Queueing Network Controls via Deep Reinforcement Learning

J. G. Dai[1,2] and Mark Gluzman [*,3]

[1] School of Data Science, Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen
[2]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY
[3]Center for Applied Mathematics, Cornell University, Ithaca, NY

Abstract

Novel advanced policy gradient (APG) methods, such as Trust Region policy optimization and Proximal policy optimization (PPO), have become the dominant reinforcement learning algorithms because of its ease of implementation and good practical performance. A conventional setup for notoriously difficult queueing network control problems is a Markov decision problem (MDP) that has three features: infinite state space, unbounded costs, and long-run average cost objective. We extend the theoretical framework of these APG methods for such MDP problems. The resulting PPO algorithm is tested on a parallel-server system and large-size multiclass queueing networks. The algorithm consistently generates control policies that outperform state-of-art heuristics in literature in a variety of load conditions from light to heavy traffic. These policies are demonstrated to be near-optimal when the optimal policy can be computed.

A key to the successes of our PPO algorithm is several variance reduction techniques in estimating the relative value function via sampling. First, we use a discounted relative value function as an approximation of the relative value function. Second, we propose regenerative simulation to estimate the discounted relative value function. Finally, we incorporate the approximating martingale-process method into the regenerative estimator.

Keywords: Multiclass Queueing Network, Control Variate, Reinforcement Learning, Long-Run Average Cost

1 Introduction

For more than 30 years, one of the most difficult problems in applied probability and operations research is to find a scalable algorithm for approximately solving the optimal control of stochastic processing networks, particularly when they are heavily loaded. These optimal control problems have many important applications including healthcare [22] and communications networks [75, 51], data centers [56, 53] and manufacturing systems [66, 46]. Stochastic processing networks are a broad class of models that were advanced in [31] and [32] and recently recapitulated and extended in [23].

In this paper, we demonstrate a class of *deep reinforcement learning* algorithms known as proximal policy optimization (PPO), generalized from [69, 71] to our setting, can generate control policies that consistently beat the performance of all state-of-arts control policies known in literature. The superior performances of our control policies appear to be robust as stochastic processing networks and their load conditions vary, with little or no problem-specific configurations of the algorithms.

Multiclass queueing networks (MQNs) are a special class of stochastic processing networks. They were introduced in [28] and have received intensive studies for more than 30 years for performance analysis and controls; see, for example, [33, 47, 13, 17, 81, 11, 12, 19, 37, 78]. Our paper focuses primarily on MQNs with long-run average cost objectives for two reasons. First, these stochastic control problems have been

*Corresponding author: Mark Gluzman, Center for Applied Mathematics, 657 Frank H.T. Rhodes Hall, Cornell University, Ithaca, NY 14853. Telephone: (607) 280-4554. Email: mg2289@cornell.edu

demonstrated to be notoriously difficult due to the size of the state space, particularly in heavy traffic. Second, there have been a large body of research, motivating various algorithms and control policies that are based on either heavy traffic asymptotic analysis or heuristics. See, for example, fluid policies [18], BIGSTEP policies [29], affine shift policies [57], discrete-review policies [52, 5], tracking policies [7], and "robust fluid" policies [12] in the former group and [50, 48] in the latter group. We demonstrate that our algorithms outperform the state-of-art algorithms in [12]. In this paper, we will also consider an N-model that belongs to the family of parallel-server systems, another special class of stochastic processing networks. Unlike a MQN in which routing probabilities of jobs are fixed, a parallel-server system allows dynamic routing of jobs in order to achieve load-balancing among service stations. We demonstrate that our algorithms achieves near optimal performance in this setting, again with little special configuration of our algorithms.

For the queueing networks with Poisson arrival and exponential service time distribution, the control problems can be modeled within the framework of Markov decision processes (MDPs) [67] via uniformization. A typical algorithm for solving an MDP is via *policy iteration* or *value iteration*. However, in our setting, the corresponding MDP suffers from the usual curse of dimensionality: there are a large number of job classes, and the buffer capacity for each class is assumed to be infinite. Even with a truncation of the buffer capacity either as a mathematical convenience or a practical control technique, the resulting state space is huge when the network is large and heavily loaded.

A *reinforcement learning (RL) problem* often refers to a (*model-free*) MDP problem in which the underlying model governing the dynamics is not known, but sequences of data (actions, states, and rewards), known as episodes in this paper, can be observed under a given policy. It has been demonstrated that *RL algorithms* for solving RL problems can successfully overcome the curve of dimensionality in both the model-free and model-based MDP problems. Two factors are the keys to the success in a model-based MDP. First, one uses Monte Carlo sampling method to approximately evaluate expectations. The sampling method also naturally supports explorations needed in RL algorithms. Second, one uses a parametric, low-dimensional representation of a value function and/or a policy. In recent years, RL algorithms that use neural networks as an architecture for value function and policy parametrizations have shown state-of-art results [10, 60, 74, 64].

In this paper we extend the Proximal Policy Optimization (PPO) algorithm to MDP problems with *unbounded cost* function and *long-run average cost* objective. The original PPO algorithm [71] was proposed for problems with *bounded cost* function and *infinite-horizon discounted* objective. It was based on the trust region policy optimization (TRPO) algorithm developed in [69]. We use two separate feedforward neural networks, one for parametrization of the control policy and the other for value function approximation, a setup common to *actor-critic* algorithms [60]. We propose approximating martingale-process (AMP) method for variance reduction to estimate policy value functions and show that the AMP estimation speed-ups convergence of PPO algorithm in the model-based setting. We provide guidance in the implementation details of PPO algorithm specific for MQNs such as the choice of initial stable randomized policy, the architecture of the value and policy neural networks and the choice of hyperparameters.

The actor-critic methods can be considered as a combination of approaches of value-based and policy based methods. In a *value-based* approximate dynamic programming (ADP) algorithm, one assumes a low-dimensional approximation of the optimal value function. For example, one assumes that the optimal value function is a linear combination of known *features*; see [25, 68, 2]. Value-based ADP algorithm has dominated in the stochastic control of queueing networks literature; see, for example, [19, 61, 20, 78]. These algorithms however have not achieved robust empirical success for a wide class of control problems. It is now known that the optimal value function might have a complex structure which is hard to decompose on features, especially if decisions have effect over a long horizon [49].

