Queueing Network Controls via 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, 39], two-players games [52], team games [43]; are successful in wide range applications from medicine [30] to cooling datacenters [17]. Following the trend many dynamic resource allocation and sequential decision making problems in communications and networking are solved applying deep RL, see the review [11].

One of the hard problems in stochastic processing networks domain is the scheduling of multiclass queueing networks (MQNs). This type of problems have important applications in healthcare and communication management, data centers and manufacturing systems. MQNs involve two or more stations (servers) which process incoming tasks (jobs, customers) with deterministic or random routing. Each station of such a system typically has a limited service capacity and gives priority to a single job class, forcing the remaining jobs wait in associated buffers. In the scheduling problem of MQNs the decision maker determines what class should be processed first in each 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) [20, 21, 58]. 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 [59, 31], 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 [13]. 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 [36], discrete-review policies [34], 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 and remains feasible for large multiclass networks.

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.

Stochastic processes parameters typically are known and a fundamental difficulty in solving the MDP problem is the curse of dimensionality: the buffers capacity is unlimited and the corresponding optimization problem has infinite state space and infinite number of constrains. 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. Based on information that is available for the system manager we distinguish two approaches of solving MDPs: approximate dynamic programming and reinforcement learning.

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 [15, 2, 56].

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 [46], SARSA(λ) with linear Q-functions representation [47], simulation-based policy gradient [44], MCTS [7], model-based methods [33]. Except [7] , only small-size 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 [28]. 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 [48] 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) [48] 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 [50].

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 summarize the major contributions of our study:

1. In Section 3.2 we provide a theoretical justification that the trust region policy optimization algorithm can be applied for MDP problems with countable state space and unbounded cost-to-go function. We show that starting from a stable policy it is possible to improve long-run average performance with sufficiently small changes to the initial policy.

2. In Section 4 we discuss a new way of estimating advantage functions that combines a well-known in RL literature the generalized advantage estimation method [49] with approximating martingale-process method [23] which, to the best of our knowledge, has not been used in simulation-based optimization yet.

3. We provide extensive computational experiments in Section 5. We claim that the performance of scheduling policies resulting from the proposed RL algorithm outperform robust fluid network policies [8], [7] and other baseline heuristics.

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 [22] 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.

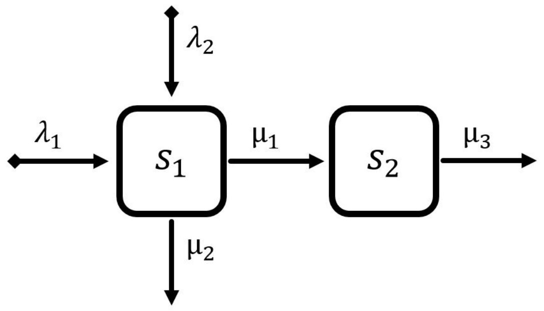


Figure 1: The criss-cross network

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 j jobs (including possibly the one in service) in the system at time $t,\ j = 1, 2, 3$. We use $x(t) = \left[x^1(t), x^2(t), x^3(t)\right]$ 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:

$$\inf_{\pi} \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\pi} \int_0^T \left[x^1(t) + x^2(t) + x^3(t)\right] dt \tag{2.1}$$

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 [32].

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 $\sum\limits_{j\in B_l} a_j(k) = 1$ 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 [39], [6], and physics engines [16], [55]. 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 strategies 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. In this section we propose a version of Proximal Policy Optimization algorithm which is appropriate to optimize the long-run average performance.

3.1 Some results from Markov chains theory

Let $\mathcal{X}$ be a countable state space and $\mathcal{M}_{\mathcal{X},\mathcal{X}}$ be a set of all matrices on the countable space $\mathcal{X} \times \mathcal{X}$.

For a vector ν on $\mathcal{X}$, we define $\mathbb{V}$-weighted norm w.r.t. a vector $\mathbb{V} : \mathcal{X} \to [1, \infty)$:

$$||\nu||_{\infty,\mathbb{V}} := \sup_{x\in X} \frac{|\nu(x)|}{\mathbb{V}(x)}$$

By Lemma 6 in Section 7.3 the corresponding induced $\mathbb{V}$-weighted operator norm for any matrix M on

$\mathcal{X} \times \mathcal{X}$ is given by

$$||M||_V = \sup_{x\in\mathcal{X}} \frac{1}{\mathbb{V}(x)} \sum_{y\in\mathcal{X}} |M(x,y)|\mathbb{V}(y)$$

We note that for any matrices $A, B, C \in \mathcal{M}_{\mathcal{X}\times\mathcal{X}}$ s.t. $||A||_{\mathbb{V}} < \infty, ||B||_{\mathbb{V}} < \infty, ||C||_{\mathbb{V}} < \infty$ commutativity

$$(A+B)C = AC + BC \qquad A(B+C) = AB + AC$$

and associativity

$$ABC = (AB)C = A(BC)$$

properties hold, see [26, Proposition 2.1, Lemma 2.1].

Consider an irreducible Markov chain on a countable state space $\mathcal{X}$ with a transition matrix $P \in \mathcal{M}_{\mathcal{X},\mathcal{X}}$. We denote $P(y|x)$ the transition probability from state $x \in \mathcal{X}$ to state $y \in \mathcal{X}$.

Assume that there exists a vector $\mathbb{V} : \mathcal{X} \to [1, \infty)$ s.t. the following drift condition holds for some constants $b < 1$, $d \geq 0$ and a finite subset $C \subset \mathcal{X}$:

$$\sum_{y\in\mathcal{X}} P(y|x)\mathbb{V}(y) \leq b\mathbb{V}(x) + d\mathbb{I}_C(x), \quad \text{for each } x \in \mathcal{X}, \tag{3.1}$$

where $\mathbb{I}_C(x) = 1$ if and only if $x \in C$, otherwise $\mathbb{I}_C(x) = 0$.

Then the Markov chain with the transition matrix P is positive recurrent with a unique stationary distribution μ [38, Theorem 11.3.4]. By [38, Theorem 14.3.7] we have

$$\mu^T\mathbb{V} < \infty,$$

where for any function $f : \mathcal{X} \to \mathbb{R}$ we define $\mu^T f$ as

$$\mu^T f := \sum_{x\in\mathcal{X}} \mu(x)f(x).$$

Vector $\mathbb{V}$ in the drift condition (3.1) is called a *Lyapunov function* for the Markov chain.

An irreducible, aperiodic Markov chain with a transition matrix P is called $\mathbb{V}$-*uniformly ergodic* if

$$||P^n - \Pi||_{\mathbb{V}} \to 0 \text{ as } n \to \infty,$$

where every row of Π equals to the stationary distribution μ, i.e. $\Pi(x,y) := \mu(y)$, for any $x, y \in \mathcal{X}$.

The drift condition (3.1) is sufficient and necessary for an irreducible, aperiodic Markov chain to be $\mathbb{V}$-uniformly ergodic [38, Theorem 16.0.1].

We define a cost function $g : \mathcal{X} \to \mathbb{R}$ s.t. $\mu^T|g| < \infty$ and consider the following equation:

$$g(x) - \mu^T g + \sum_{y\in\mathcal{X}} P(y|x)h(y) - h(x) = 0, \text{ for each } x \in \mathcal{X}, \tag{3.2}$$

Equation (3.2) is called a *Poisson's equation* of the Markov chain with transition matrix P and cost function g. Function $h : \mathcal{X} \to \mathbb{R}$ that satisfies (3.2) is called a *solution to Poisson's equation*. One can note that if some vector h is a solution to Poisson's equation (3.2), then for any $b \in \mathbb{R}$ vector $h + be$ is also a solution to the Poisson's equation. For example, a solution can be defined w.r.t. any positive recurrent state.

Lemma 1. *Consider $\mathbb{V}$-uniformly ergodic Markov chain with transition matrix P and the stationary distribution μ. Let $x^* \in X$ be an arbitrary state of the positive recurrent Markov chain.*

For any cost function $g : \mathcal{X} \to \mathbb{R}$ s.t. $|g| \leq \mathbb{V}$, the Poisson's equation (3.2) admits a unique solution

$$h^{(x^*)}(x) := \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \left(g(x_k) - \mu^T g\right) \,\middle|\, x_0 = x\right] \text{ for each } x \in \mathcal{X}, \tag{3.3}$$

where $\sigma(x^*) = \min\{k > 0 |\ x_k = x^*\}$ is the first future time when state x^* is visited.

Furthermore, the solution has a finite $\mathbb{V}-$weighted norm: $||h^{(x^*)}||_{\infty,\mathbb{V}} < \infty$.

The proof of Lemma 1 is given in [37, Proposition A.3.1]. We will refer to state x^* as a *regeneration state*, and to the times $\sigma(x^*)$ when the regeneration state is visited as *regeneration times*.

A solution to Poisson's equation h is called a *fundamental solution* if $\mu^T h = 0$. The proof of the following lemma is provided in [37, Proposition A.3.11].

Lemma 2. *Consider $\mathbb{V}$-uniformly ergodic Markov chain with transition matrix P and the stationary distribution μ.*

For any cost function $g : \mathcal{X} \to \mathbb{R}$ s.t. $|g| \leq \mathbb{V}$, the Poisson's equation (3.2) admits a unique fundamental solution

$$h^{(f)}(x) := \mathbb{E}\left[\sum_{k=0}^{\infty} \left(g(x_k) - \mu^T g\right) |x_0 = x\right] \text{ for each } x \in \mathcal{X}, \tag{3.4}$$

where x_k is the state of the Markov chain at time k.

Let $e = (1, 1, .., 1, ...)^T$ be a unit vector. It follows from Lemma 2 that equation

$$(I - P + \Pi)h = g - (\mu^T g)e$$

has a unique solution whenever $|g| \leq \mathbb{V}$. We can define a *fundamental matrix* Z that maps any cost function $|g| \leq \mathbb{V}$ into a corresponding fundamental solution $h^{(f)}$ as

$$Z := \sum_{k=0}^{\infty} (P - \Pi)^k \tag{3.5}$$

s.t. the fundamental solution (3.4) is equal to $h^{(f)} = Z\left(g - (\mu^T g)e\right)$. It follows from [38, Theorem 16.1.2] that the series (3.5) converges in $\mathbb{V}$-weighted norm and, moreover, $||Z||_{\mathbb{V}} < \infty$.

We want to note that if h_1 and h_2 are two solutions to Poisson's equation (3.2) with $\mu^T(|h_1| + |h_2|) < \infty$, then there exists a constant $b \in \mathbb{R}$ s.t. $h_1(x) = h_2(x) + b$ for each $x \in \mathcal{X}$, see [38, Proposition 17.4.1]. For example, for any regeneration state $x^* \in \mathcal{X}$:

$$h^{(x^*)}(x) = h^{(f)}(x) - h^{(f)}(x^*) \text{ for each } x \in \mathcal{X}. \tag{3.6}$$

3.2 Improvement guarantee for average cost objective

Consider a MDP problem with a countable state space $\mathcal{X}$, finite action space $\mathcal{A}$, one-step cost function $g(x)$ and transition function $P(\cdot|x, a)$. We assume that for each state-action pair (x, a) the number of distinguish states where the chain can transit is finite, i.e. set $\{y \in \mathcal{X} : P(y|x,a) > 0\}$ is finite for each $(x,a) \in \mathcal{X} \times \mathcal{A}$.

Suppose that $\{\theta \in \Theta\}$ denotes some open subset of Euclidean space. With every $\theta \in \Theta$, we associate a randomized Markovian policy π_θ, which at any state $x \in \mathcal{X}$ chooses action $a \in \mathcal{A}$ with probability $\pi_\theta(a|x)$. The corresponding family of transition kernels is given by

$$\left\{ P_\theta(\cdot|x) : \ P_\theta(\cdot|x) = \sum_{a\in A} \pi(a|x) P(\cdot|x,a), \ \theta \in \Theta \right\}.$$

We assume that each of the resulting Markov chains with transition probabilities $P_\theta(\cdot|x)$, $\theta \in \Theta$ is irreducible and aperiodic.

We also assume that there exists $\eta \in \Theta$, s.t. the drift condition (3.1) is satisfied for the transition kernel P_η with a Lyapunov function $\mathbb{V} : \mathcal{X} \to [1,\infty)$. By Lemma 2 a corresponding fundamental matrix Z_η is well-defined. For any cost function $|g| < V$ we denote a corresponding fundamental solution to the Poisson's equation as h_η.

Lemma 3. *Assume that drift condition (3.1) holds for P_η, $\eta \in \Theta$. Let some $\theta \in \Theta$ satisfies*

$$||[P_\theta - P_\eta] Z_\eta||_{\mathbb{V}} < 1$$

Then the Markov chain with transition matrix P_θ has a unique stationary distribution μ_θ.

The proof of Lemma 3 can be found in Section 7.3.

For each policy π_θ, $\theta \in \Theta$, s.t. the corresponding Markov chain P_θ is positive recurrent, we consider a long-run average cost as a performance measure. Let μ_θ be the stationary distribution of P_θ, then we define the long-run average cost objective as

$$\mu_\theta^T g := \sum_{x\in\mathcal{X}} \mu_\theta(x) g(x). \tag{3.7}$$

The following theorem provides a bound on the difference of long-run average performance of policies π_θ and π_η. The proof of Theorem 1 can be found in Section 7.3.

