural network $\pi_\theta(a|x)$ uses the system state x as an input and deterministically maps it to a distribution over action space. Then we can sample an action for state x according to the likelihood of each action.

To represent the policy we use a fully-connected multilayer perceptron (MLP) (except the output layer) with three hidden layers and tanh activation functions. The input layer has J units corresponding to each job class, the first hidden layer consists of $10 \times J$ units, the third layer has $10 \times L$, where L is number of services in the queueing system. Number of units in the second layer is a geometric mean of units in the first and third layers, that is $10 \times \sqrt{LJ}$.

For the last output layer we use a factored action space, where each factor corresponds to a server and is parameterized as a categorical distribution. We represent the output action of $\pi_\theta(a|x)$ as a tuple $(a_1, a_2, ..., a_L)$, where $a_l \in B_l$ represents an action (priority class) for lth server. Each of these components is assumed to have a categorical distribution which is specified by a probability vector $\varsigma_l = [p_j^l : j \in B_l]$, where p_j^l is a probability of giving priority to jth class in the server l. In the neural network each components of $\varsigma = [\varsigma_1, ..., \varsigma_L]$ computed applying the softmax operator to the third layer yielding normalized probabilities for each factor.

To represent the value function we use a fully-connected MLP with three hidden layers and tanh activation functions. The input layer is composed of J units corresponding to each job class, the first hidden layer formed of $10 \times J$ units, the third layer has 10 units. The number of units in the second layer is $10 \times \sqrt{J}$. The output layer contains one unit with a linear activation function.

6.4 Experiment Parameters

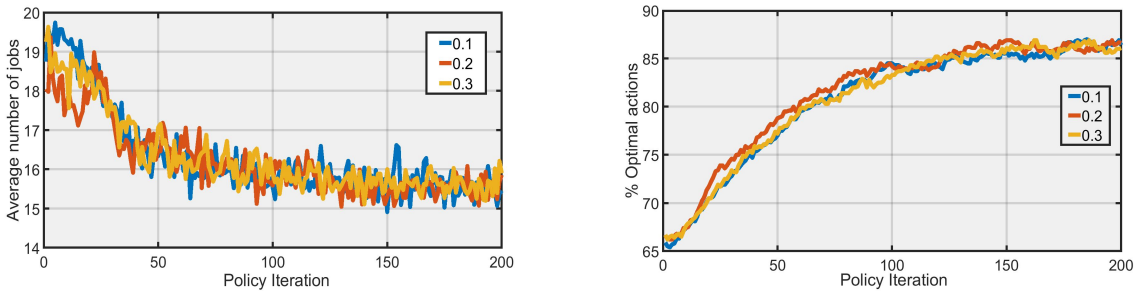
In this section we explore learning process of the RL algorithm under different hyper-paraments settings. We chose a computationally cheap benchmark, namely the criss-cross network from Figure 1 under the B.H. load condition. In each experiment we vary only

one or two parameters and the rest parameters are fixed to the values in the last column of Table 5. We note that we use since the

Parameter	Value
Clipping parameter (ϵ)	$0.2 \times \alpha$
Horizon (T)	20,000
Num. of episodes/actors (N)	50
Adam stepsize for policy NN	$5 \cdot 10^{-4} \times \max[\alpha, 0.1]$
Adam stepsize for value NN	$2.5 \cdot 10^{-4}$
Discount factor (β)	0.995
GAE parameter (λ)	0.97
Num. of epochs (K)	3
Minibatch size	256
Skipped decision epochs	0 for Table 2; 3 for the rest experiments
Routine to choose int. states	Algorithm 2
Bound for a sampled initial state (Λ)	300

Table 5: Values of hyperparameters used in the experiments. Parameter α is linearly annealed from 1 to 0 over the course of learning.

First we check dependency on clipping parameter. We have tested three options $\epsilon = \{0.1, 0.2, 0.3\}$ that had been proposed in [37]. We have not identified any significant affect of the clipping parameter on learning and use $\epsilon = 0.2$ for the rest experiments.