In a *policy-based* ADP algorithm, one aims to learn the optimal policy directly. Policy gradient algorithm is a type of policy-based methods that optimizes the objective value within a parameterized family policies via gradient descent; see, for example, [54, 65]. Policy gradient algorithm is particularly effective for problems with high-dimensional action space. However, a key challenge is how to reliably estimate the gradient of the value under current policy. A direct sample-based estimation typically suffers from high variance in gradient estimation. Actor-critic methods have been proposed [45] to overcome this issue: the value function is additionally estimated and is used as a baseline and bootstrap for gradient direction approximation. The actor-critic method with Boltzmann parametrization of policies and linear approximation of the value functions has been applied for parallel-server system control in [15]. The standard policy gradient methods typically perform one gradient update per data sample which yields

poor data efficiency and robustness. While an attempt to use a finite batch of samples to estimate the gradient and perform multiple steps of optimization "empirically ... leads to destructively large policy updates" [71]. In [71], the authors also note that the deep Q-learning algorithm [60] "fails on many simple problems".

In [69, 71], the authors propose *"advanced policy gradient"* methods to overcome the aforementioned problem by designing novel objective functions. Their proposed objective functions constrain the magnitude of policy update to avoid performance collapse caused by large changes in the policy. In [69] the authors prove that minimizing a certain surrogate objective function guarantees decreasing of expected discounted cost. Unfortunately, their theoretically-justified step-sizes of policy updates cannot be computed from available information for the RL algorithm. Trust Region Policy Optimization (TRPO) [69] has been proposed as a practical method to search for step-sizes of policy updates. Proximal Policy Optimization (PPO) method proposed [71] is an alternative way of computing these step-sizes based on a clipped, "proximal" objective function.

We summarize the major contributions of our study:

1. In Section 3 we provide a theoretical justification that the advanced policy gradient algorithms can be applied for long-run average cost 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 4.2 we discuss a new way of estimating relative value and advantage functions if transition probabilities are known. We adopt the approximating martingale-process method [35] which, to the best of our knowledge, has not been used in simulation-based approximate policy improvement setting.
3. In Section 4.3 we introduce a biased estimator of the relative value function through discounting the future costs. We interpret the discounting as the modification to the transition dynamics that shortens the regenerative cycles. We propose a regenerative estimator of the discounted relative value function.

   The discounting combined with the AMP method and regenerative simulation significantly reduces the variance of the relative value function estimation at the cost of a tolerable bias. The use of the proposed variance reduction techniques speeds up the learning process of the PPO algorithm that we demostrate by computational experiments in Section 5.1.
4. We provide extensive computational experiments in Section 5 for multiclass queueing networks and parallel servers systems. We propose to choose architectures of neural networks automatically as the size of a queueing network varies. We demonstrate the effectiveness of these choices as well as other hyperparameter choices such as learning rate used in gradient decent. We demonstrate that the performance of control policies resulting from the proposed PPO algorithm outperforms other heuristics. We empirically show how sample complexity of the processing network control optimization problem depends on the network load.

1.1 Notation

The set of real numbers is denoted by $\mathbb{R}$. The set of nonnegative integers is denoted by $\mathbb{Z}_+$. For a vector a and a matrix A, a^T and A^T denote their transposes.

2 Control of multiclass queueing networks

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

2.1 The criss-cross network

The criss-cross network has been studied in [34], [6], and [55] among others. The network is depicted in Figure 1, which is taken from Figure 1.2 of [23]. It consists of two stations that process three classes of jobs. Each job class has its own dedicated buffer where jobs wait to be processed.

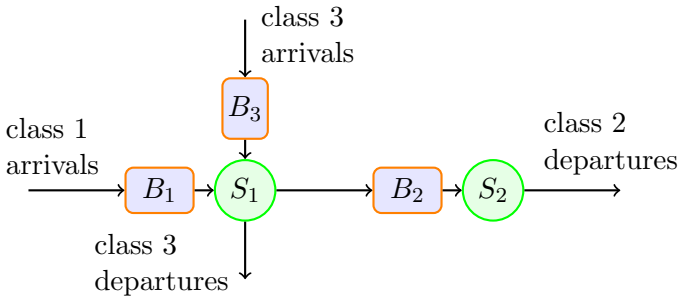


Figure 1: The criss-cross network

We assume that the jobs of class 1 and class 3 arrive to the system following Poisson processes with rates λ_1 and λ_3, respectively. Both classes are processed by server 1, one job at a time. After being processed class 1 jobs become class 2 jobs and wait in buffer 2 for server 2 to process. Class 2 and class 3 jobs leave the system after their processings are completed. Processing times for class j jobs are assumed to be i.i.d. having exponential distribution with mean m_j, $j = 1, 2, 3$. We denote $\mu_j := 1/m_j$ the service rate of class j jobs. We assume that the following load conditions are satisfied:

$$\lambda_1 m_1 + \lambda_3 m_3 < 1 \quad \text{and} \quad \lambda_1 m_2 < 1. \tag{2.1}$$

Again, we assume each server processes one job at a time. Therefore, processor sharing among multiple jobs is not allowed for each server. Jobs within a buffer are processed in the first-in–first-out order. A service policy dictates the order in which jobs from different classes are processed. (See below for a precise definition of a stationary Markov policy.) In this paper, we assume a decision time occurs when a new job arrives at the system or a service is completed. For concreteness, we adopt a preemptive service policy — suppose the service policy dictates that server 1 processes a class 3 job next while it is in the middle of processing a class 1 job, the server preempts the unfinished class 1 job to start the processing of the leading class 3 job from buffer 3. Due to the memoryless property of an exponential distribution, it does not matter whether the preempted job keeps its remaining processing time or is assigned a new service time sampled from the original exponential distribution.