Theorem 1. *Suppose that the Markov chain with transition matrix P_η is an irreducible chain s.t. the drift condition (3.1) holds for some function $\mathbb{V} \geq 1$ and the cost function satisfies $|g| < \mathbb{V}$.*

For any $\theta \in \Theta$ s.t.

$$D_{\theta,\eta} := ||[P_\theta - P_\eta] Z_\eta||_{\mathbb{V}} < 1$$

the difference of long-run average costs of policies π_θ and π_η is bounded by:

$$\mu_\theta^T g - \mu_\eta^T g \ \leq \ M_1(\theta,\eta) + M_2(\theta,\eta), \tag{3.8}$$

where $M_1(\theta,\eta)$, $M_2(\theta,\eta)$ are finite and equal to

$$M_1(\theta,\eta) := \mu_\eta^T [g - (\mu_\eta^T g) e + P_\theta h_\eta - h_\eta]$$
$$M_2(\theta,\eta) := \frac{D_{\theta,\eta}^2}{1 - D_{\theta,\eta}} \left(1 + \frac{D_{\theta,\eta}}{(1 - D_{\theta,\eta})} (\mu_\eta^T \mathbb{V}) ||I - \Pi_\eta + P_\eta||_{\mathbb{V}} ||Z_\eta||_{\mathbb{V}} \right) \ ||g - (\mu_\eta^T g) e||_{\infty,\mathbb{V}} \ (\mu_\eta^T \mathbb{V})$$

It follows from Theorem 1 that the negativity of the RHS of inequality (3.8) guarantees the policy π_θ yields an improved performance comparing with the initial policy π_η.

One can get that $M_1(\theta,\eta) = O(D_{\theta,\eta})$:

$$\begin{aligned} |M_1(\theta,\eta)| &:= \left|\mu_\eta^T[g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta]\right| \\ &\leq (\mu_\eta^T \mathbb{V})||g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta||_{\infty,\mathbb{V}} \\ &= (\mu_\eta^T \mathbb{V})||[P_\theta - P_\eta]h_\eta||_{\infty,\mathbb{V}} \\ &= (\mu_\eta^T \mathbb{V})||[P_\theta - P_\eta]Z_\eta\left(g - (\mu_\eta^T g)e\right)||_{\infty,\mathbb{V}} \\ &\leq (\mu_\eta^T \mathbb{V})||g - (\mu_\eta^T g)e||_{\infty,\mathbb{V}} D_{\theta,\eta} \end{aligned}$$

The second term $M_2(\theta,\eta)$ is nonnegative, but $M_2(\theta,\eta) = O(D^2_{\theta,\eta})$, i.e. it is comparably small when policy π_η has been subject to small changes.

Hence, there is a chance to find $\theta \in \Theta$ s.t. $M_1(\theta,\eta)$ is negative and its absolute value is greater than the second term of the RHS of (3.8) for some sufficiently small $D_{\theta,\eta}$.

By minimizing the RHS of (3.8), we guarantee that the true objective η is non-increasing:

$$\mu_\theta^T g - \mu_\eta^T g \leq \min_{\theta\in\Theta:\ D_{\theta,\eta}<1} [M_1(\theta,\eta) + M_2(\theta,\eta)] \leq M_1(\eta,\eta) + M_2(\eta,\eta) = 0 \tag{3.9}$$

We can show that the distance $D_{\theta,\eta}$ can be controlled by the probability ratio $r_{\theta,\eta}(a|x) := \frac{\pi_\theta(a|x)}{\pi_\eta(a|x)}$ between the policies:

Lemma 4.

$$D_{\theta,\eta} \leq ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \sum_{a\in A} |r_{\theta,\eta}(a|x) - 1|\, G_\eta(x,a),$$

where $G_\eta(x,a) := \frac{1}{\mathbb{V}(x)} \sum_{y\in X} \pi_\eta(a|x)P(y|x,a)\mathbb{V}(y)$.

The proof of the lemma can be found in Appendix in Section 7.3. Lemma 4 implies that $D_{\theta,\eta}$ is small when $r_{\theta,\eta}(a|x)$ is close to 1 for each state-action pair (x,a). Note that $r_{\theta,\eta}(a|x) = 1$ and $D_{\theta,\eta} = 0$ when $\theta = \eta$.

3.3 Proximal Policy Optimization

In [28, 48] the authors propose to fix the maximum change between policies π_θ and π_η by bounding $M_2(\theta,\eta)$ term and focus on minimizing $M_1(\theta,\eta)$.

We define a relative advantage function $A_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ of policy π_θ, $\theta \in \Theta$ as

$$A_\theta(x,a) := \mathop{\mathbb{E}}_{x'\sim P(\cdot|x,a)} \left[g(x) - \mu_\theta^T g + h_\theta(x') - h_\theta(x)\right] \tag{3.10}$$

Then the first term on the RHS of (3.8) can be rewritten as

$$
\begin{aligned}
M_1(\theta,\eta) &= \mu_\eta^T[g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta] \\
&= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\theta(\cdot|x) \\ x'\sim P(\cdot|x,a)}} \left[g(x) - (\mu_\eta^T g)e + h_\eta(x') - h_\eta(x)\right] \\
&= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\theta(\cdot|x)}} A_\eta(x,a) \\
&= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} \left[\frac{\pi_\theta(a|x)}{\pi_\eta(a|x)} A_\eta(x,a)\right]
\end{aligned}
\tag{3.11}
$$

We want to accomplish two goals: keep the ratio $r_{\theta,\eta}(a|x) = \frac{\pi_\theta(a|x)}{\pi_\eta(a|x)}$ close to 1 and minimize

$$
M_1(\theta,\eta) = \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} r_{\theta,\eta}(a|x) A_\eta(x,a).
$$

In [50] the authors propose the clipped surrogate objective which is minimized w.r.t. $\theta \in \Theta$:

$$
L(\theta,\eta) := \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} \max\left[r_{\theta,\eta}(a|x)A_\eta(x,a),\ \operatorname{clip}(r_{\theta,\eta}(a|x), 1-\epsilon, 1+\epsilon)A_\eta(x,a)\right],
\tag{3.12}
$$

where $\operatorname{clip}(c, 1-\epsilon, 1+\epsilon) := \begin{cases} 1-\epsilon, & \text{if } c < 1-\epsilon, \\ 1+\epsilon, & \text{if } c > 1+\epsilon, \\ c, & \text{otherwise} \end{cases}$ for some $\epsilon \in (0,1)$.

The objective term $\operatorname{clip}(r_{\theta,\eta}(a|x), 1-\epsilon, 1+\epsilon)A_\eta(x,a)$ penalizes changes to the policy that move $r_{\theta,\eta}(a|x)$ far from 1. Then the final objective (3.12) is a upper bound (i.e., a pessimistic bound) on the unclipped objective. Thus, the objective makes $M_1(\theta,\eta)$ improve only within $\theta \in \Theta$ s.t. $r_{\theta,\eta} \in (1-\epsilon, 1+\epsilon)$.

Several alternative heuristics have been proposed in the literature (see, [48, 57, 50, 60]) to define a loss function which controls $M_2(\theta,\eta)$ and minimize $M_1(\theta,\eta)$ terms. We use loss function (3.12) in our study because of its implementation simplicity [50].

To compute objective function one needs to evaluate the expectation and precompute advantage function in (3.12). We use Monte-Carlo simulation method and replace the expectations in (3.12) and (3.10) by sample averages.

Let $\{x_0, x_1, ...\}$ be a simulated path of a positive recurrent Markov chain with transition matrix P. Let σ_n be the nth time when a regeneration state x^* is visited (we omit a superscript (x^*)). We refer to the sequence $\{x_{\sigma_n}, x_{\sigma_n+1}, .., x_{\sigma_{n+1}-1}\}$ as the nth *regenerative cycle*. We define the length of the nth regenerative cycle R_n by

$$
R_n := \sigma_{n+1} - \sigma_n.
$$

We also define

$$
G_n := \sum_{k=\sigma_n}^{\sigma_{n+1}-1} g(x_k)
$$

as a cumulative cost over nth regenerative cycle.

For any fixed recurrent state x^* the cycle lengths $\{R_n : n \geq 0\}$ are i.i.d and we denote the mean of the cycle length as $\mathbb{E}R$. The cumulative costs $\{G_n : n \geq 0\}$ are i.i.d. and we denote the mean of the cumulative cost over a cycle as $\mathbb{E}G$. By the renewal reward theorem [3, Section VI], the long-run average cost $\mu^T g$, when it is finite, is equal to

$$\mu^T g = \frac{\mathbb{E}G}{\mathbb{E}R}$$

A simulation experiment for long-run average cost estimation involves running one full regenerative cycle, i.e. $\{x_0, x_1, ..., x_{K-1}, x_{\sigma_1}\}$ s.t. $x_0 = x^*$. From one experiment, the estimates of $\mathbb{E}R$ and $\mathbb{E}G$ are equal to $\widehat{R}^{(1)} := \sigma_1$ and $\widehat{G}^{(1)} := \sum\limits_{k=0}^{K-1} g(x_k)$ correspondingly. If we continue the simulation and generate N independent repetitions (cycles), we get the following estimates:

$$\widehat{R}^{(N)} := \frac{1}{N}\sum_{n=1}^{N}(\sigma_{n+1} - \sigma_n) \quad \widehat{G}^{(N)} := \frac{1}{N}\sum_{n=1}^{N}\sum_{k=\sigma_n}^{\sigma_{n+1}-1} g(x_k)$$

Then, the expected average cost can be estimated by:

$$\widehat{\mu^T g}^{(N)} := \frac{\widehat{G}^{(N)}}{\widehat{R}^{(N)}} \tag{3.13}$$

In further discussion we refer to the average cost estimate as $\widehat{\mu^T g}$ without specifying the number of simulation experiment repetitions.

The estimation of the advantage function (3.10) requires availability of h_η. Even for a known policy it is computationally intractable task to solve Poisson's equation (3.2). Let assume that a trajectory which consists of N regenerative cycles have been generated and $\widehat{\mu_\eta^T g}$ has been computed. Consider an arbitrary state x_k from the generated trajectory under policy π_η. One-repetition estimate of solution to Poisson's equation (3.3) for a state x_k visited at time k is defined as

$$\hat{h}_k := \sum_{t=0}^{\sigma(k)-1}\left(g(x_{k+t}) - \widehat{\mu_\eta^T g}\right), \tag{3.14}$$

where $\sigma(k)$ is the first time when the regeneration state x^* is visited after time k. We note that the one-repetition estimate (3.14) is computed every timestep.

Remark 1. *For any state $x \in \mathcal{X}$ the one-repetition estimate (3.14) is computed every time when the state is visited (so-called every-visit Monte-Carlo method). One can also implement a first-visit Monte-Carlo method which implies that the one-repetition estimate is computed once for the first time when state x is visited and next visits at state x within the same cycle are ignored, e.g. [10]. For more details about every-visit and first-visit Monte-Carlo methods see [53, Section 5.1].*

Remark 2. *We want to highlight the importance of the use of regenerative simulation and discuss some critical disadvantages of alternative simulation schemes. One can estimate $h^{(f)}(x)$, $x \in \mathcal{X}$ by considering a sample path of the Markov chain $\{x_0, x_1, ...\}$ which starts at $x_0 = x$ and a stationary version of the* *sample path $\{y_0, y_1, ..\}$ with initial state y_0 sampled from the stationary distribution. Then a unbiased one-repetition estimate of the Poisson's solution (3.4) at state x is equal to*

$$\hat{h}^{(f)}(x) := \sum_{k=0}^{\sigma^{(couple)}} [g(x_k) - g(y_k)],$$

where $\sigma^{(couple)} = \min\{k \geq 0 | x_k = y_k\}$ is the time when the paths meet. Unfortunately, in practice the stationary distribution is unknown and one has to simulate stationary sample path by perfect sampling [45], which might require tremendous computational efforts. Also one may attempt to estimate average performance (3.7) without exploiting the regenerative structure – given a sample path $\{x_0, ..., x_N\}$ with an arbitrary initial distribution, the point estimator of average performance is defined as

$$\widehat{\mu^T g}^{(p)} := \frac{1}{N+1} \sum_{k=0}^{N} g(x_k) \tag{3.15}$$

The estimator (3.15) is typically biased. Therefore, if the average cost estimate required in (3.14) has been obtained based on equation (3.15), the estimator $\hat{h}(x)$ returns biased estimates of the solution to Poisson's equation. Section 5 in [12] provides more detailed discussion of the mentioned estimation schemes.

We use function $f_\psi : \mathcal{X} \to \mathbb{R}$ from a family of function approximators $\{f_\psi, \psi \in \Psi\}$ to represent function h_η. A single function f_ψ is chosen from $\{f_\psi, \psi \in \Psi\}$ to minimize the mean square distance to the one-repetition estimates $\{\hat{h}_k\}_{k=0}^{\sigma_N - 1}$:

$$\psi^* = \arg\min_{\psi \in \Psi} \sum_{k=0}^{\sigma_N - 1} \left(f_\psi(x_k) - \hat{h}_k \right)^2. \tag{3.16}$$

Remark 3. *One might expect that the estimation of solution to Poisson's equation h_η has to be done over multiple regenerative cycles before solving optimization problem (3.16). Let x_{k_n} be nth time in the simulation when state x has been visited with $x_{k_0} := x_k$, where x is an arbitrary state sampled from the generated trajectory. Let $n(x)$ be the total number of visits to state x over the simulation. We define $n(x)-$repetition estimate of $h_\eta(x)$ as*

$$\hat{h}^{(n)}(x_k) := \frac{1}{n(x_k)} \sum_{i=1}^{n(x_k)} \sum_{t=k_i}^{\sigma(x_{k_i})-1} \left(g(x_t) - \widehat{\mu_\eta^T g} \right),$$

Then the optimization problem

$$\psi^* = \arg\min_{\psi \in \Psi} \sum_{k=0}^{\sigma_N - 1} \left(f_\psi(x_k) - \hat{h}_\eta^{(n)}(x_k) \right)^2,$$

yields the same solution as problem (3.16), but requires additional efforts for the repetitions averaging.