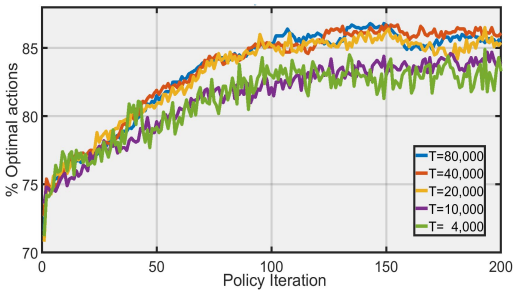


(a) average cost **(b)** Percentage of optimal actions

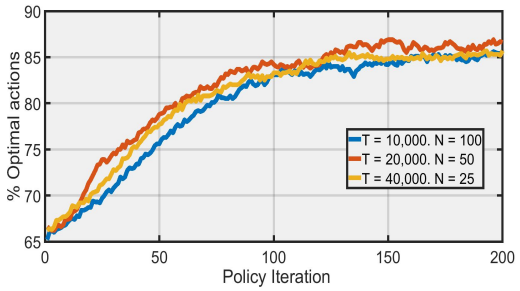
Figure 4: Varying the clipping parameter within $\epsilon = \{0.1, 0.2, 0.3\}$ does not have significant effect on learning.

Now we test dependency on number of episodes E and their duration T. In Figure 5a we fix number of episodes/actors to $N = 25$ and test how the learning process depends on an episode horizon T. We observe that increasing horizon more than $T = 40000$ does not yield further improvement. Next we fix total sample size to $T \times N = 10^9$ and examine the trade-off between number of episodes N and an episode duration T. In Figure

5b we show learning curves for three sets of parameters $\{(T = 10000, N = 100), (T = 20000, N = 50), (T = 40000, N = 25)\}$. We observe that $T = 20000$ and $N = 50$ is the best combination.



(a) Results for different values of T if number of actors is fixed $N = 25$.



(b) Trade-off between the number of actors N and horizon length T.

Figure 5: Effect of episode duration T and number of actors N on the learning process.

Another important parameter is discount factor β. We examine a discount factor jointly with GAE parameter λ. In our tests parameters $\beta = 0.995$ and $\lambda = 0.97$ yield the best result.

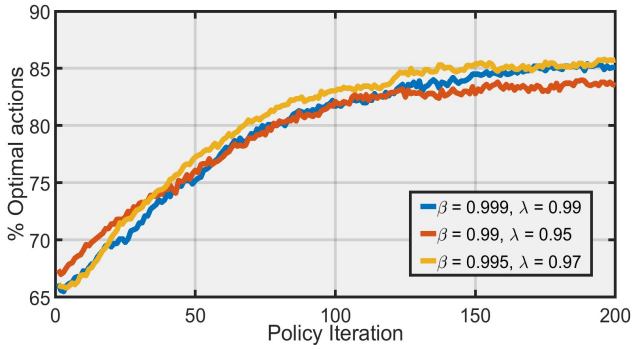
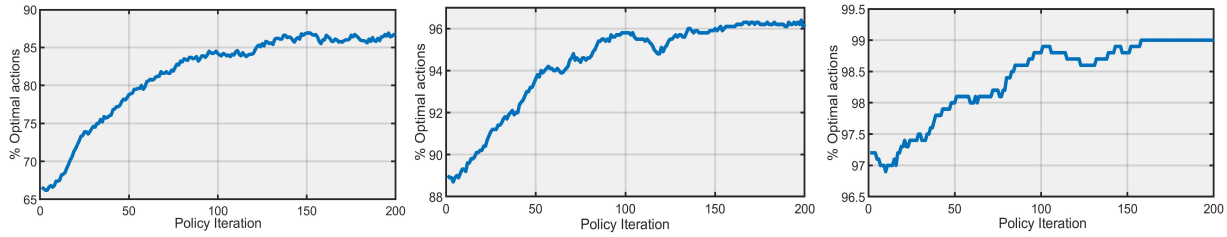
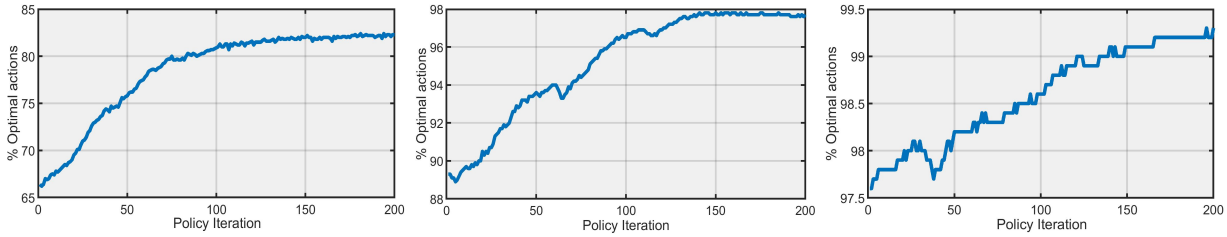