Under any service policy, at each decision time, the system manager needs to simultaneously choose action a_1 from set $\{0, 1, 3\}$ for server 1 and action a_2 from the set $\{0, 2\}$ for server 2; for server 1 to choose action j, $j = 1, 3$, means that the server processes a class j job next (if buffer j is non-empty), and to choose action 0 means the server idles. Similarly, for server 2 to choose action 2 means that server 2 processes a class 2 job, and to choose action 0 means server 2 idles. Each server is allowed to choose action 0 even if there are waiting jobs at its station. Therefore, our service policies are not necessarily non-idling. We define the action set as $\mathcal{A} = \{(a_1, a_2) \in \{0, 1, 3\} \times \{0, 2\}\}$.

The service policy studied in this paper is *randomized*. By randomized, we mean each server takes a random action sampled from a certain distribution on the action set. For a set A, we use $\mathcal{P}(A)$ to denote the set of probability distributions on $\mathcal{A}$. Therefore, for a pair $(p_1, p_2) \in \mathcal{P}(\{0, 1, 3\}) \times \mathcal{P}(\{0, 2\})$, server 1 takes a random action sampled from distribution p_1 and server 2 takes a random action sampled from distribution p_2. For notational convenience, we note that such a pair has one-to-one correspondence to a vector u in the following set

$$\mathcal{U} = \left\{u = (u_1, u_2, u_3) \in \mathbb{R}^3_+ : u_1 + u_3 \leq 1 \text{ and } u_2 \leq 1\right\},$$

where $p_1 = (1 - u_1 - u_3, u_1, u_3) \in \mathcal{P}(\{0, 1, 3\})$ is a probability distribution on the action set $\{0, 1, 3\}$ and $p_2 = (1 - u_2, u_2) \in \mathcal{P}(\{0, 2\})$ is a probability distribution on the action set $\{0, 2\}$. Throughout this subsection, we use $u \in \mathcal{U}$ to denote such a pair (p_1, p_2).

To define a randomized stationary Markovian service policy, let $x_j(t)$ be the number of class j jobs (including possibly the one in service) in the system at time t, $j = 1, 2, 3$. Then $x(t) = (x_1(t), x_2(t), x_3(t))$ is the vector of jobcount at time t. Clearly, $x(t) \in \mathbb{Z}^3_+$. By convention, we assume the sample path of $\{x(t), t \geq 0\}$ is right continuous, which implies that when t is a decision time (triggered either an external arrival or a service completion), $x(t)$ has taken into account the arriving job or completed job at time t. By a randomized stationary Markovian service policy, we mean a map

$$\pi : \mathbb{Z}^3_+ \to \mathcal{U}.$$

Given this map π, at each decision time t, the system manager observes jobcounts $x = x(t)$, computes $\pi(x) \in \mathcal{U}$ and the corresponding pair $(p_1(x), p_2(x)) \in \mathcal{P}(\{0,1,3\}) \times \mathcal{P}(\{0,2\})$. Then server 1 chooses a random action sampled from $p_1(x)$ and server 2 chooses a random action sampled from $p_2(x)$. When each distribution is concentrated on a single action, the corresponding policy is a *deterministic* stationary Markovian service policy. In the following, we use the term stationary Markovian service policies to mean randomized policies, which include deterministic service policies as special cases.

Under a stationary Markovian policy π, $\{x(t), t \geq 0\}$ is a continuous time Markov chain (CTMC). From now on, we call jobcount vector $x(t)$ the state at time t, and we denote the state space as $\mathcal{X} = \mathbb{Z}_+^3$. The objective of our optimal control problem is to find a stationary Markovian policy that minimizes the long-run average number of jobs in the network:

$$\inf_{\pi} \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_\pi \int_0^T \Big(x_1(t) + x_2(t) + x_3(t)\Big) dt. \tag{2.2}$$

Because the interarrival and service times are exponentially distributed, the optimal control problem (2.2) fits the framework of semi-Markov decision process (SMDP). See, for example, [67, Chapter 11]. Indeed, one can easily verify that at each state x, taking action a, the distribution of the time interval until next decision time is exponential with rate $\beta(x,a)$ to be specified below. We use $P(y|x,a)$ to denote the transition probabilities of the embedded Markov decision process, where $y \in \mathcal{X}$ is a state at the next decision time. For any state $x \in \mathcal{X}$ and any action $a \in \mathcal{A}$, the following transition probabilities always hold

$$P\big((x_1+1, x_2, x_3)|x,a\big) = \frac{\lambda_1}{\beta(x,a)}, \quad P\big((x_1, x_2, x_3+1)|x,a\big) = \frac{\lambda_3}{\beta(x,a)} \tag{2.3}$$

In the following, we specify $\beta(x,a)$ for each $(x,a) \in \mathcal{X} \times \mathcal{A}$ and additional transition probabilities. For action $a = (1,2)$ and state $x = (x_1, x_2, x_2)$ with $x_1 \geq 1$ and $x_2 \geq 1$, $\beta(x,a) = \lambda_1 + \lambda_3 + \mu_1 + \mu_2$,

$$P\big((x_1-1, x_2+1, x_3)|x,a\big) = \frac{\mu_1}{\beta(x,a)}, \quad P\big((x_1, x_2-1, x_3)|x,a\big) = \frac{\mu_2}{\beta(x,a)};$$

for action $a = (3,2)$ and state $x = (x_1, x_2, x_2)$ with $x_3 \geq 1$ and $x_2 \geq 1$, $\beta(x,a) = \lambda_1 + \lambda_3 + \mu_3 + \mu_2$,

$$P\big((x_1, x_2, x_3-1)|x,a\big) = \frac{\mu_3}{\beta(x,a)}, \quad P\big((x_1, x_2-1, x_3)|x,a\big) = \frac{\mu_2}{\beta(x,a)};$$

for action $a = (0,2)$ and state $x = (x_1, x_2, x_3)$ with $x_2 \geq 1$, $\beta(x,a) = \lambda_1 + \lambda_3 + \mu_2$,

$$P\big((x_1, x_2-1, x_3)|x,a\big) = \frac{\mu_2}{\beta(x,a)};$$

for action $a = (1,0)$ and state $x = (x_1, x_2, x_3)$ with $x_1 \geq 1$, $\beta(x,a) = \lambda_1 + \lambda_3 + \mu_1$,

$$P\big((x_1-1, x_2+1, x_3)|x,a\big) = \frac{\mu_1}{\beta(x,a)};$$

for action $a = (3,0)$ and state $x = (x_1, x_2, x_3)$ with $x_3 \geq 1$, $\beta(x,a) = \lambda_1 + \lambda_3 + \mu_3$,

$$P\big((x_1, x_2, x_3-1)|x,a\big) = \frac{\mu_3}{\beta(x,a)};$$

for action $a = (0,0)$ and state $x = (x_1, x_2, x_3)$, $\beta(x,a) = \lambda_1 + \lambda_3$. In addition, $x_2 = 0$ implies that $a_2 = 0$, $x_1 = 0$ implies that $a_1 \neq 1$, $x_3 = 0$ implies that $a_1 \neq 3$.