The equivalence of the optimization problems follow from the fact that for an arbitrary sequence of real numbers $a_1, .., a_n \in \mathbb{R}$:

$$\arg\min_{x \in \mathbb{R}} \sum_{i=1}^{n} (x - a_i)^2 = \frac{1}{n} \sum_{i=1}^{n} a_i \tag{3.17}$$

With available function approximation of h_η, the advantage function (3.10) can be estimated as

$$\hat{A}_\eta(x_k, a_k) := g(x_k) - \widehat{\mu_\eta^T g} + \sum_{y\in\mathcal{X}} P(y|x_k, a_k) f_\psi(y) - f_\psi(x_k) \tag{3.18}$$

Given a trajectory of N regenerative cycles $\{x_0, a_0, x_1, a_1, ..., , x_{\sigma_N-1}, a_{\sigma_N-1}, x_{\sigma_N}\}$ which starts from the regeneration state $x_0 = x^*$ the loss function (3.12) is estimated as a sample average over the cycles:

$$\hat{L}(\theta,\eta) = \sum_{k=0}^{\sigma_N-1} \max\left[\frac{\pi_\theta(a_k|x_k)}{\pi_\eta(a_k|x_k)}\hat{A}_\eta(x_k,a_k), \text{clip}\left(\frac{\pi_\theta(a_k|x_k)}{\pi_\eta(a_k|x_k)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_\eta(x_k,a_k)\right]$$

If Q trajectories can be simulated in parallel, each of $q = 1, .., Q$ (parallel) actors collect a trajectory

$$\{x_{0,q}, a_{0,q}, x_{1,q}, a_{1,q}, ..., x_{\sigma_N-1,q}, a_{\sigma_N^q-1,q}, x_{\sigma_N^q,q}\}$$

with N regenerative cycles, where σ_N^q is Nth regeneration time in the simulation of qth actor and $x_{0,q} = x^*$ for each $q = 1, ..., Q$.

Each iteration the surrogate loss is constructed based on these $\sum\limits_{q=1}^{Q} \sigma_N^q$ timesteps of data. Optimization of the loss function yields a new policy for the next iteration.

Algorithm 1: Base proximal policy optimization algorithm

```
   Result: policy $\pi_{\theta_I}$
 1 Initialize policy $\pi_{\theta_0}$ and value function $f_{\psi_0}$ approximators ;
 2 for policy iteration $i = 1, 2, ..., I$ do
 3     for actor $q = 1, 2, ..., Q$ do
 4         Run policy $\pi_{\theta_i}$ until it reaches $N$th regeneration time on $\sigma_N^q$ step: collect an episode
             $\{x_{0,q}, a_{0,q}, x_{1,q}, a_{1,q}, ...., x_{\sigma_N-1,q}, a_{\sigma_N^q-1,q}, x_{\sigma_N^q,q}\}$;
 5     end
 6     Compute average cost estimate $\widehat{\mu^T g}$ by (3.13);
 7     Compute $\hat{h}_{k,q}$, estimate of $h_{\theta_i}(x_{k,q})$, by (3.14) for each $q = 1, .., Q$, $k = 1, .., \sigma_N^q - 1$;
 8     Update $\psi_i := \psi$, where $\psi$ minimizes $\arg\min_{\psi\in\Psi} \sum_{q=1}^{Q}\sum_{k=0}^{\sigma_N^q-1}\left(f_\psi(x_{k,q}) - \hat{h}_{k,q}\right)^2$ following (3.16) ;
 9     Estimate advantage functions $\hat{A}_{\theta_i}(x_{k,q}, a_{k,q})$ using (3.18) for each $q = 1, .., Q$, $k = 1, .., \sigma_N^q - 1$;
10     Minimize surrogate objective function w.r.t. $\theta \in \Theta$:

         $\hat{L}(\theta,\theta_i) = \sum_{q=1}^{Q}\sum_{k=0}^{\sigma_N^q-1} \max\left[\frac{\pi_\theta(a_{k,q}|x_{k,q})}{\pi_{\theta_i}(a_{k,q}|x_{k,q})}\hat{A}_{\theta_i}(x_{k,q},a_{k,q}), \text{clip}\left(\frac{\pi_\theta(a_{k,q}|x_{k,q})}{\pi_{\theta_i}(a_{k,q}|x_{k,q})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{\theta_i}(x_{k,q},a_{k,q})\right]$

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

4 Advantage function estimation

The main goal of the advantage function $A_\eta(x, a)$ is to predict the effect of action a realization at state x. If approximation f_ψ is not sufficiently close to a true solution of Poisson's equation, the effect of the action on the future can be evaluated incorrectly by estimator (3.18) that can be a cause of suboptimal policy update.

4.1 Approximating martingale-process method

Estimator (3.14) of the solution to Poisson's equation suffers from the high variance when the regenerative cycles are long – the sum of relative costs $g(x_k) - \mu_\eta^T g$ becomes large before it is reset to zero at the regeneration state. In this section we discuss how one can decrease the variance by reducing the magnitude of summands in (3.14) if an approximation ζ of the solution to Poisson's equation h_η is available.

Let assume a trajectory $\{x_0, a_1, x_1, a_2, ..., x_{K-1}, a_{K-1}, x_{\sigma_N}\}$ has been generated under policy π_η. From the definition of a solution to Poisson's equation (3.2):

$$g(x_k) - \mu^T g = \sum_{y\in X} P_\eta(y|x_k)h_\eta(y) - h_\eta(x_k) \text{ for each state } x_k \text{ in the simulated trajectory.}$$

If the approximation ζ is sufficiently close to h_η, then the correlation between

$$g(x_k) - \widehat{\mu^T g} \quad \text{and} \quad \sum_{y\in X} P_\eta(y|x_k)\zeta(y) - \zeta(x_k)$$

is positive and the control variate method can be used to reduce the variance. This idea gives rise to the approximating martingale-process (AMP) method proposed in [23].

Following [23, Proposition 7] we define a martingale process $M = (M_n : n \geq 0)$, where for $x = x_0$

$$M_n(x) := \zeta(x_0) - \zeta(x_n) + \sum_{k=0}^{n-1}\left[\sum_{y\in\mathcal{X}} P_\eta(y|x_k)\zeta(y) - \zeta(x_k)\right] \tag{4.1}$$

The martingale process (4.1) has zero expectation $\mathbb{E}M_n = 0$ for all $n \geq 0$ and can be used as a control variate.

From equality (3.6) $h_\eta(x^*) = 0$ and we can assume that $\zeta(x^*) = 0$. Consider the martingale process starting from an arbitrary state x_k until the first regeneration time:

$$M_{\sigma_k}(x_k) = \zeta(x_k) + \sum_{t=0}^{\sigma_k-1}\left[\sum_{y\in\mathcal{X}} P_\eta(y|x_{k+t})\zeta(y) - \zeta(x_{k+t})\right]$$

Then adding M_{σ_k} to estimator (3.14) we get the AMP estimator of the solution to Poisson's equation:

$$\begin{aligned}\hat{h}_\eta^{AMP(\zeta)}(x_k, a_k) &:= \zeta(x_k) + \sum_{t=0}^{\sigma_k-1}\left(g(x_{k+t}) - \widehat{\mu_\eta^T g} + \sum_{y\in\mathcal{X}} P_\eta(y|x_{k+t})\zeta(y) - \zeta(x_{k+t})\right) \\ &= g(x_k) - \widehat{\mu_\eta^T g} + \sum_{y\in\mathcal{X}} P_\eta(y|x_k)\zeta(y) + \sum_{t=1}^{\sigma_k-1}\left(g(x_{k+t}) - \widehat{\mu_\eta^T g} + \sum_{y\in\mathcal{X}} P_\eta(y|x_{k+t})\zeta(y) - \zeta(x_{k+t})\right).\end{aligned} \tag{4.2}$$

Let assume that estimation of the average performance is accurate, i.e. $\widehat{\mu_\eta^T g} = \mu_\eta^T g$. In this case estimator (4.2) has zero variance if the approximation is exact $\zeta = h_\eta$.

Now we want replace the standard regenerative estimation in line 8 of Algorithm 1 by AMP estimator (4.2). As the approximation ζ we use approximation $f_{\psi_{i-1}}$ that corresponds to preceding policy, i.e. in line 8 of Algorithm 1 we replace $\hat{h}(x_k)$ by estimates $\hat{h}^{AMP(f_{\psi_{i-1}})}(x_k)$ that are computed by (4.2).

```
Algorithm 2: Proximal policy optimization with AMP method
   Result: policy $\pi_{\theta_I}$
1  Initialize policy $\pi_{\theta_0}$ and value function $f_{\psi_0}$ approximators ;
2  for policy iteration $i = 1, 2, ..., I$ do
3      for actor $q = 1, 2, ..., Q$ do
4          Run policy $\pi_{\theta_i}$ until it reaches $N$th regeneration time on $\sigma_N^q$ step: collect an episode
             $\{x_{0,q}, a_{0,q}, x_{1,q}, a_{1,q}, ...., x_{\sigma_N-1,q}, a_{\sigma_N^q-1,q}, x_{\sigma_N^q,q}\}$;
5      end
6      Compute average cost estimate $\widehat{\mu^T g}$ by (3.13);
7      Compute $\hat{h}_{k,q}^{AMP(f_{\psi_{i-1}})}$, estimate of $h_{\theta_i}(x_{k,q})$, by (4.2) for each $q = 1, .., Q$, $k = 1, .., \sigma_N^q - 1$;
8      Update $\psi_i := \psi$, where $\psi$ minimizes $\arg\min_{\psi\in\Psi} \sum_{q=1}^{Q} \sum_{k=0}^{\sigma_N^q-1} \left( f_\psi(x_{k,q}) - \hat{h}_{k,q}^{AMP(f_{\psi_{i-1}})} \right)^2$ following
        (3.16) ;
9      Estimate advantage functions $\hat{A}_{\theta_i}(x_{k,q}, a_{k,q})$ using (3.18) for each $q = 1, .., Q$, $k = 1, .., \sigma_N^q - 1$;
10     Minimize surrogate objective function w.r.t. $\theta \in \Theta$:
         $$\hat{L}(\theta, \theta_i) = \sum_{q=1}^{Q} \sum_{k=0}^{\sigma_N^q-1} \max \Big[ \frac{\pi_\theta(a_{k,q}|x_{k,q})}{\pi_{\theta_i}(a_{k,q}|x_{k,q})} \hat{A}_{\theta_i}(x_{k,q}, a_{k,q}), \operatorname{clip}\left( \frac{\pi_\theta(a_{k,q}|x_{k,q})}{\pi_{\theta_i}(a_{k,q}|x_{k,q})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_{\theta_i}(x_{k,q}, a_{k,q}) \Big]$$
       Update $\theta_{i+1} := \theta$
11 end
```

4.2 Generalized estimation

In the previous section we applied AMP method to reduce the variance of the summands in (3.14). Unless an approximation ζ is exact, each term in the summation in (4.2) is random with nonzero variance. When the expected length of a regeneration cycle is large the cumulative variance in estimator (4.2) still can be devastating.

The common approach to overcome this issue is to introduce a forgetting factor $\gamma < 1$ to discount the future relative costs [25, 5, 35, 27, 54, 49]. We approximate solution of Poisson's equation h_η by a corresponding infinite-horizon discounted relative value function:

$$J_\eta^{(\gamma)}(x) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t+1}(g(x_t) - \mu_\eta^T g) \Big| x_0 = x \right] \quad \text{for each } x \in \mathcal{X},$$

where x_k is the state of the Markov chain P_η at time k and $\gamma < 1$ is a discount factor.

We note that $J_\eta^{(\gamma)} \to h_\eta$ in $\mathbb{V}$-weighted norm as $\gamma \to 1$ for $\mathbb{V}$-uniformly ergodic Markov chain, see Lemma 7.

Now we replace h_η by $J_\eta^{(\gamma)}$ in the expression for $M_1(\theta,\eta)$:

$$M_1^{(\gamma)}(\theta,\eta) := \mu_\eta^T(g - \mu_\eta^T g e + P_\theta J_\eta^{(\gamma)} - J_\eta^{(\gamma)})$$

We define an infinite-horizon discounted value function

$$V_\eta^{(\gamma)}(x) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t g(x_t) \middle| x_0 = x\right] \text{ for each } x \in \mathcal{X},$$

and note that $J_\eta^{(\gamma)}(x) = \gamma V_\eta^{(\gamma)}(x) + \frac{\gamma}{1-\gamma}\mu_\eta^T g$ for each $x \in \mathcal{X}$.

The optimization problem (3.8) searches $\theta \in \Theta$ s.t. $M_1(\theta,\eta) + M_2(\theta,\eta)$ is minimized. Its solution can be approximately found minimizing $\mu_\eta^T(g + \gamma P_\theta V_\eta^{(\gamma)} - V_\eta^{(\gamma)}) + M_2(\theta,\eta)$ since

$$\begin{aligned}\arg\min_{\theta\in\Theta}[M_1(\theta,\eta) + M_2(\theta,\eta)] &\approx \arg\min_{\theta\in\Theta}[M_1^{(\gamma)}(\theta,\eta) + M_2(\theta,\eta)] \\ &= \arg\min_{\theta\in\Theta}\left[\mu_\eta^T(g - \mu_\eta^T g e + P_\theta J_\eta^{(\gamma)} - J_\eta^{(\gamma)} + M_2(\theta,\eta)\right] \\ &= \arg\min_{\theta\in\Theta}\left[\mu_\eta^T(g - \mu_\eta^T g e + \gamma P_\theta V_\eta^{(\gamma)} - \gamma V_\eta^{(\gamma)}) + M_2(\theta,\eta)\right] \\ &= \arg\min_{\theta\in\Theta}\left[\mu_\eta^T(g + \gamma P_\theta V_\eta^{(\gamma)} - V_\eta^{(\gamma)}) + M_2(\theta,\eta)\right]\end{aligned}$$

We define an infinite-horizon discounted advantage function $A_\theta^{(\gamma)} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ of policy π_θ, $\theta \in \Theta$ as

$$A_\theta^{(\gamma)}(x,a) := \mathop{\mathbb{E}}_{x'\sim P(\cdot|x,a)}\left[g(x) + \gamma V_\theta^{(\gamma)}(x') - V_\theta^{(\gamma)}(x)\right] \tag{4.3}$$

Let episode $\{x_0, a_0, ..., x_{N_1}, a_{N-1}\}$ be generated by policy π_η. With available approximation f_ψ of the discounted value function $V_\eta^{(\gamma)}$, the advantage function (4.3) can be estimated as

$$\hat{A}_\eta^{(\gamma)}(x_k, a_k) := g(x_k) + \gamma \sum_{y\in\mathcal{X}} P(y|x_k,a_k) f_\psi(y) - f_\psi(x_k), \tag{4.4}$$

for each state-action pair (x_k, a_k) from the generated episode.

Then we can rewrite $M_1^{(\gamma)}(\theta,\eta)$ using an advantage function definition following (3.11):

$$\begin{aligned}M_1^{(\gamma)}(\theta,\eta) &= \mu_\eta^T(g + \gamma P_\theta V_\eta^{(\gamma)} - V_\eta^{(\gamma)}) \\ &= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}}\left[\frac{\pi_\theta(a|x)}{\pi_\eta(a|x)} A_\eta^{(\gamma)}(x,a)\right]\end{aligned}$$

The PPO loss function becomes:

$$L^{(\gamma)}(\theta,\eta) := \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} \max\left[r_{\theta,\eta}(a|x) A_\eta^{(\gamma)}(x,a),\ \text{clip}(r_{\theta,\eta}(a|x), 1-\epsilon, 1+\epsilon) A_\eta^{(\gamma)}(x,a)\right]. \tag{4.5}$$

The AMP estimator for the discounted value function can be derived following the procedure in the previous section. The details can be found in Appendix in Section 7.2. Let ζ be an approximation of the discounted value function. We get the corresponding AMP estimator for $V_{\eta}^{(\gamma)}(x_k)$:

$$\hat{V}_k^{AMP(\zeta),(\gamma)} := g(x_k) + \gamma \sum_{y\in X} P_\eta(y|x_k)\zeta(y) + \sum_{k=1}^{\infty} \gamma^k \left(g(x_k) + \gamma \sum_{y\in X} P_\eta(y|x_k)\zeta(y) - \zeta(x_k) \right) \tag{4.6}$$

Similarly to (4.2), estimator (4.6) has zero variance if approximation of the discounted value function is exact $\zeta = V_\eta$.

Further variance reduction in discounted value function estimation is possible when the summation in (4.6) is done only for the next T timesteps after time k. Generalizing this choice one can assume that the number of summands T follows geometrical distribution with parameter $\lambda < 1$ as in TD(λ) method [53, Section 12]. We use a truncated version of TD(λ) which reduces to (4.6) when $\lambda = 1$ and $N = \infty$:

$$\hat{V}_k^{AMP(\zeta),(\gamma,\lambda)} := g(x_k) + \gamma \sum_{y\in X} P_\eta(y|x_k)\zeta(y) + \sum_{t=1}^{N-k} (\gamma\lambda)^k \left(g(x_{k+t}) + \gamma \sum_{y\in X} P_\eta(y|x_{k+t})\zeta(y) - \zeta(x_{k+t}) \right). \tag{4.7}$$

We provide the proximal policy optimization algorithm where each of $q = 1, .., Q$ parallel actors simulate an episode with length N: $\{x_{0,q}, a_{0,q}, x_{1,q}, a_{1,q}, ..., x_{N-1,q}, a_{N-1,q}\}$. Formula (4.7) is used to estimate discounted value function values for each state $x_{k,q}$ in the simulated data. As the approximation ζ we use approximation of the discounted value function of the preceding policy $\zeta = f_{\psi_{i-1}}$, see Algorithm 3.

Algorithm 3: Proximal policy optimization with discounting