Figure 6: Effect of discount factor and GAE parameter.

6.5 Learning Curves

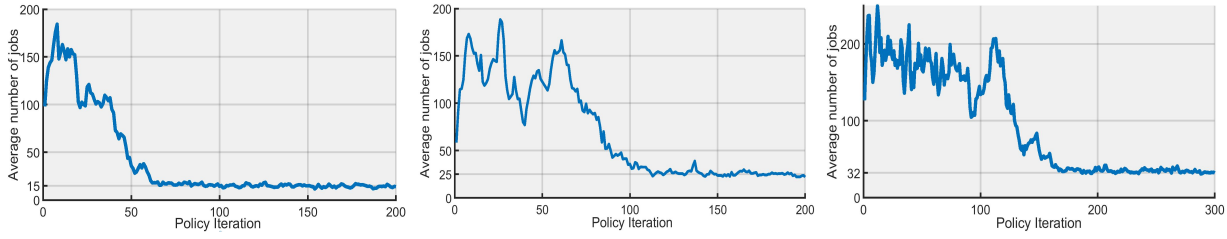


(a) Balanced heavy (BH) traffic **(b)** Balanced medium (BM) traffic **(c)** Balanced low (BL) traffic

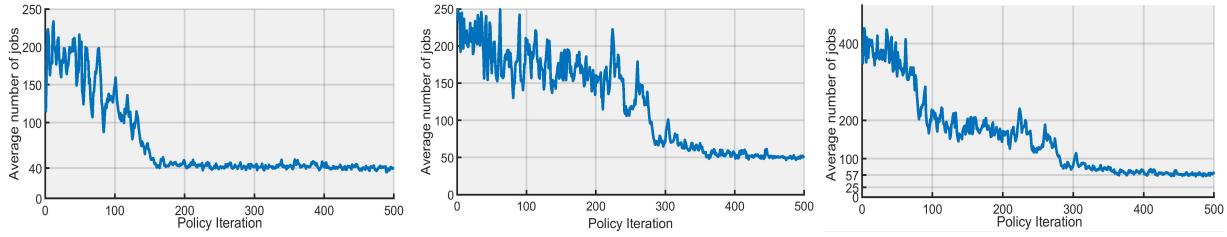


(d) Imbalanced heavy (IH) traffic **(e)** Imbalanced medium (IM) traffic **(f)** Imbalanced low (IL) traffic

Figure 7: Learning of optimal actions. Results for the criss-cross network of Figure 1 under different traffic regimes.