Because the time between state transitions are exponentially distributed, we can and *will* adopt the method of uniformization for solving the SMDP; see, for example, [73] and [67, Chapter 11]. Denote

$$B = \lambda_1 + \lambda_3 + \mu_1 + \mu_2 + \mu_3. \tag{2.4}$$

For the new control problem under uniformization, the decision times are determined by the arrival times of a Poisson process with (uniform) rate B that is independent of the underlying state. Given current state $x \in \mathcal{X}$ and action $a \in \mathcal{A}$, new transition probabilities into $y \in \mathcal{X}$ are given by

$$\tilde{P}(y|x,a) = \begin{cases} P(y|x,a)\beta(x,a)/B & \text{if } x \neq y, \\ 1 - \beta(x,a)/B & \text{otherwise.} \end{cases} \tag{2.5}$$

The transition probabilities $\tilde{P}$ in (2.5) well define a (discrete time) MDP. The objective is given by

$$\inf_{\pi} \lim_{N\to\infty} \frac{1}{N}\mathbb{E}_{\pi}\left[\sum_{k=0}^{N-1}\left(x_1^{(k)}+x_2^{(k)}+x_3^{(k)}\right)\right], \tag{2.6}$$

where π belongs to the family of stationary Markov policies, and $x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$ is the state (jobcount) at the time of the kth decision (in the uniformized framework). Under a stationary Markov policy π, $\{x^{(k)} : k = 0, 1, 2, \ldots\}$ is a discrete time Markov chain (DTMC).

The existence of a stationary Markovian policy π^* that minimizes (2.6) follows from [57, Theorem 4.3] if the load conditions (2.1) are satisfied. Under a mild condition on π^*, which can be shown to be satisfied following the argument in [57, Theorem 4.3], the policy π^* is an optimal Markovian stationary policy for (2.2) [14, Theorem 2.1]. Moreover, under policy π^*, the objective in (2.2) is equal to that in (2.6); see [14, Theorem 3.6].

2.2 General formulation of multiclass queueing network control problem

Consider a multiclass queueing network that has L stations and J job classes. For notation convenience we denote $\mathcal{L} = \{1, ..., L\}$ as the set of stations, $\mathcal{J} = \{1, ..., J\}$ as the set of job classes.

Each station has a single server that processes jobs from the job classes that belong to the station. Each job class belongs to one station. We use $\ell = s(j) \in \mathcal{L}$ to denote the station that class j belongs to. We assume the function $s : \mathcal{J} \to \mathcal{L}$ satisfies $s(\mathcal{J}) = \mathcal{L}$. Jobs arrive externally to the network and are processed sequentially at various stations in the network, moving from one class to next after each processing step until they exit the network. If upon its arrival, a class j job finds the associated server busy, the job waits in the corresponding buffer j. For each station $\ell \in \mathcal{L}$, we define

$$\mathcal{B}(\ell) := \{j \in \mathcal{J} : \ s(j) = \ell\} \tag{2.7}$$

to be the set of job classes that are processed by server ℓ.

Class $j \in \mathcal{J}$ jobs arrive externally to buffer j following Poisson process with rate λ_j; when $\lambda_j = 0$, there are no external arrivals into buffer j. Class j jobs are processed by server $s(j)$ following a certain service policy to be specified below. The service times for class j jobs are assumed to be i.i.d having exponential distribution with mean $1/\mu_j$. Class j job, after being processed by server $s(j)$, becomes class $k \in \mathcal{J}$ job with probability r_{jk} and leaves the network with probability $1 - \sum_{k=1}^{K} r_{jk}$. We define $J \times J$ matrix $R := (r_{jk})_{j,k=1,...,J}$ as the routing matrix. We assume that the network is open, meaning that $I - R$ is invertible. Let vector $q = (q_1, q_2, ..., q_J)^T$ satisfy the following system of linear equations

$$q = \lambda + R^T q. \tag{2.8}$$

Equation (2.8) is known as the traffic equations, and under the open network assumption it has a unique solution. For each class $j \in \mathcal{J}$, q_j is interpreted to be the total arrival rate into buffer j, accouting to both the external arrivals and internal arrivals from service completions at stations. We define *the load* ρ_ℓ of station $\ell \in \mathcal{L}$ as

$$\rho_\ell = \sum_{j\in\mathcal{B}(\ell)} \frac{q_j}{\mu_j}.$$

We assume that

$$\rho_\ell < 1 \quad \text{for each station } \ell = 1, ..., L. \tag{2.9}$$

Let $x(t) = (x_1(t), .., x_J(t))$ be the vector of jobcount at time t. When a new job arrives at the system or a service is completed a decision time occurs. Under any service policy, at each decision time, the system manager needs to simultaneously choose an action for each server $\ell \in \mathcal{L}$. For each server $\ell \in \mathcal{L}$ the system manager selects an action from set $\mathcal{B}(\ell) \cup \{0\}$: action $j \in \mathcal{B}(\ell)$ means that the system manager gives priority to job class j at station ℓ; action 0 means server ℓ idles until next decision time.