```
   Result: policy $\pi_{\theta_I}$
 1 Initialize policy $\pi_{\theta_0}$ and value function $f_{\psi_0}$ approximators ;
 2 for policy iteration $i = 1, 2, ..., I$ do
 3     for actor $q = 1, 2, ..., Q$ do
 4         Run policy $\pi_{\theta_i}$ for $N$ timesteps: collect an episode
             $\{x_{0,q}, a_{0,q}, x_{1,q}, a_{1,q}, ...., x_{N-1,q}, a_{N-1,q}, x_{N,q}\}$;
 5     end
 6     Compute $\hat{V}_{k,q}^{AMP(f_{\psi_{i-1}}),(\gamma,\lambda)}$ estimates by (4.7) for each $q = 1, .., Q$, $k = 1, .., N-1$;
 7     Update $\psi_i := \psi$, where $\psi$ minimizes $\arg\min_{\psi\in\Psi} \sum_{q=1}^{Q}\sum_{k=0}^{N-1}\left(f_\psi(x_{k,q}) - \hat{V}_{k,q}^{AMP(f_{\psi_{i-1}}),(\gamma,\lambda)}\right)^2$ following
         (3.16) ;
 8     Estimate advantage functions $\hat{A}_{\theta_i}^{(\gamma)}(x_{k,q}, a_{k,q})$ using (4.4) for each $q = 1, .., Q$, $k = 1, .., N-1$;
 9     Minimize surrogate objective function w.r.t. $\theta \in \Theta$:
           $\hat{L}^{(\gamma)}(\theta, \theta_i) = \sum_{q=1}^{Q}\sum_{k=0}^{N-1} \max\Big[\frac{\pi_\theta(a_{k,q}|x_{k,q})}{\pi_{\theta_i}(a_{k,q}|x_{k,q})}\hat{A}_{\theta_i}^{(\gamma)}(x_{k,q}, a_{k,q}),$
               $\text{clip}\left(\frac{\pi_\theta(a_{k,q}|x_{k,q})}{\pi_{\theta_i}(a_{k,q}|x_{k,q})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{\theta_i}^{(\gamma)}(x_{k,q}, a_{k,q})\Big]$
       Update $\theta_{i+1} := \theta$
10 end
```

5 Experimental results for multiclass queuing networks

In this section we evaluate performance of the proposed proximal policy optimization Algorithms 1, 2 and 3 for multiclass queuing networks control optimization task discussed in Section 2.

We use two separate fully-connected feed-forward neural networks to represent policies π_θ, $\theta \in \Theta$ and value functions f_ψ, $\psi \in \Psi$ with the architecture details provided in Section 7.4 in Appendix. We will refer to the neural network used to represent a policy as *the policy NN* and to the neural network used to approximate a value function as *the value function NN*. In each experiment we run the algorithm for $I = 200$ iterations. Each iteration the algorithm uses $Q = 50$ actors to simulate data in parallel. See Section 7.5 in the Appendix for more details on the experimental setup and hyperparameters.

5.1 Criss-cross network

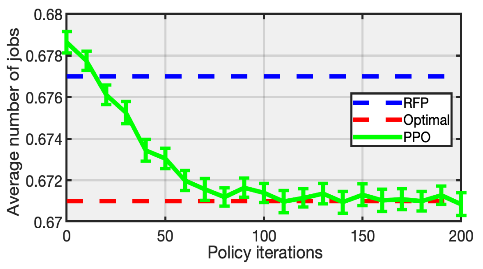
We first study PPO algorithm and compare its base version Algorithm 1 and its modification Algorithm 2 that incorporates the AMP method. We check the robustness of the algorithms for the criss-cross system with various load (traffic) intensity regimes, including I.L. (imbalanced light), B.L. (balanced light), I.M. (imbalanced medium), B.M. (balanced medium), I.H. (imbalanced heavy), and B.H. (balanced heavy) regimes. The corresponding arrival and service rates are presented in Table 1. The criss-cross network executed with any of these traffic regimes is stable under any work-conserving policy [14].

We summarize control policies that have been proposed in the literature in Table 2. In the first column of Table 2 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 [44] and denoted by OTP in the table. In the fourth column we list the performance of a threshold policy proposed in [22]. In the fifth and sixth columns we list the performance of fluid (FP) and robust fluid (RFP) policies from [8].

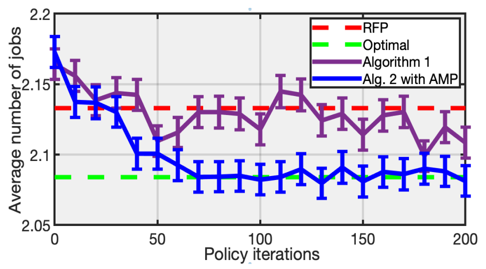
In the last column of Table 2 we provide simulation results of PPO policy π_{θ_I} obtained from Algorithm 2. We want to highlight that we report the performance of the policy resulting from the *last* iteration of the algorithm, which might be not the best policy over the course of learning.

The policy NN parameters θ_0 are initialized using standard Xavier initialization [18]. The resulting policy π_{θ_0} is close to the policy that chooses actions uniformly at random. We take the empty system state $x^* = (0, 0, 0)$ as a regeneration state and simulate $N = 5000$ independent regenerative cycles per actor in each iteration of the algorithm. Although the number of generated cycles is fixed for all traffic regimes, the length of regenerative cycles varies and highly depends on the load.

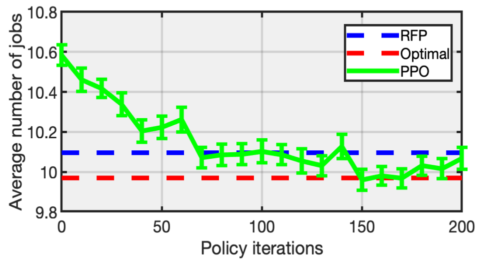
In order to present the learning curves in Figure 9 we save policy parameters $\{\theta_i\}_{i=0,10,...,200}$ every 10th iteration over the course of learning. After the algorithm terminates we independently simulate policies $\{\pi_{\theta_i} : i = 0, 10, ..., 200\}$ starting from the regeneration state $x = (0, .., 0)$ until fixed number of regenerative events happened. For light (medium, heavy) traffic regime the simulation is run for 10^7 (5×10^6, 10^6) regenerative cycles correspondingly. 95%−confidence intervals are computed using the strongly consistent estimator of asymptotic variance, see details in [3, Section VI.2d].



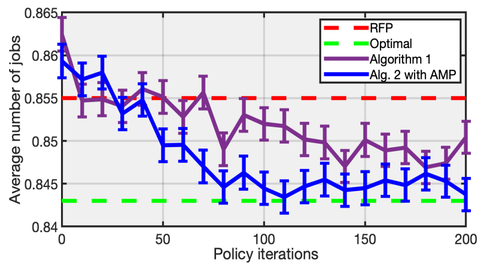
(a) Imbalanced low (IL) traffic



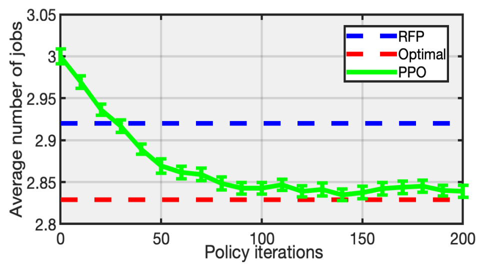
(b) Imbalanced medium (IM) traffic



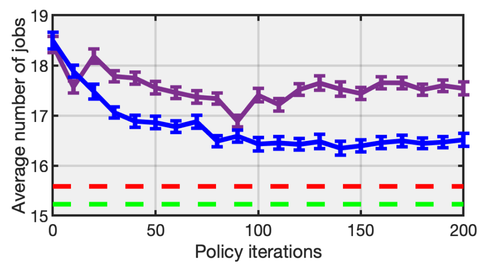
(c) Imbalanced heavy (IH) traffic



(d) Balanced low (BL) traffic



(e) Balanced medium (BM) traffic



(f) Balanced heavy (BH) traffic

Figure 2: Comparison of learning curves from Algorithm 1 and Algorithm 2 on the criss-cross network with different traffic regimes. *The purple (blue) solid line* shows the performance of PPO policies obtained at the end of every 10th iterations of Algorithm 1 (Algorithm 2, correspondingly); *red dash line* –performance of the robust fluid policy (RFP), *green dash line* –performance of the optimal policy.

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	PPO (Algorithm 2)
I.L.	0.671	0.678	0.679	0.678	0.677	0.671
B.L.	0.843	0.856	0.857	0.857	0.855	0.844
I.M.	2.084	2.117	2.129	2.162	2.133	2.084
B.M.	2.829	2.895	2.895	2.965	2.920	2.839
I.H.	9.970	10.13	10.15	10.398	10.096	10.067
B.H.	15.228	15.5	15.5	18.430	15.585	16.513

Table 2: Average number of jobs per unit time in the criss-cross network under different polices. The traffic regime in the network varies from imbalanced low to balanced heavy.

Unfortunately, Algorithm 2 is not robust enough and converges to a suboptimal policy when the criss-cross network operates in balanced heavy load regime. We run Algorithm 3 with discount factor $\gamma = 0.998$ and TD parameter $\lambda = 0.99$. Each iteration we use $Q = 50$ parallel processes to generate trajectories each with length $N = 50000$. We note that Algorithm 3 uses approximately 10 times less samples per iteration than

Algorithm 2. Algorithm 3 outputs policy $\pi_{\theta_{200}}$ which performance is 15.353 jobs per unit time. See the comparison of the learning curves of Algorithm 2 and Algorithm 3 in Figure 3.

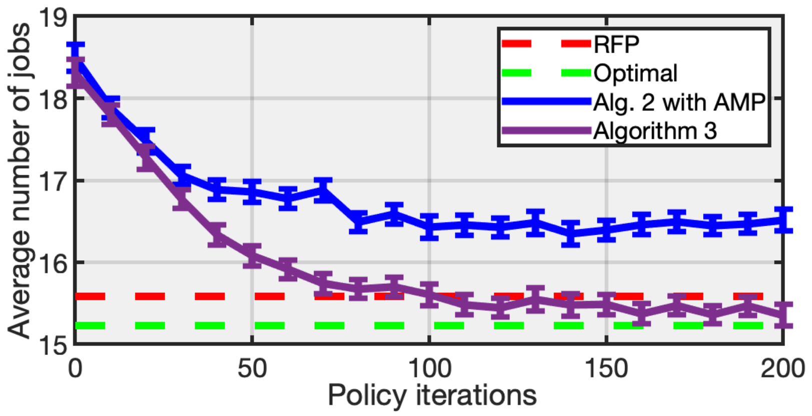


Figure 3: Comparison of learning curves from Algorithm 2 and Algorithm 3 on the criss-cross network with balanced heavy regime.

In the next section we test the robustest of Algorithm 3 to the size of the queuing network.

5.2 Extended six-class queuing network

In the next experiment we consider the family of extended six-class networks from [8] and apply Algorithm 3 to find good control policies.

The structure of the extended six-class networks is shown in Figure 4. The experiments have been run 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. The load is the same for each station and is equal to $\rho = 0.9$.

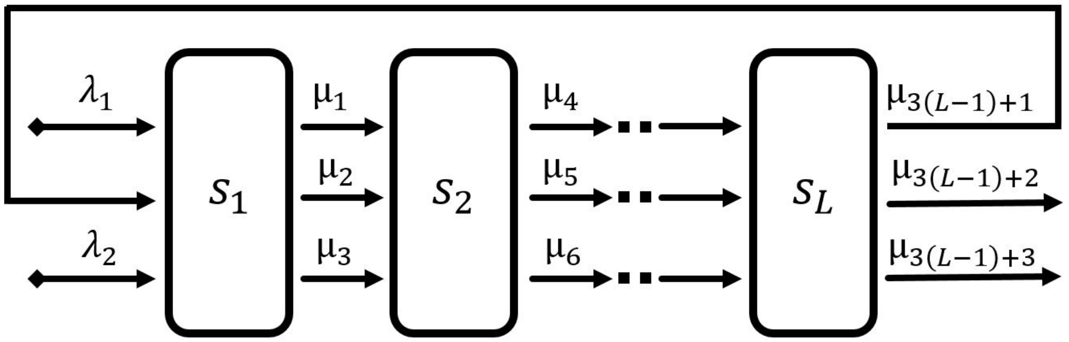


Figure 4: The extended six-class network

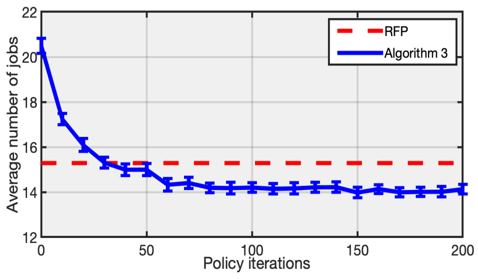
Table 3 provides the performance of the PPO policy and compares it with other heuristic methods for the extended six-class queuing networks. In the experiments we change the size of the network to test the robustness of the PPO policies. The size of the networks varies from 6 to 21 classes. In all experiments we generate $Q = 50$ episodes with $N = 50000$ timesteps. The discount factor and TD parameter are fixed and equal to $\gamma = 0.998$ and $\lambda = 0.99$ correspondingly. In the table FP and RFP refer to fluid and robust fluid policies [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.

We have found that Xavier initialization of the policy NN yields unstable policy for extended six-class networks. To overcome this issue we use a domain knowledge and pre-train the policy NN so that π_{θ_0} operates as a *proportional randomized (PR) policy* . When the network operates under the proportional randomized policy, whenever an arrival or service completion event happens in the system, a nonempty buffer j receives a priority over other classes associated to server $s(j)$ with probability