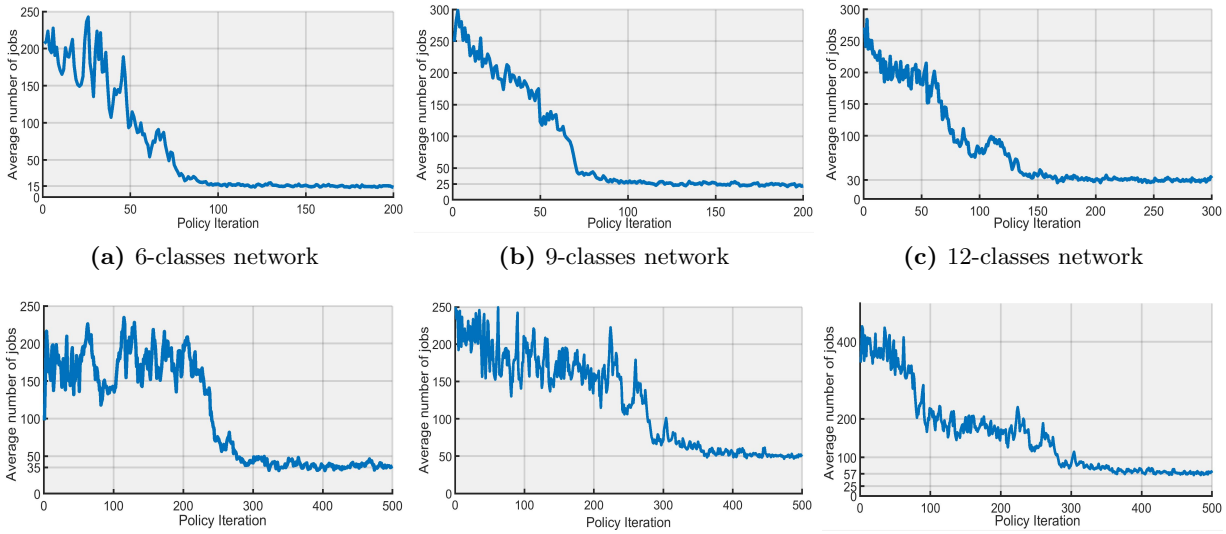


(a) 6-classes network **(b)** 9-classes network **(c)** 12-classes network



(d) 15-classes network **(e)** 18-classes network **(f)** 21-classes network

Figure 8: Learning curves for the 6-class extended network of Figure 2



(a) 6-classes network **(b)** 9-classes network **(c)** 12-classes network

(d) 15-classes network **(e)** 18-classes network **(f)** 21-classes network

Figure 9: Learning curves for the reentrant extended network of Figure 3

6.6 Implementation Details

We use Tensorflow v1.13.1 framework [1] to build a training routine of the neural networks and Ray package v0.6.3 [28] to maintain parallel simulation of actors. All experiments have been proceeded on a 2.1 GHz 32-core processor with 125 GB of RAM. We note that in most of our experiments we used $N = 50$ actors that approximately double simulation time.

Num. of classes $3L$	Time, minutes
6	0.75
9	0.78
12	1.07
15	1.54
18	2.11
21	2.61

Table 6: Running time of one policy iteration of the RL algorithm for the extended six-class network of Figure 2.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265—-283. USENIX Association, 2016.

[2] Yasin Abbasi-Yadkori, Peter Bartlett, and Alan Malek. Linear programming for large-scale Markov decision problems. In *Proceeding ICML'14 Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, pages 496–504, 2014.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. may 2017.

[4] Nicole Bauerle. Asymptotic optimality of tracking policies in stochastic networks. *The Annals of Applied Probability*, 10(4):1065–1083, nov 2001.

[5] Jonathan Baxter and Peter L Bartlett. Infinite-Horizon Policy-Gradient Estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[6] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47(1):253–279, 2013.

[7] Dimitris Bertsimas, J. Daniel Griffith, Vishal Gupta, Mykel J. Kochenderfer, and Velibor V. Mišić. A comparison of Monte Carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems. *European Journal of Operational Research*, 263(2):664–678, dec 2017.

[8] Dimitris Bertsimas, Ebrahim Nasrabadi, and Ioannis Ch. Paschalidis. Robust Fluid Processing Networks. *IEEE Transactions on Automatic Control*, 60(3):715–728, mar 2015.

[9] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. 2018.

[10] J. G. Dai. On Positive Harris Recurrence of Multiclass Queueing Networks: A Unified Approach Via Fluid Limit Models. *The Annals of Applied Probability*, 5(1):49–77, feb 1995.

[11] D. P. de Farias and B. Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 51(6):850–865, 2003.