We define set

$$\mathcal{U} = \left\{ u = (u_1, u_2, ..., u_J) \in \mathbb{R}_+^J : \sum_{j\in\mathcal{B}(\ell)} u_j \le 1 \text{ for each } \ell \in \mathcal{L} \right\}. \tag{2.10}$$

For each station $\ell \in \mathcal{L}$ vector $u \in \mathcal{U}$ defines a probability distribution u_ℓ on the action set $\mathcal{B}(\ell) \cup \{0\}$: probability of action j is equal to u_j for $j \in \mathcal{B}(\ell)$ and probability of action 0 is equal to $\left(1 - \sum_{j \in \mathcal{B}(\ell)} u_j\right)$. We define a randomized stationary Markovian service policy as a map from a set of jobcount vectors into set $\mathcal{U}$ defined in (2.10):

$$\pi : \mathbb{Z}_+^J \to \mathcal{U}.$$

Given this map π, at each decision time t, the system manager observes jobcounts $x(t)$, chooses $\pi(x) \in \mathcal{U}$ and based on $\pi(x)$ computes probability distribution u_ℓ for each $\ell \in \mathcal{L}$. Then the system manager independently samples one action from u_ℓ for each server $\ell \in \mathcal{L}$.

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

$$\inf_\pi \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_\pi \int_0^T \left(\sum_{j=1}^J x_j(t) \right) dt. \tag{2.11}$$

Under a stationary Markovian policy π, we adopt the method of uniformization to obtain a uniformized discrete time Markov chain (DTMC) $\{x^{(k)} : k = 0, 1, \ldots\}$. We abuse the notation and denote a system state as $x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, ..., x_J^{(k)}\right)$ after k transitions of the DTMC.

In this paper, we will develop algorithms to approximately solve the following (discrete-time) MDP problem:

$$\inf_\pi \lim_{N \to \infty} \frac{1}{N} \mathbb{E}_\pi \left[\sum_{k=0}^{N-1} \sum_{j=1}^J x_j^{(k)} \right]. \tag{2.12}$$

Remark 1. *It has been proved in [57] that the MDP (2.12) has an optimal policy that satisfies the conditions in [14, Theorem 3.6] if the "fluid limit model" under some policy is L_2-stable. Under the load condition (2.9), conditions in [57] can be verified as follows. First, we adopt the randomized version of the head-of-line static processor sharing (HLSPS) as defined in [23, Section 4.6]. We apply this policy to the discrete-time MDP to obtain the resulting DTMC. The fluid limit path of this DTMC can be shown to satisfy the fluid model defined in [23, Definition 8.17] following a procedure that is similar to, but much simpler than, the proof of [23, Theorem 12.24]. Finally, [23, Theorem 8.18] shows the fluid model is stable, which is stronger than the L_2-stability needed.*

3 Reinforcement learning approach for queueing network control

The policy gradient algorithms has originally been designed to find optimal policies which optimize the finite horizon total cost or infinite horizon discounted total cost objectives. For stochastic processing networks and their applications, it is often appropriate to optimize the long-run average cost. In this section we develop a version of Proximal Policy Optimization algorithm. In Section 5, we will demonstrate this version is effective in finding near optimal control policies for stochastic processing networks.

3.1 Positive recurrence and $\mathcal{V}$-uniform ergodicity

As discussed in Section 2, operating under a fixed randomized stationary control policy, the dynamics of a stochastic processing network is a DTMC. We restrict policies so that the resulting DTMC is irreducible and aperiodic. Such a DTMC does not always have a stationary distribution. When the DTMC does not have a stationary distribution, the performance of the control policy is necessarily poor, leading to an infinite long-run average cost. It is well known that an irreducible DTMC has a unique stationary distribution if and only if it is positive recurrent. Hereafter, when the DTMC is positive recurrent, we call the corresponding control policy *stable*. Otherwise, we call it *unstable*. Clearly, a good RL algorithm should keep each policy in the iteration *stable*.

A sufficient condition for an irreducible DTMC positive recurrent is the Foster-Lyapunov drift condition. The drift condition (3.1) in the following lemma is stronger than the classical Foster-Lyapunov drift condition. For a proof of the lemma, see for example, Theorem 11.3.4 and Theorem 14.3.7 of [59].

Lemma 1. *Consider an irreducible Markov chain on a countable state space $\mathcal{X}$ with a transition matrix P on $\mathcal{X} \times \mathcal{X}$. Assume that there exists a vector $\mathcal{V} : \mathcal{X} \to [1, \infty)$ such that the following drift condition holds for some constants $b \in (0, 1)$ and $d \geq 0$, and a finite subset $C \subset \mathcal{X}$:*

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

where $\mathbb{I}_C(x) = 1$ if $x \in C$ and $\mathbb{I}_C(x) = 0$ otherwise. Here, $P(y|x) = P(x, y)$ is the transition probability from state $x \in \mathcal{X}$ to state $y \in \mathcal{X}$. Then (a) the Markov chain with the transition matrix P is positive recurrent with a unique stationary distribution μ; (b) $\mu^T \mathcal{V} < \infty$, 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 $\mathcal{V}$ in the drift condition (3.1) is called a *Lyapunov function* for the Markov chain. For any matrix M on $\mathcal{X} \times \mathcal{X}$, its $\mathcal{V}$-norm is defined to be

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

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

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

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 $\mathcal{V}$-uniformly ergodic [59, Theorem 16.0.1]. For an irreducible, aperiodic Markov chain that satisfies (3.1), for any $g : \mathcal{X} \to \mathbb{R}$ with $|g(x)| \leq \mathcal{V}(x)$ for $x \in \mathcal{X}$, there exist constants $R < \infty$ and $r < 1$ such that

$$\left| \sum_{y \in \mathcal{X}} P^n(y|x) g(y) - \mu^T g \right| \leq R\mathcal{V}(x) r^n \tag{3.2}$$

for any $x \in \mathcal{X}$ and $n \geq 0$; see Theorem 15.4.1 of [59].

3.2 Poisson equation

For an irreducible DTMC on state space $\mathcal{X}$ (possibly infinite) with transition matrix P, assume that there exists a Lyapunov function $\mathcal{V}$ satisfying (3.1). For any cost function $g : \mathcal{X} \to \mathbb{R}$ satisfying $|g(x)| \leq \mathcal{V}(x)$ for each $x \in \mathcal{X}$, it follows from Lemma 1 that $\mu^T |g| < \infty$. Lemma 2 below asserts that the following equation has a solution $h : \mathcal{X} \to \mathbb{R}$:

$$g(x) - \mu^T g + \sum_{y \in \mathcal{X}} P(y|x) h(y) - h(x) = 0 \quad \text{for each } x \in \mathcal{X}. \tag{3.3}$$

Equation (3.3) is called *Poisson equation* of the Markov chain corresponding transition matrix P and cost function g. Function $h : \mathcal{X} \to \mathbb{R}$ that satisfies (3.3) is called a *solution* to the Poisson equation. The solution is unique up to a constant shift, namely, if h_1 and h_2 are two solutions to Poisson's equation (3.3) with $\mu^T(|h_1| + |h_2|) < \infty$, then there exists a constant $b \in \mathbb{R}$ such that $h_1(x) = h_2(x) + b$ for each $x \in \mathcal{X}$, see [59, Proposition 17.4.1].