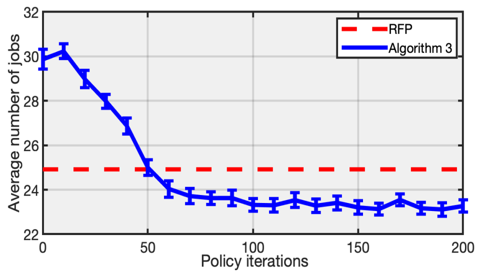
$$\frac{x^j}{\sum\limits_{i \in B_{s(j)}} x^i}, \tag{5.1}$$

where $B_{s(j)}$ is a set of buffers associated to server $s(j)$ and $x = (x^1, ..., x^J)$ is a state at the time of the event. The priority stays fixed until the next arrival or service completion event happens. The proportional randomized policy is similar to HLPPS (head-of-line proportional processor sharing) policy. The HLPPS policy allows processor sharing and all nonempty buffers receive service simultaneously. The fraction of server $s(j)$ capacity that is allocated to class j is equal to (5.1). Therefore, the HLPPS policy has the same expected service rate for each job class as the PR policy. The HLPPS policy is *maximally stable* for MQNs [9] meaning that if the system is unstable under HLPPS policy there is no other policy that can stabilize it. The PR policy is also maximally stable for open MQNs, see the proof in Section 7.1 in Appendix.

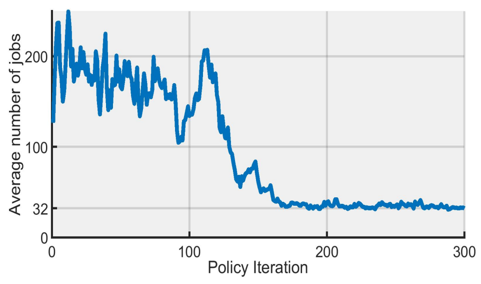
We save policy parameters $\{\theta_i\}_{i=0,10,...,200}$ every 10th iteration for further profound evaluation. The regenerative simulation is computationally intractable for large-size systems with a high load – time between returns to a regeneration state is very large, so that a run with many timesteps would be required to obtain even a single regenerative cycle. For large-size networks *the method of batch means* is recommended to estimate the performance [24, Section 6]. We simulate policies $\{\pi_{\theta_i} : i = 0, 10, ..., 200\}$ starting from an empty state $x = (0, .., 0)$ until 5×10^6 arrival events happened. For each simulated trajectory the confidence intervals are computed by splitting on 50 sub-episodes (batches), see also [41]. Then we estimate average performance (batch mean) based on each sub-episode. Pretending that the obtained 50 estimates are i.i.d. we compute 95%−confidence intervals shown on the plots.



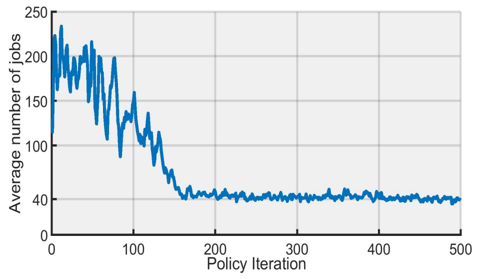
(a) 6-classes network



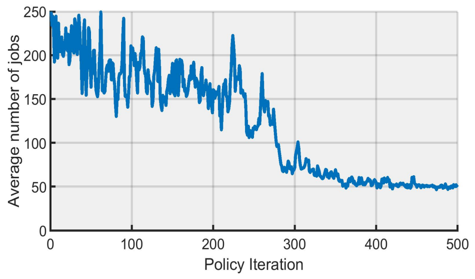
(b) 9-classes network



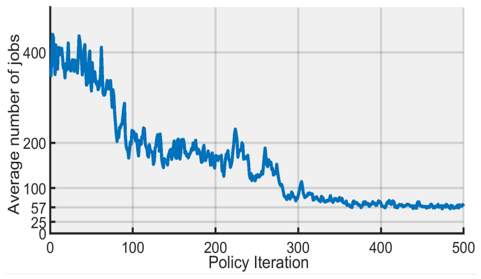
(c) 12-classes network



(d) 15-classes network



(e) 18-classes network



(f) 21-classes network

Figure 5: Learning curves from Algorithm 3 for the 6-class extended networks. *The blue solid line* shows the performance of PPO policies obtained at the end of every 10th iterations of Algorithm 3, *red dash line* –performance of the robust fluid policy (RFP).

Num. of classes $3L$	LBFS	FCFS	FP	RFP	PPO (Algorithm 3)
6	15.749	40.173	15.422	15.286	14.130
9	25.257	71.518	26.140	24.917	23.269
12	34.660	114.860	38.085	36.857	
15	45.110	157.556	45.962	43.628	
18	55.724	203.418	56.857	52.980	
21	65.980	251.657	64.713	59.051	

Table 3: Numerical results for the extended six-class network of Figure 4.

6 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 **??** 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 7.5 and neural network structures for policy parametrization and approximation of the value function in Section 7.4. 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 9. 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) [42].

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.

7 Appendix

7.1 Maximal Stability of Proportional Randomized Policy

In this section we consider proportional randomized policy.

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

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

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

$$\beta_i(t) = \begin{cases} \gamma_i \text{ if } z_i(t) > 0, \\ 0, \text{ otherwise} \end{cases} \tag{7.1}$$

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:

$$T_j(t) := \int_0^t \beta_i(u)du, \qquad \text{for each class } j \text{ and } t \geq 0. \tag{7.2}$$

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

$$\dot{Z}(t) = \lambda - (I - P')M\beta(t) \tag{7.3}$$
$$C\beta(t) \leq e \tag{7.4}$$
$$Z(t) \geq 0 \tag{7.5}$$
$$\beta(t) \geq 0 \tag{7.6}$$

The functions $Z_i(t)$ and $T_i(t)$ are absolutely continuous, and thus, differentiable almost everywhere. The equations 7.3 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 lengths at time t is

$$\hat{\beta}_i(t) := \begin{cases} \frac{z_i(t)}{\sum\limits_{j \in B_l} z_j(t)} \text{ if } \sum\limits_{j \in B_l} z_j(t) > 0 \\ 0, \text{ otherwise} \end{cases} \tag{7.7}$$

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

$$\xi_i^z(t) := \int_0^t (\beta_i^z(u) - \hat{\beta}_i^z(u))du \tag{7.8}$$

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

Lemma 5. *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:*

$$\lim_{r \to \infty} \frac{\xi^{z^r}(|z^r, \omega|)}{|z^r|} = 0 \tag{7.9}$$

Proof. By Chebyshev's inequality:

$$\mathbb{P}\left\{\omega : \frac{|\xi_i^{z^r}(|z^r, \omega|)|}{|z^r|} > \epsilon\right\} \leq \frac{\mathbb{E}\left[\xi_i^{z^r}(|z^r|)\right]^2}{\epsilon^2 |z^r|^2} \tag{7.10}$$

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

$$\mathbb{E}\left[\xi_i^{z^r}(|z^r|)\right]^2 = \mathbb{E}\left[\int_0^{|z^r|}(\beta(u)-\hat{\beta}(u))du\right]^2 = \mathbb{E}\left[\sum_{k=0}^{K}\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right]^2 = \tag{7.11}$$

$$\mathbb{E}\sum_{k=0}^{K}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)^2 + 2\sum_{k<k'}\mathbb{E}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)\left(\int_{\tau'_k}^{\tau_{k'+1}}(\beta(u)-\hat{\beta}(u))du\right) = \tag{7.12}$$

$$\mathbb{E}\sum_{k=0}^{K}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)^2 \leq \mathbb{E}\sum_{k=0}^{K}\left(\int_{\tau_k}^{\tau_{k+1}}1du\right)^2 \leq \mathbb{E}\sum_{\tau'\in\text{arrivals}}(\tau'_{k+1}-\tau'_k)^2 \leq O(|z^r|) \tag{7.13}$$

where the second inequality follows from the fact that

$$\mathbb{E}\sum_{k<k'}^{K}\mathbb{E}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)\left(\int_{\tau'_k}^{\tau_{k'+1}}(\beta(u)-\hat{\beta}(u))du\right) = \tag{7.14}$$

$$\mathbb{E}\left[\sum_{k<k'}^{K}\mathbb{E}\left(\int_{\tau_k}^{\tau_{k+1}}(\beta(u)-\hat{\beta}(u))du\right)\mathbb{E}\left[\left(\int_{\tau'_k}^{\tau_{k'+1}}(\beta(u)-\hat{\beta}(u))du\right)|Z(\tau'_k)\right]\right] = 0 \tag{7.15}$$

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 $i \in I$*

$$\frac{d}{dt}\hat{T}_i(t) = \frac{\hat{Z}_i(t)}{\sum\limits_{j\in B_l}\hat{Z}_j(t)} \quad \textit{when} \quad \sum_{j\in B_l}\hat{Z}_j(t) > 0. \tag{7.16}$$

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 associated 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.

$$\epsilon := \min_{u\in[t-\delta,t+\delta]}\hat{Z}_i(u) > 0. \tag{7.17}$$

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,

$$T_i^{z_r}(|z_r|u_2) - T_i^{z_r}(|z_r|u_1) = \int\limits_{|z_r|u_1}^{|z_r|u_2} \beta_i^{z}(u)du = \int\limits_{|z_r|u_1}^{|z_r|u_2} \hat{\beta}_i^{z}(u)du + \xi_i^{z_r}(|z_r|u_2) - \xi_i^{z_r}(|z_r|u_1). \tag{7.18}$$

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

Hence

$$\hat{T}_i^{z_n}(u_2) - \hat{T}_i^{z_n}(u_1) = \int\limits_{u_1}^{u_2} \hat{\beta}(\hat{Z}^{z_r}(u))du \tag{7.19}$$

□

7.2 AMP estimator for discounted advantage function

In order to derive the corresponding AMP estimator for the discounted advantage function we define an infinite-horizon discounted state-action value function

$$Q_\eta^{(\gamma)}(x,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t g(x_t) \Big| x_0 = x\right] \text{ for each } (x,a) \in \mathcal{X} \times \mathcal{A},$$

where $x = x_0$, action $a = a_0$ is taken at time $k = 0$ and actions according to policy π_η are taken at each time $k \geq 1$.

Assume that an approximation $\tilde{V}$ of the infinite-horizon discounted value function V_η is available. Define the following martingale sequence $(M_\eta^{(n)} : n \geq 0),\ \eta \in \Theta$ where

$$M_\eta^{(n)}(x) := \sum_{k=1}^{n} \gamma^k [\tilde{V}(x_k) - \sum_{x' \in X} P_\eta(x'|x_{k-1})\tilde{V}(x')], \tag{7.20}$$

where $x = x_0$ and x_k is a state of the Markov chain P_η at time k.

By [23, Propositions 3, 7] if $\mu_\theta(V_\theta) < \infty$ then $M_\theta^{(n)} \to M^\infty$ a.s. and $\mathbb{E}M_\theta^{(\infty)} = 0$. The fact that $\mu_\eta^T V_\eta < \infty$ follows from $\mu_\eta^T V_\eta = \frac{\mu_\eta^T g}{1-\gamma}$, see [53, Section 10.4.].

The method does not add any bias subtracting M_∞ from $\hat{Q}_\eta^{(\gamma)}$. We get the AMP estimator of the discounted state-action value function:

$$\begin{aligned}\hat{Q}_\eta^{AMP(\tilde{V}),(\gamma)}(x) &:= \hat{Q}_\eta^{(\gamma)}(x,a) - M_\eta^{(\infty)}(x)\\ &= g(x_0) + \gamma \sum_{x' \in X} P_\eta(x'|x_0)\tilde{V}(x') + \sum_{k=1}^{\infty} \gamma^k \left(g(x_k) + \gamma \sum_{x' \in X} P_\eta(x'|x_k)\tilde{V}(x') - \tilde{V}(x_k) \right)\end{aligned}$$