[12] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pages 1329–1338. JMLR.org, 2016.

[13] J. Gao and R. Evans. DeepMind AI Reduces Google Data Centre Cooling Bill by 40%, 2016.

[14] J. Michael Harrison. Brownian Models of Queueing Networks with Heterogeneous Customer Populations. pages 147–186. Springer, New York, NY, 1988.

[15] J. Michael Harrison and Lawrence M. Wein. Scheduling networks of queues: Heavy traffic analysis of a simple open network. *Queueing Systems*, 5(4):265–279, dec 1989.

[16] J. Michael Harrison and Lawrence M. Wein. Scheduling Networks of Queues: Heavy Traffic Analysis of a Two-Station Closed Network. *Operations Research*, 38(6):1052–1064, 1990.

[17] Sham Kakade and John Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann Publishers, 2002.

[18] N. Kartashov. Criteria for uniform ergodicity and strong stability of Markov chains with a common phase space. *Theory Probab. Appl.*, 30:71–89, 1985.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2017.

[20] Matthieu Komorowski, Leo A. Celi, Omar Badawi, Anthony C. Gordon, and A. Aldo Faisal. The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11):1716–1720, nov 2018.

[21] C. N. Laws and G. M. Louth. Dynamic Scheduling of a Four-Station Queueing Network. *Probability in the Engineering and Informational Sciences*, 4(1):131–156, jan 1990.

[22] Steven A. Lippman. Applying a New Device in the Optimization of Exponential Queuing Systems. *Operations Research*, 23(4):687–710, aug 1975.

[23] Yuanyuan Liu. Perturbation Bounds for the Stationary Distributions of Markov Chains. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1057–1074, jan 2012.

[24] Constantinos Maglaras. Discrete-review policies for scheduling stochastic networks: trajectory tracking and fluid-scale asymptotic optimality. *The Annals of Applied Probability*, 10(3):897–929, aug 2000.

[25] Sean Meyn. Stability and Optimization of Queueing Networks and Their Fluid Models. *Lectures in applied mathematics-American Mathematical Society*, 33:175–200, 1997.

[26] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. feb 2016.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[28] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A Distributed Framework for Emerging AI Applications. dec 2018.

[29] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta Learn Fast: A New Benchmark for Generalization in RL. apr 2018.

[30] OpenAI. OpenAI Five, 2019.

[31] I.C. Paschalidis, C. Su, and M.C. Caramanis. Target-Pursuing Scheduling and Routing Policies for Multiclass Queueing Networks. *IEEE Transactions on Automatic Control*, 49(10):1709–1722, oct 2004.

[32] J.A. Ramirez-Hernandez and E. Fernandez. A Case Study in Scheduling Reentrant Manufacturing Lines: Optimal and Simulation-Based Approaches. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2158–2163. IEEE, 2005.

[33] Jose A. Ramirez-Hernandez and Emmanuel Fernandez. An Approximate Dynamic Programming Approach for Job Releasing and Sequencing in a Reentrant Manufacturing Line. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 201–208. IEEE, apr 2007.

[34] Jose A. Ramirez-Hernandez and Emmanuel Fernandez. Control of a re-entrant line manufacturing model with a reinforcement learning approach. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 330–335. IEEE, dec 2007.

[35] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. In *Proceeding ICML'15*, pages 1889–1897, feb 2015.

[36] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*, 2016.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. jul 2017.

[38] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, oct 2017.

[39] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning : an introduction*. MIT press, 2nd edition, 2018.

[40] Philip S Thomas. Bias in Natural Actor-Critic Algorithms. In *Proceedings of the 31 st International Conference on Machine Learning*, 2014.

[41] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, oct 2012.

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

[43] Lawrence M. Wein. Optimal Control of a Two-Station Brownian Network. *Mathematics of Operations Research*, 15(2):215–242, may 1990.

[44] R.J. Williams. Diffusion approximations for open multiclass queueing networks: sufficient conditions involving state space collapse. *Queueing Systems*, 30(1/2):27–88, 1998.