A solution h to the Poisson equation is called a *fundamental solution* if $\mu^T h = 0$. The proof of the following lemma is provided in [58, Proposition A.3.11].

Lemma 2. *Consider a $\mathcal{V}$-uniformly ergodic Markov chain with transition matrix P and the stationary distribution μ. For any cost function $g : \mathcal{X} \to \mathbb{R}$ satisfying $|g| \leq \mathcal{V}$, Poisson equation (3.3) admits a fundamental solution*

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

where $x^{(k)}$ is the state of the Markov chain after k timesteps.

We define *fundamental matrix* Z of the Markov chain with transition kernel P as

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

It follows from [59, Theorem 16.1.2] that the series (3.5) converges in $\mathcal{V}$-norm and, moreover, $\|Z\|_{\mathcal{V}} < \infty$. Then, it is easy to see that fundamental matrix Z is an inverse matrix of $(I-P+\Pi)$, i.e. $Z(I-P+\Pi) = (I-P+\Pi)Z = I$.

Lemma 3. *Fundamental matrix Z maps any cost function $|g| \leq \mathcal{V}$ into a corresponding fundamental solution $h^{(f)}$ defined by (3.4):*

$$h^{(f)} = Z\left(g - (\mu^T g)e\right), \tag{3.6}$$

where $e = (1, 1, .., 1, ...)^T$ is a unit vector.

Remark 2. *Consider matrices A, B, C on $\mathcal{X} \times \mathcal{X}$. The associativity property,*

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

does not always hold for matrices defined on countable state space, see a counterexample in [43, Section 1.1]. However, if $\|A\|_{\mathcal{V}} < \infty, \|B\|_{\mathcal{V}} < \infty, \|C\|_{\mathcal{V}} < \infty$ then matrices A, B, C associate, see [40, Lemma 2.1]. Hence, there is no ambiguity in the definition of the fundamental matrix (3.5):

$$(P-\Pi)^k = (P-\Pi)(P-\Pi)^{k-1} = (P-\Pi)^{k-1}(P-\Pi), \quad \text{for } k \geq 1$$

where $\|P - \Pi\|_{\mathcal{V}} < \infty$ is held due to the drift condition (3.1).

3.3 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 $\Theta \subset \mathbb{R}^d$ for some integer $d > 0$ and Θ is open. 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 $\pi_\theta(a|x)$. Under the policy π_θ, the corresponding DTMC has transition matrix P_θ given by

$$P_\theta(x,y) = \sum_{a \in \mathcal{A}} \pi(a|x) P(y|x,a) \text{ for } x, y \in \mathcal{X}.$$

We assume that for each $\theta \in \Theta$ the resulting Markov chains with transition probabilities P_θ is irreducible and aperiodic.

We also assume that there exists $\eta \in \Theta$ such that the drift condition (3.1) is satisfied for the transition matrix P_η with a Lyapunov function $\mathcal{V} : \mathcal{X} \to [1, \infty)$. By Lemma 2 the corresponding fundamental matrix Z_η is well-defined. The following lemma says that if P_η is positive recurrent and P_θ is "close" to P_η, then P_θ is also positive recurrent. Its proof can be found in Section A.

Lemma 4. *Fix an $\eta \in \Theta$. Assume that drift condition (3.1) holds for P_η. Let some $\theta \in \Theta$ satisfies*

$$\|(P_\theta - P_\eta) Z_\eta\|_{\mathcal{V}} < 1$$

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

Assume the assumption of Lemma 4 holds. For any cost function $|g| \leq V$, we denote the corresponding fundamental solution to the Poisson equation as h_η and the long-run average cost

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

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 A.

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

For any $\theta \in \Theta$ such that

$$D_{\theta,\eta} := \|(P_\theta - P_\eta)Z_\eta\|_{\mathcal{V}} < 1 \tag{3.8}$$

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

$$\mu_\theta^T g - \mu_\eta^T g \;\leq\; N_1(\theta,\eta) + N_2(\theta,\eta), \tag{3.9}$$

where $N_1(\theta,\eta)$, $N_2(\theta,\eta)$ are finite and equal to

$$N_1(\theta,\eta) := \mu_\eta^T(g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta), \tag{3.10}$$

$$N_2(\theta,\eta) := \frac{D_{\theta,\eta}^2}{1 - D_{\theta,\eta}} \left(1 + \frac{D_{\theta,\eta}}{(1 - D_{\theta,\eta})}(\mu_\eta^T \mathcal{V})\|I - \Pi_\eta + P_\eta\|_{\mathcal{V}}\|Z_\eta\|_{\mathcal{V}}\right) \left\|g - (\mu_\eta^T g)e\right\|_{\infty,\mathcal{V}} (\mu_\eta^T \mathcal{V}), \tag{3.11}$$

where, for a vector ν on $\mathcal{X}$, we define $\mathcal{V}$-norm

$$\|\nu\|_{\infty,\mathcal{V}} := \sup_{x\in\mathcal{X}} \frac{|\nu(x)|}{\mathcal{V}(x)}. \tag{3.12}$$

It follows from Theorem 1 that the negativity of the RHS of inequality (3.9) guarantees the policy π_θ yields an improved performance comparing with the initial policy π_η. Since

$$\min_{\theta\in\Theta:\ D_{\theta,\eta}<1} [N_1(\theta,\eta) + N_2(\theta,\eta)] \leq N_1(\eta,\eta) + N_2(\eta,\eta) = 0,$$

clearly, one should pick $\theta = \theta^*$:

$$\theta^* = \text{argmin}_{\theta\in\Theta:\ D_{\theta,\eta}<1} [N_1(\theta,\eta) + N_2(\theta,\eta)] \tag{3.13}$$

to achieve the maximum improvement in the upper bound. The optimization problem (3.13) is challenging. In the setting of finite horizon and infinite discounted RL problems, [42, 69] propose to fix the maximum change between policies π_θ and π_η by bounding $N_2(\theta,\eta)$ term and focus on minimizing $N_1(\theta,\eta)$. In the rest of this subsection, we motivate our version of PPO algorithm to be presented in the next subsection, leading to a practical algorithm to approximately solve optimization (3.13).