where $x = x_0$, action $a = a_0$ is taken at time $k = 0$ and actions according to policy π_η are taken at each time $k \geq 1$.

We take $\tilde{V}$ as a baseline for the state-action value function and get the AMP estimator of the discounted advantage function:

$$\hat{A}^{AMP(\tilde{V}),(\gamma)}(x_k, a_k) := \sum_{k=0}^{\infty} \gamma^k \left(g(x_k) + \gamma \sum_{x' \in X} P_\eta(x'|x_k)\tilde{V}(x') - \tilde{V}(x_k) \right)$$

7.3 Proofs

Lemma 6. *For any vector ν on $\mathcal{X}$ we consider the following norms w.r.t. vector $\mathbb{V} : \mathcal{X} \to [1, \infty]$:*

$$||\nu||_{1,\mathbb{V}} := \sum_{x \in \mathcal{X}} |\nu(x)| \mathbb{V}(x)$$
$$||\nu||_{\infty,\mathbb{V}} := \sup_{x \in \mathcal{X}} \frac{|\nu(x)|}{\mathbb{V}(x)}$$

We also consider the following $\mathbb{V}$-weighted operator norm for a matrix $T \in \mathcal{M}_{\mathcal{X} \times \mathcal{X}}$:

$$||T||_{\mathbb{V}} := \sup_{x \in \mathcal{X}} \frac{1}{\mathbb{V}(x)} \sum_{y \in \mathcal{X}} |T(x,y)| \mathbb{V}(y).$$

The operator norm $|| \cdot ||_{\mathbb{V}}$ is equivalent to the norm induced from norm $|| \cdot ||_{1,\mathbb{V}}$ and to the norm induced from norm $|| \cdot ||_{\infty,\mathbb{V}}$, i.e. for any matrix $T \in \mathcal{M}_{\mathcal{X} \times \mathcal{X}}$:

$$||T||_{\mathbb{V}} = \sup_{\nu : ||\nu||_{1,\mathbb{V}} = 1} ||\nu T||_{1,\mathbb{V}} = \sup_{h : ||h||_{\infty,\mathbb{V}} = 1} ||Th||_{\infty,\mathbb{V}}$$

Proof. First, consider $\sup\limits_{h : ||h||_{1,\mathbb{V}} = 1} ||Th||_{1,\mathbb{V}}$ norm. On the one hand,

$$\sup_{\nu : ||\nu||_{1,\mathbb{V}} = 1} ||\nu T||_{1,\mathbb{V}} \geq \sum_{y \in X} \mathbb{V}(y) \left| \sum_{x \in X} \frac{T(x,y)}{\mathbb{V}(x)} \right| = ||T||_{\mathbb{V}}$$

On the other hand,

$$\begin{aligned}
\sup_{\nu : ||\nu||_{1,\mathbb{V}} = 1} ||\nu T||_{1,\mathbb{V}} &= \sup_{\nu : ||\nu||_{1,\mathbb{V}} = 1} \sum_{y \in X} \sum_{x \in X} |T(x,y)\nu(x)| \mathbb{V}(y) \\
&\leq \sup_{\nu : ||\nu||_{1,\mathbb{V}} = 1} \sum_{y \in X} \sum_{x \in X} \frac{1}{\mathbb{V}(x)} \; |T(x,y)| \; \mathbb{V}(y) \; |\nu(x)| \; \mathbb{V}(x) \\
&\leq \sup_{x \in X} \sum_{y \in X} \frac{1}{\mathbb{V}(x)} |T(x,y)| \mathbb{V}(y) \sup_{\nu : ||\nu||_{1,\mathbb{V}} = 1} \sum_{x \in X} |\nu(x)| \mathbb{V}(x) \\
&= ||T||_{\mathbb{V}}
\end{aligned}$$

Similar for $\sup\limits_{h : ||h||_{\infty,\mathbb{V}} = 1} ||Th||_{\infty,\mathbb{V}}$ norm:

$$\sup_{h : ||h||_{\infty,\mathbb{V}} = 1} ||Th||_{\infty,\mathbb{V}} \geq ||T\mathbb{V}||_{\infty,V} = ||T||_{\mathbb{V}}$$

and

$$
\begin{aligned}
\sup_{h:||h||_{\infty,\mathbb{V}}=1} ||Th||_{\infty,\mathbb{V}} &= \sup_{h:||h||_{\infty,\mathbb{V}}=1} \sup_{x\in X} \frac{1}{\mathbb{V}(x)} \sum_{y\in X} |T(x,y)h(y)| \\
&\leq \sup_{h:||h||_{\infty,\mathbb{V}}=1} \sup_{x\in X} \frac{1}{\mathbb{V}(x)} \sum_{y\in X} |T(x,y)|\mathbb{V}(y) \frac{|h(y)|}{\mathbb{V}(y)} \\
&\leq \sup_{x\in X} \frac{1}{\mathbb{V}(x)} \sum_{y\in X} |T(x,y)|\mathbb{V}(y) \sup_{h:||h||_{\infty,\mathbb{V}}=1} \sup_{y\in \mathcal{X}} \frac{|h(y)|}{\mathbb{V}(y)} \\
&= ||T||_{\mathbb{V}}
\end{aligned}
$$

□

Lemma 3. *Assume that drift condition (3.1) holds for P_η, $\eta \in \Theta$. Let some $\theta \in \Theta$ satisfies*

$$||[P_\theta - P_\eta]Z_\eta||_{\mathbb{V}} < 1$$

Then the Markov chain with transition matrix P_θ has a unique stationary distribution μ_θ.

Proof. We denote $U_{\theta,\eta} := [P_\theta - P_\eta]Z_\eta$ and define matrix $H_{\theta,\eta}$ as

$$H_{\theta,\eta} := \sum_{k=0}^{\infty} U_{\theta,\eta}^k \tag{7.21}$$

Convergence in $\mathbb{V}$-weighted norm in definition (7.21) follows from assumption $||U_{\theta,\eta}||_{\mathbb{V}} < 1$.

The goal of this proof is to show that the Markov chain has a unique stationary distribution μ_θ s.t.

$$\mu_\theta^T = \mu_\eta^T H_{\theta,\eta} \tag{7.22}$$

Let $\nu^T := \mu_\eta^T H_{\theta,\eta}$. We use $e = (1, 1, ..., 1, ..)^T$ to denote a unit vector.

We first verify that $\nu^T e = \sum_{x\in\mathcal{X}} \nu(x) = 1$. We note that $(I - P_\eta + \Pi_\eta)e = e - e + e = e$, hence $Z_\eta e = e$. Then

$$\nu^T e = \mu_\eta^T H_{\theta,\eta} e = \mu_\eta^T \sum_{k=0}^{\infty} [P_\theta - P_\eta]^k Z_\eta^k e = \mu_\eta^T I e = 1.$$

Next we verify that $\nu^T P_\theta = \nu^T$. Suppose that $P_\theta - P_\eta + U_{\theta,\eta} P_\eta = U_{\theta,\eta}$ then we get

$$\begin{aligned}
\nu^T P_\theta &= \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k P_\theta \\
&= \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k P_\theta - \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k P_\eta + \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k P_\eta \\
&= \mu_\eta^T + \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k P_\theta - \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k P_\eta + \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^{k+1} P_\eta \\
&= \mu_\eta^T + \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k [P_\theta - P_\eta + U_{\theta,\eta} P_\eta] \\
&= \mu_\eta^T + \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^{k+1} \\
&= \mu_\eta^T \sum_{k=0}^{\infty} U_{\theta,\eta}^k \\
&= \nu^T
\end{aligned}$$

It is left to show

$$P_\theta - P_\eta + U_{\theta,\eta} P_\eta = U_{\theta,\eta}. \tag{7.23}$$

Indeed,

$$\begin{aligned}
P_\theta - P_\eta + U_{\theta,\eta} P_\eta &= P_\theta - P_\eta + [P_\theta - P_\eta] Z_\eta P_\eta && \Big([P_\theta - P_\eta] Z_\eta\Big) P_\eta = [P_\theta - P_\eta]\Big(Z_\eta P_\eta\Big) \text{ by [29, Corollary 1.9]} \\
&= (P_\theta - P_\eta)(I + Z_\eta P_\eta) \\
&= (P_\theta - P_\eta)(I - \Pi_\eta + Z_\eta P_\eta) && (P_\theta - P_\eta)\Pi_\eta = 0 \\
&= (P_\theta - P_\eta)(I - Z_\eta \Pi_\eta + Z_\eta P_\eta) && Z_\eta \Pi_\eta = \Pi_\eta \\
&= (P_\theta - P_\eta) Z_\eta && I = Z_\eta(I + \Pi_\eta - P_\eta) = Z_\eta + Z_\eta \Pi_\eta - Z_\eta P_\eta \\
&= U_{\theta,\eta}
\end{aligned}$$

The uniqueness of the stationary distribution follows from the fact that the Markov chain with transition matrix P_θ is irreducible.

□

Theorem 1. *Suppose that the Markov chain with transition matrix P_η is an irreducible chain s.t. the drift condition (3.1) holds for some function $\mathbb{V} \geq 1$ and the cost function satisfies $|g| < \mathbb{V}$.*

For any $\theta \in \Theta$ s.t.

$$D_{\theta,\eta} := ||[P_\theta - P_\eta] Z_\eta||_{\mathbb{V}} < 1$$

the difference of long-run average costs of policies π_θ and π_η is bounded by:

$$\mu_\theta^T g - \mu_\eta^T g \;\leq\; M_1(\theta, \eta) + M_2(\theta, \eta), \tag{3.8}$$

where $M_1(\theta,\eta)$, $M_2(\theta,\eta)$ are finite and equal to

$$M_1(\theta,\eta) := \mu_\eta^T[g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta]$$
$$M_2(\theta,\eta) := \frac{D^2_{\theta,\eta}}{1-D_{\theta,\eta}} \left(1 + \frac{D_{\theta,\eta}}{(1-D_{\theta,\eta})}(\mu_\eta^T \mathbb{V})||I - \Pi_\eta + P_\eta||_{\mathbb{V}}||Z_\eta||_{\mathbb{V}}\right) ||g - (\mu_\eta^T g)e||_{\infty,\mathbb{V}} \, (\mu_\eta^T \mathbb{V})$$

Proof. We denote $U_{\theta,\eta} := [P_\theta - P_\eta]Z_\eta$. Under assumption $||U_{\theta,\eta}||_{\mathbb{V}} < 1$ operator $H_{\theta,\eta} := \sum\limits_{k=0}^{\infty} U^k_{\theta,\eta}$ is well-defined and $||H_{\theta,\eta}||_{\mathbb{V}} \leq \frac{1}{1-D_{\theta,\eta}}$, see the proof of Lemma 3.

The stationary distribution of the Markov chain with transition matrix P_θ can be represented as $\mu_\theta^T = \mu_\eta^T H_{\theta,\eta}$, see equation (7.22). We get $\mu_\theta^T \mathbb{V} = \mu_\eta H_{\theta,\eta}\mathbb{V} < \infty$.

By [38, Theorem 14.1.4] the fundamental matrix for the Markov chain with transition matrix P_θ

$$Z_\theta = \sum_{k=0}^{\infty}(P_\theta - \Pi_\theta)^k$$

is well-defined in $\mathbb{V}$-weighted norm sense.

Consider matrix $T \in \mathcal{M}_{\mathcal{X}\times\mathcal{X}}$ that has to satisfy two properties: 1) $||T||_V < \infty$ and 2) $(I - P_\theta + \Pi_\theta)T = T(I - P_\theta + \Pi_\theta) = I$. We note that if such a matrix exists its uniqueness follows from associativity of matrices with finite $\mathbb{V}-$weighted norm. It is easy to see that Z_θ satisfies both requirements.

The following equation has been proved for the finite state space in [51, Theorem 2], see also [19]:

$$Z_\theta = Z_\eta H_{\theta,\eta} - \Pi_\eta H_{\theta,\eta}[P_\theta - P_\eta]Z_\eta^2 H_{\theta,\eta}. \tag{7.24}$$

We have shown that all matrices in (7.24) are well-defined and one can directly follow the proof of [51, Theorem 2] to generalize the statement for the countable state space.

The long-run average costs difference is equal to

$$\begin{aligned}
\mu_\theta^T g - \mu_\eta^T g &= \mu_\theta^T g + \mu_\theta([P_\theta - I]h_\eta) - \eta_\eta & \mu_\theta([P_\theta - I]f) = 0\\
&= \mu_\theta(g + P_\theta h_\eta - h_\eta) - \eta_\eta\\
&= \mu_\eta(g - \mu_\eta^T ge + P_\theta h_\eta - h_\eta) + (\mu_\theta - \mu_\eta)(g + P_\theta h_\eta - h_\eta) & \pm\, \mu_\eta(g + P_\theta h_\eta - h_\eta)\\
&= \mu_\eta(g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta) + (\mu_\theta - \mu_\eta)(g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta) & (\mu_\theta - \mu_\eta)(\mu_\eta^T ge) = 0
\end{aligned}$$

For any vector ν on $\mathcal{X}$ we define the following norm w.r.t. vector $\mathbb{V} : \mathcal{X} \to [1, \infty]$:

$$||\nu||_{1,\mathbb{V}} := \sum_{x\in\mathcal{X}} |\nu(x)|\mathbb{V}(x)$$

We note that for any vectors ν_1, ν_2 on $\mathcal{X}$ and matrix $T \in \mathcal{M}_{\mathcal{X}\times\mathcal{X}}$ the following inequality holds, see Lemma 6:

$$||\nu_1^T T \nu_2||_{\mathbb{V}} \leq ||\nu_1||_{1,\mathbb{V}} ||T||_{\mathbb{V}} ||\nu_2||_{\infty,\mathbb{V}}$$

Now we are ready to bound the last term:

$$\begin{aligned}
&|(\mu_\theta - \mu_\eta)(g - \eta_\eta e + P_\theta h_\eta - h_\eta)| \\
&\qquad \le ||\mu_\theta - \mu_\eta||_{1,\mathbb{V}} \; ||g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta||_{\infty,\mathbb{V}} \\
&\qquad = ||\mu_\theta - \mu_\eta||_{1,\mathbb{V}} \; ||[P_\theta - P_\eta]h_\eta||_{\infty,\mathbb{V}} && g - (\mu_\eta^T g)e + P_\eta h_\eta - h_\eta \equiv 0 \\
&\qquad \le ||\mu_\theta - \mu_\eta||_{1,\mathbb{V}} \; ||[P_\theta - P_\eta]Z_\eta(g - (\mu_\eta^T g)e)||_{\infty,\mathbb{V}} && h_\eta = Z_\eta(g - \mu_\eta^T ge) \\
&\qquad \le ||\mu_\theta - \mu_\eta||_{1,\mathbb{V}} \; ||[P_\theta - P_\eta]Z_\eta||_{\mathbb{V}} \; ||(g - \mu_\eta^T ge)||_{\infty,\mathbb{V}} \\
&\qquad = D_{\theta,\eta}||\mu_\theta - \mu_\eta||_{1,\mathbb{V}} \; ||(g - (\mu_\eta^T g)e)||_{\infty,\mathbb{V}} && ||[P_\theta - P_\eta]Z_\eta||_{\mathbb{V}} = D_{\theta,\eta} \\
&\qquad = D_{\theta,\eta}||\mu_\eta[P_\eta - P_\theta]Z_\theta||_{1,\mathbb{V}} \; ||(g - \mu_\eta^T ge)||_{\infty,\mathbb{V}} && \mu_\eta - \mu_\theta = \mu_\eta[P_\eta - P_\theta]Z_\theta \text{ from (7.22)}
\end{aligned}$$

The only term that depends on θ is $||\mu_\eta[P_\eta - P_\theta]Z_\theta||_{1,V}$ which we bound next:

$$\begin{aligned}
||\mu_\eta[P_\eta - P_\theta]Z_\theta||_{1,\mathbb{V}} &= ||\mu_\eta[P_\eta - P_\theta]Z_\eta(I + \Pi_\eta - P_\eta)Z_\theta||_{1,\mathbb{V}} && Z_\eta(I + \Pi_\eta - P_\eta) = I \\
&\le \mu_\eta(\mathbb{V})D_{\theta,\eta}||(I + \Pi_\eta - P_\eta)Z_\theta||_{\mathbb{V}} && ||[P_\eta - P_\theta]Z_\eta||_{\mathbb{V}} = D_{\theta,\eta} \\
&= \mu_\eta(\mathbb{V})D_{\theta,\eta}||H_{\eta,\theta} - (I + \Pi_\eta - P_\eta)\Pi_\eta H_{\eta,\theta}[P_\theta - P_\eta]Z_\eta^2 H_{\eta,\theta}||_{\mathbb{V}} && \text{by (7.24)} \\
&\le \mu_\eta(\mathbb{V})D_{\theta,\eta}\left(\frac{1}{1 - D_{\theta,\eta}} + ||I + \Pi_\eta - P_\eta||_{\mathbb{V}}\mu_\eta(\mathbb{V})\frac{D_{\theta,\eta}}{(1 - D_{\theta,\eta})^2}||Z_\eta||_{\mathbb{V}}\right)
\end{aligned}$$

□

Lemma 4.

$$D_{\theta,\eta} \le ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \sum_{a\in A} |r_{\theta,\eta}(a|x) - 1| \, G_\eta(x,a),$$

where $G_\eta(x,a) := \frac{1}{\mathbb{V}(x)} \sum_{y\in X} \pi_\eta(a|x)P(y|x,a)\mathbb{V}(y)$.

Proof.

$$\begin{aligned}
||[P_\theta - P_\eta]Z_\eta||_{\mathbb{V}} &\le ||P_\theta - P_\eta||_{\mathbb{V}}||Z_\eta||_{\mathbb{V}} \\
&= ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \frac{1}{\mathbb{V}(x)} \sum_{y\in X} |P_\theta - P_\eta|_{x,y}\mathbb{V}(y) \\
&= ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \frac{1}{\mathbb{V}(x)} \sum_{y\in X} |\sum_{a\in A} P(y|x,a)\pi_\theta(a|x) - \sum_{a\in A} P(y|x,a)\pi_\eta(a|x)|\mathbb{V}(y) \\
&\le ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \frac{1}{\mathbb{V}(x)} \sum_{y\in X}\sum_{a\in A} P(y|x,a)|\pi_\theta(a|x) - \pi_\eta(a|x)|\mathbb{V}(y) \\
&= ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \sum_{a\in A} |\pi_\theta(a|x) - \pi_\eta(a|x)| \frac{\sum_{y\in X} P(y|x,a)\mathbb{V}(y)}{\mathbb{V}(x)} \\
&= ||Z_\eta||_{\mathbb{V}} \sup_{x\in X} \sum_{a\in A} \left|\frac{\pi_\theta(a|x)}{\pi_\eta(a|x)} - 1\right| G(x,a)
\end{aligned}$$

□

Lemma 7. *Consider a $\mathbb{V}$-uniformly ergodic Markov chain with transition matrix P. Let h be the fundamental solution of the Poisson's equation (3.2) for the chain with one-step cost function $g(\cdot)$, s.t. $|g(x)| < \mathbb{V}(x)$ for each $x \in \mathcal{X}$. Let $J_\gamma := \sum\limits_{t=0}^{\infty} \gamma^{t+1} P^t (g - \mu^T g)$ be the corresponding infinite-horizon discounted relative value function.*

Then for some constants $R < \infty$ and $r < 1$ we have

$$|J_\gamma(x) - h(x)| \leq \frac{R}{1-r}\frac{1-\gamma}{1-\gamma r}\mathbb{V}(x) \text{ for each } x \in \mathcal{X}$$

Proof. By [38, Theorem 15.4.1] there exist constants $R < \infty$ and $r < 1$ s.t. for any $x \in \mathcal{X}$ and $t \geq 0$:

$$\Big| \sum_{y \in \mathcal{X}} P^t(y|x) g(y) - \mu^T g \Big| \leq R\mathbb{V}(x) r^t.$$

Then

$$\begin{aligned}
|J_\gamma(x) - h(x)| &\leq \sum_{t=0}^{\infty} |\gamma^{t+1} - 1| \Big| \sum_{y \in \mathcal{X}} P^t(y|x)(g(y) - \mu^T g) \Big| \\
&\leq R\mathbb{V}(x) \sum_{t=0}^{\infty} (1 - \gamma^{t+1}) r^t \\
&= R\mathbb{V}(x) \frac{1-\gamma}{(1-r)(1-r\gamma)}
\end{aligned}$$

□

7.4 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 queuing 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 $\varsigma_l = [p^l_j : j \in B_l]$, where p^l_j 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.

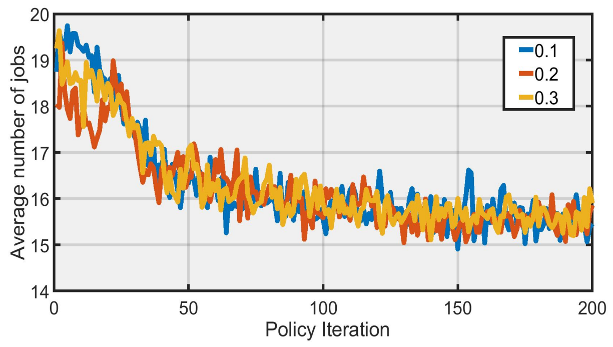
7.5 Experiment Parameters

In this section we explore learning process of the RL algorithm under different hyperparaments 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 4. We note that we use since the

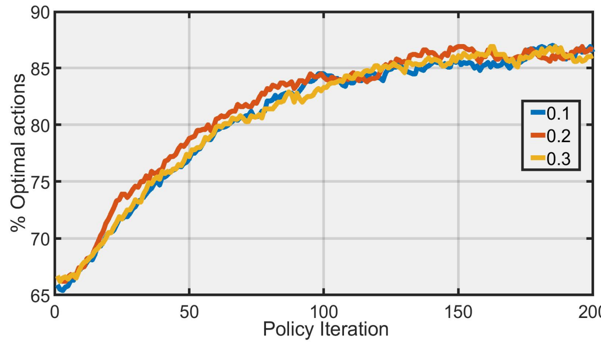
Parameter	Value
Clipping parameter (ϵ)	$0.2 \times \max[\alpha, 0.01]$
Horizon (T)	20,000
Num. of episodes/actors (N)	50
Adam stepsize for policy NN	$5 \cdot 10^{-4} \times \max[\alpha, 0.01]$
Adam stepsize for value NN	$2.5 \cdot 10^{-4}$
Discount factor (β)	0.998
GAE parameter (λ)	0.99
Num. of epochs in PPO loss optimization	3
Minibatch size in value NN optimization	256
Num. of epochs in value NN optimization	5
Routine to choose int. states	Algorithm 4

Table 4: 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 [50]. We have not identified any significant affect of the clipping parameter on learning and use $\epsilon = 0.2$ for the rest experiments.



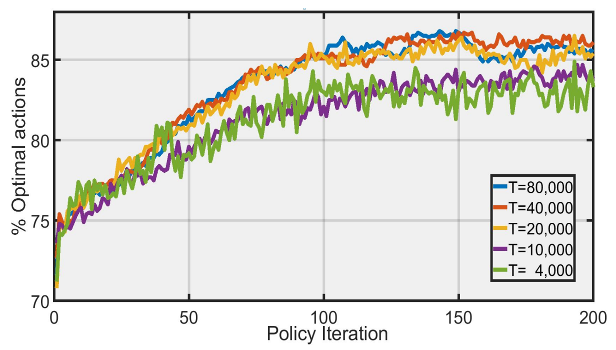
(a) average cost



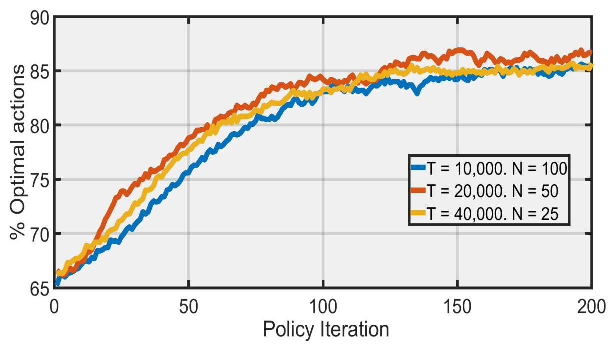
(b) Percentage of optimal actions

Figure 6: 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 7a 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 7b 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 7: 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.

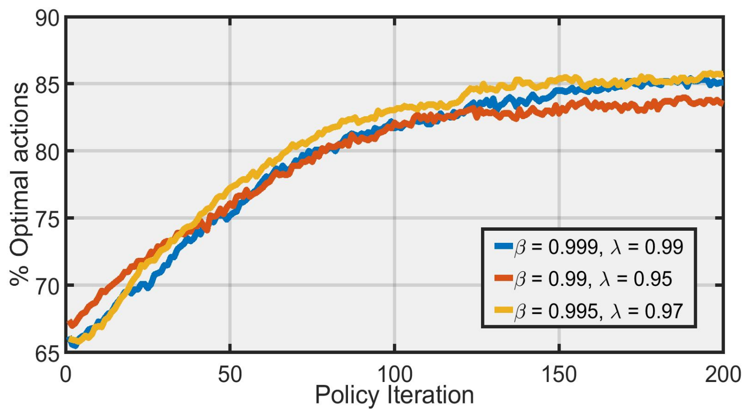
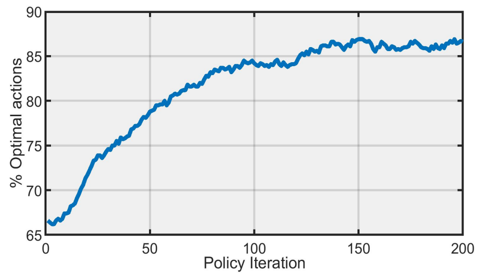
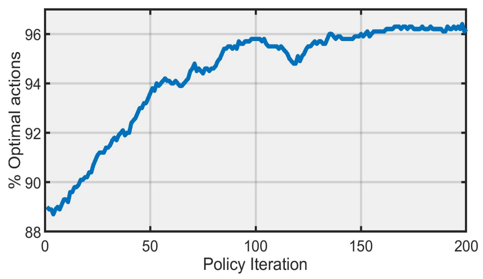


Figure 8: Effect of discount factor and GAE parameter.

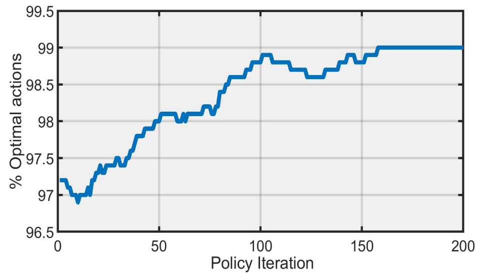
7.6 Learning Curves



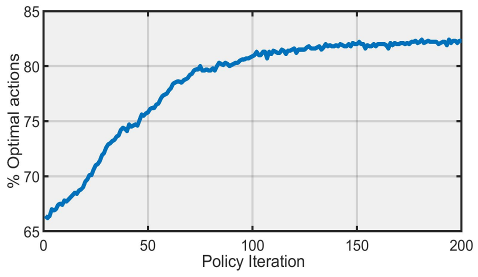
(a) Balanced heavy (BH) traffic



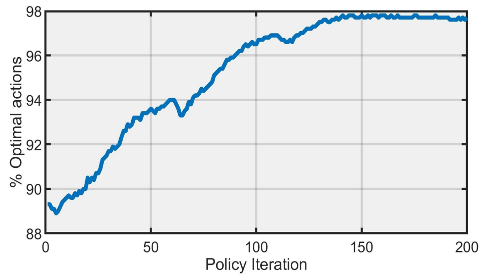
(b) Balanced medium (BM) traffic



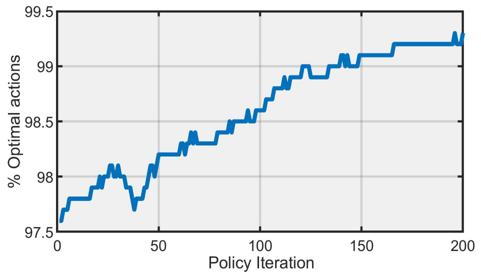
(c) Balanced low (BL) traffic



(d) Imbalanced heavy (IH) traffic

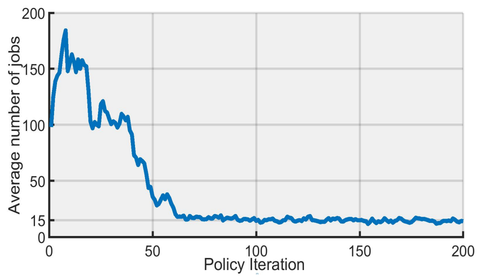


(e) Imbalanced medium (IM) traffic

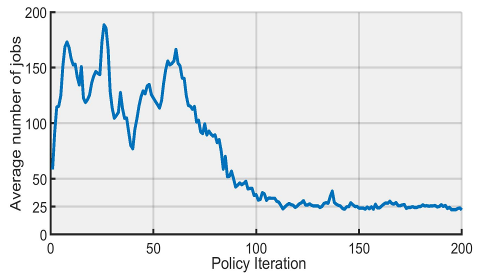


(f) Imbalanced low (IL) traffic

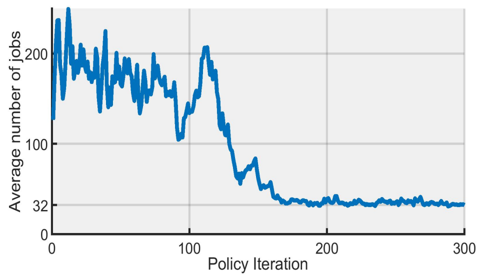
Figure 9: Learning of optimal actions. Results for the criss-cross network of Figure 1 under different traffic regimes.



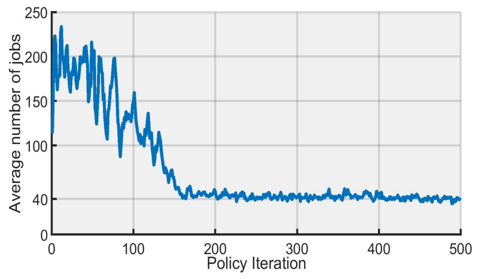
(a) 6-classes network



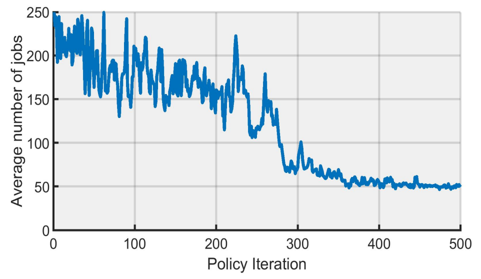
(b) 9-classes network



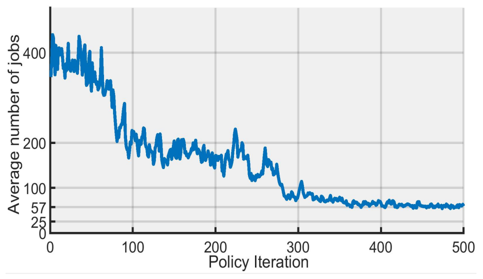
(c) 12-classes network



(d) 15-classes network

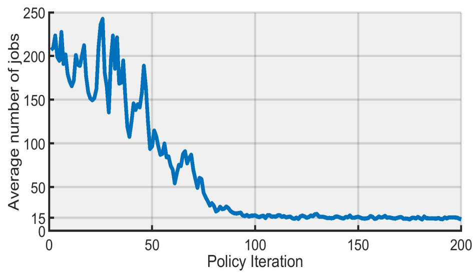


(e) 18-classes network

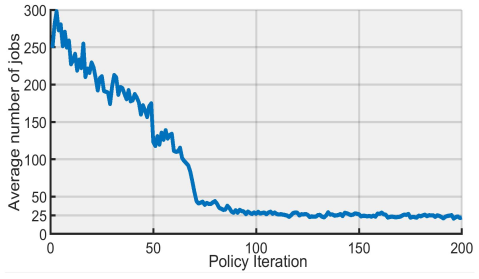


(f) 21-classes network

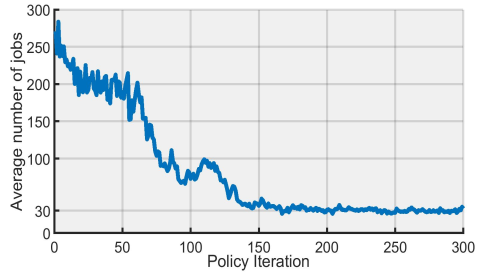
Figure 10: Learning curves for the 6-class extended network of Figure 4



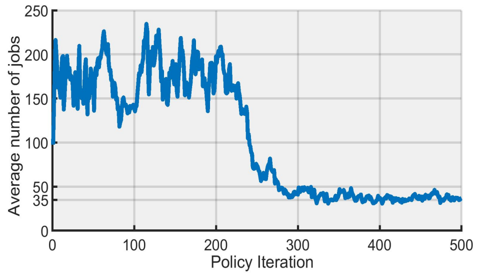
(a) 6-classes network



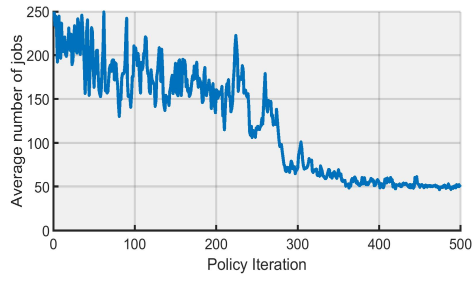
(b) 9-classes network



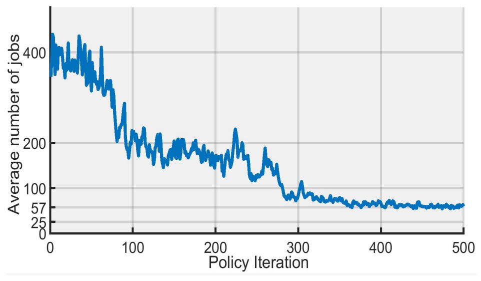
(c) 12-classes network



(d) 15-classes network



(e) 18-classes network



(f) 21-classes network

Figure 11: Learning curves for the reentrant extended network of Figure **??**

7.7 Choosing initial states for simulation

In the Algorithm 3 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.

Algorithm 4: Episodes are initialized from the states sampled from simulations conducted in the preceding policy iteration