First observe that

$$\begin{aligned}
|N_1(\theta,\eta)| :&= \left|\mu_\eta^T(g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta)\right| \\
&\leq (\mu_\eta^T \mathcal{V})\|g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta\|_{\infty,\mathcal{V}} \\
&= (\mu_\eta^T \mathcal{V})\|(P_\theta - P_\eta)h_\eta\|_{\infty,\mathcal{V}} \\
&= (\mu_\eta^T \mathcal{V})\|(P_\theta - P_\eta)Z_\eta\left(g - (\mu_\eta^T g)e\right)\|_{\infty,\mathcal{V}} \\
&\leq (\mu_\eta^T \mathcal{V})\|g - (\mu_\eta^T g)e\|_{\infty,\mathcal{V}} D_{\theta,\eta}.
\end{aligned}$$

Therefore, when θ is close to η, one expects that $D_{\theta,\eta}$ in (3.8) is small and $N_1(\theta,\eta) = O(D_{\theta,\eta})$.

From (3.11), $N_2(\theta,\eta)$ is nonnegative and $N_2(\theta,\eta) = O(D_{\theta,\eta}^2)$. Therefore, when $N_1(\theta,\eta) < 0$ and $D_{\theta,\eta}$ is small enough, π_θ is a strict improvement over π_η. The next lemma shows that the distance $D_{\theta,\eta}$ can be controlled by the probability ratio

$$r_{\theta,\eta}(a|x) := \frac{\pi_\theta(a|x)}{\pi_\eta(a|x)} \tag{3.14}$$

between the two policies.

Lemma 5.

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

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

The proof of the lemma can be found in Appendix in Section A. Lemma 5 implies that $D_{\theta,\eta}$ is small when the ration $r_{\theta,\eta}(a|x)$ in (3.14) 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 $\theta = \eta$.

3.4 Proximal Policy Optimization

The first term on the RHS of (3.9) can be rewritten as

$$\begin{aligned} N_1(\theta,\eta) &= \mu_\eta^T(g-(\mu_\eta^T g)e + P_\theta h_\eta - h_\eta) \\ &= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\theta(\cdot|x) \\ y\sim P(\cdot|x,a)}} \left[g(x)-(\mu_\eta^T g)e + h_\eta(y) - h_\eta(x)\right] \\ &= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\theta(\cdot|x)}} A_\eta(x,a) \\ &= \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} \left[\frac{\pi_\theta(a|x)}{\pi_\eta(a|x)} A_\eta(x,a)\right] = \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} \left[r_{\theta,\eta}(a|x)A_\eta(x,a)\right], \end{aligned} \tag{3.15}$$

where we define an advantage function $A_\eta : \mathcal{X}\times\mathcal{A}\to\mathbb{R}$ of policy π_η, $\eta\in\Theta$ as

$$A_\eta(x,a) := \mathop{\mathbb{E}}_{y\sim P(\cdot|x,a)} \left[g(x) - \mu_\eta^T g + h_\eta(y) - h_\eta(x)\right]. \tag{3.16}$$

Equation (3.15) implies that if one wants to minimize $N_1(\theta,\eta)$ then the ratio $r_{\theta,\eta}(a|x)$ should be minimized (w.r.t. θ) when $A_\eta(x,a) > 0$, and maximized when $A_\eta(x,a) < 0$ for each $x\in\mathcal{X}$. The discussion towards the end of Section 3.3 suggests that we should strive for two goals: (a)

$$\text{minimize} \quad N_1(\theta,\eta) \tag{3.17}$$

and (b) keep the ratio $r_{\theta,\eta}(a|x)$ in (3.14) close to 1. In [71] the authors propose to minimized (w.r.t. $\theta\in\Theta$) the following clipped surrogate objective

$$L(\theta,\eta) := \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\eta(\cdot|x)}} \max\left[r_{\theta,\eta}(a|x)A_\eta(x,a),\ \text{clip}(r_{\theta,\eta}(a|x),1-\epsilon,1+\epsilon)A_\eta(x,a)\right], \tag{3.18}$$

where

$$\text{clip}(c,1-\epsilon,1+\epsilon) := \begin{cases} 1-\epsilon, & \text{if } c<1-\epsilon, \\ 1+\epsilon, & \text{if } c>1+\epsilon, \\ c, & \text{otherwise,} \end{cases}$$

and $\epsilon\in(0,1)$ is a hyperparameter. In [71] the authors have coined the term proximal policy optimization (PPO) for their algorithm, and demonstrated the effectiveness of the PPO algorithm in terms of the simplicity in implementation and the ability of finding good control policies.

The objective term $\text{clip}(r_{\theta,\eta}(a|x),1-\epsilon,1+\epsilon)A_\eta(x,a)$ in (3.18) prevents changes to the policy that move $r_{\theta,\eta}(a|x)$ far from 1. Then the objective function (3.18) is a upper bound (i.e., a pessimistic bound) on the unclipped objective (3.17). Thus, an improvement on the objective (3.18) translates to an improvement on $N_1(\theta,\eta)$ only when $\theta\in\Theta$ satisfies $r_{\theta,\eta}\in(1-\epsilon,1+\epsilon)$. Several alternative heuristics have been proposed in the literature (see, [69, 80, 71, 82]) to solve optimization problem (3.13). Each of these heuristics defines a loss function that controls $N_2(\theta,\eta)$ and minimizes the $N_1(\theta,\eta)$ term. Following [71], we use loss function (3.18) in our study because of its implementation simplicity.