```
   Result: policy $\pi_I$
1  Initialization: policy $\pi_0$, buffer $\mathbb{B} = \emptyset$;
2  Run policy $\pi_0$ for $T$ timesteps;
3  Sample uniformly at random $100 \times N$ states from the simulation of policy $\pi_0$;
4  Add $100 \times N$ states to $\mathbb{B}$;
5  for policy iteration $= 1, 2, ..., I$ do
6      Sample uniformly at random $N$ states $\{x^1, x^2, ..., x^N\}$ from $\mathbb{B}$;
7      for $s = 1, 2, ..., N$ do
8          if $\sum_{j=1}^{J} x_j^s > \Lambda$ then
9              $x^s = (0, ..., 0)$;
10         end
11     end
12     for actor $s = 1, 2, ..., N$ do
13         Initialize episode at state $x^s$;
14     end
15     Sample uniformly at random $100 \times N$ states from policy $\pi_i$ simulation;
16     Overwrite buffer $\mathbb{B}$ with the sampled states;
17 end
```

We propose sampling the initiate states from the set of states visited during preceding policy iteration, see Algorithm 1. 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. $\sum\limits_{j=1}^{J} x_j > \Lambda$ the corresponding episode starts from state $x = (0, ..., 0)$.

7.8 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 [40] 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 5: Running time of one policy iteration of the RL algorithm for the extended six-class network of Figure 4.

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] Søren Asmussen. *Applied Probability and Queues.* Springer New York, 2003.

[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] Maury Bramson. Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks. *Queueing Systems*, 23(1-4):1–26, mar 1996.

[10] X R Cao. Single Sample Path-Based Optimization of Markov Chains. *JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS*, 100(3):527–548, 1999.

[11] 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.

[12] William L. Cooper, Shane G. Henderson, and Mark E. Lewis. Convergence of simulation-based policy iteration. *Probability in the Engineering and Informational Sciences*, 17(2):213–234, 2003.

[13] 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.

[14] J. G. Dai and G. Weiss. Stability and Instability of Fluid Models for Reentrant Lines. *Mathematics of Operations Research*, 21(1):115–134, feb 1996.

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

[16] 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.

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

[18] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[19] Peter W. Glynn and Sean P. Meyn. A Liapounov bound for solutions of the Poisson equation. *Annals of Probability*, 24(2):916–931, 1996.

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

[21] 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.

[22] 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.

[23] Shane G. Henderson and Peter W. Glynn. Approximating martingales for variance reduction in Markov process simulation. *Mathematics of Operations Research*, 27(2):253–271, 2002.

[24] Shane G. Henderson and Sean P. Meyn. Efficient simulation of multiclass queueing networks. In *Proceedings of the 29th conference on Winter simulation - WSC '97*, pages 216–223, New York, New York, USA, 1997. ACM Press.

[25] Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, pages 345–352, 1994.

[26] Shuxia Jiang, Yuanyuan Liu, and Yingchun Tang. A unified perturbation analysis framework for countable Markov chains. *Linear Algebra and Its Applications*, 529:413–440, sep 2017.

[27] Sham Kakade. Optimizing Average Reward Using Discounted Rewards. pages 605–615. Springer, Berlin, Heidelberg, 2001.

[28] 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.

[29] John G. Kemeny, J. Laurie. Snell, and Anthony W. Knapp. *Denumerable Markov Chains.* Springer New York, 1976.

[30] 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.

[31] 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.

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

[33] Bai S.M. Liu. *Reinforcement learning in network control.* PhD thesis, Massachusetts Institute of Technology, 2019.

[34] 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.

[35] Peter Marbach and John N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, feb 2001.

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

[37] Sean Meyn. *Control techniques for complex networks.* Cambridge University Press, jan 2007.

[38] Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability.* Cambridge University Press, Cambridge, 2nd edition, 2009.

[39] 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.

[40] 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.

[41] Barry L. Nelson. Batch size effects on the efficiency of control variates in simulation. *European Journal of Operational Research*, 43(2):184–196, nov 1989.

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

[43] OpenAI. OpenAI Five, 2019.

[44] 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.

[45] James Gary Propp and David Bruce Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, aug 1996.

[46] 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.

[47] 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.

[48] 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.

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

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

[51] Paul J. Schweitzer. Perturbation theory and finite Markov chains. *Journal of Applied Probability*, 5(2):401–413, aug 1968.

[52] 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.

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

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

[55] 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.

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

[57] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample Efficient Actor-Critic with Experience Replay. nov 2016.

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

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

[60] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5285–5294, 2017.