To compute objective function in (3.18) one needs to evaluate the expectation and precompute advantage functions in (3.18). First, let us assume that that an approximation $\hat{A}_\eta : \mathcal{X}\times\mathcal{A}\to\mathbb{R}$ of the advantage function (3.16) is available and focus on the estimation the objective from simulations. We discuss a procedure to estimate $\hat{A}_\eta$ in Section 4 below.

Given an episode with length N generated under policy π_η one can compute the advantage function estimates $\hat{A}_\eta\left(x^{(k,q)},a^{(k,q)}\right)$ at the observed state-action pairs:

$$D^{(0:N-1)} := \left\{\left(x^{(0)},a^{(0)},\hat{A}_\eta(x^{(0)},a^{(0)})\right),\left(x^{(1)},a^{(1)},\hat{A}_\eta(x^{(1)},a^{(1)})\right),\cdots,\left(x^{(N-1)},a^{(N-1)},\hat{A}_\eta(x^{(N-1)},a^{(N-1)})\right)\right\}.$$

The loss function (3.18) is estimated as a sample average over the state-action pairs from the episode:

$$\hat{L}\left(\theta,\eta,D^{(0:N-1)}\right) = \sum_{k=0}^{N-1} \max\left[\frac{\pi_\theta\left(a^{(k)}|x^{(k)}\right)}{\pi_\eta\left(a^{(k)}|x^{(k)}\right)}\hat{A}_\eta\left(x^{(k)},a^{(k)}\right), \text{clip}\left(\frac{\pi_\theta\left(a^{(k)}|x^{(k)}\right)}{\pi_\eta\left(a^{(k)}|x^{(k)}\right)},1-\epsilon,1+\epsilon\right)\hat{A}_\eta\left(x^{(k)},a^{(k)}\right)\right]. \tag{3.19}$$

In theory, one long episode under policy π_η starting from any initial state $x^{(0)}$ is sufficient because the following SLLN for Markov chains holds: with probability 1,

$$\lim_{N\to\infty} \frac{1}{N}\hat{L}\left(\theta, \eta, D^{(0:N-1)}\right) = L(\theta, \eta).$$

4 Advantage function estimation

Computation of objective function (3.18) relies on the availability of an estimate of advantage function $A_\eta(x, a)$ in (3.16). We assume our MDP model is known. So the expectation on (3.16) can be computed exactly. Furthermore, we assume this expectation computation can be done in a time-efficient manner. Therefore, our focus is on how to estimate h_η, a solution to the Poisson equation (3.3) with $P = P_\eta$ and $\mu = \mu_\eta$.

To compute expectation $A\left(x^{(k)}, a^{(k)}\right)$ in (3.16) for a given state-action pair $\left(x^{(k)}, a^{(k)}\right)$, one needs to evaluate $h_\eta(y)$ for each y that is reachable from $x^{(k)}$. This requires one to estimate $h_\eta(y)$ for some states y that have not been visited in the simulation. Our strategy is to use Monte Carlo method to estimate $h_\eta(y)$ at a selected subset of y's, and then use an approximator $f_\psi(y)$ to replace $h_\eta(y)$ for an arbitrary $y \in \mathcal{X}$. The latter is standard in deep learning. Therefore, we focus on finding a good estimator $\hat{h}(y)$ for $h_\eta(y)$.

4.1 Regenerative estimation

Lemma 2 provides a representation of the fundamental solution (3.4) for h_η. Unfortunately, the known unbiased Monte-Carlo estimators of the fundamental solution relies on obtaining samples from the stationary distribution of the Markov chain [21, Section 5.1].

We define the following solution to the Poisson's equation (3.3).

Lemma 6. *Consider $\mathcal{V}$-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. $|g| \le \mathcal{V}$, the Poisson's equation (3.3) admits a solution*

$$h^{(x^*)}(x) := \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \left(g(x^{(k)}) - \mu^T g\right) \,\middle|\, x^{(0)} = x\right] \quad \text{for each } x \in \mathcal{X}, \tag{4.1}$$

where $\sigma(x^*) = \min\left\{k > 0 \mid x^{(k)} = x^*\right\}$ is the first future time when state x^* is visited. Furthermore, the solution has a finite $\mathcal{V}$-norm: $\|h^{(x^*)}\|_{\infty,\mathcal{V}} < \infty$.

The proof of Lemma 6 is given in [58, 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*.

The value of advantage function (3.16) does not depend on a particular choice of a solution of the Poisson's equation (3.3) since if h_1 and h_2 are two solutions to Poisson's equation (3.3) with $\mu^T(|h_1| + |h_2|) < \infty$, then there exists a constant $b \in \mathbb{R}$ such that $h_1(x) = h_2(x) + b$ for each $x \in \mathcal{X}$, see [59, Proposition 17.4.1]. Therefore, we use representation (4.1) for h_η in computation of (3.16).

Assume that an episode consisting of N regenerative cycles

$$\left\{x^{(0)}, x^{(1)}, \cdots, x^{(\sigma_1)}, \cdots, x^{(\sigma_N - 1)}\right\}$$

have been generated under policy π_η, where $x^{(0)} = x^*$. We compute an estimate of the long-run average cost based on N regenerative cycles as

$$\widehat{\mu_\eta^T g} := \frac{1}{\sigma(N)} \sum_{k=0}^{\sigma(N)-1} g(x^{(k)}), \tag{4.2}$$

where $\sigma(n)$ is the nth time when regeneration state x^* is visited. Consider an arbitrary state $x^{(k)}$ from the generated episode. One-replication estimate of solution to Poisson's equation (4.1) for a state $x^{(k)}$ visited at time k is defined as

$$\hat{h}_k := \sum_{t=k}^{\sigma_k - 1} \left(g(x^{(t)}) - \widehat{\mu_\eta^T g}\right), \tag{4.3}$$

where $\sigma_k = \min\left\{t > k \mid x^{(t)} = x^*\right\}$ is the first time when the regeneration state x^* is visited after time k. We note that the one-replication estimate (4.3) is computed every timestep. The estimator (4.3) has been proposed in [21, Section 5.3].

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-replication estimates $\{\hat{h}_k\}_{k=0}^{\sigma(N)-1}$:

$$\psi^* = \arg\min_{\psi\in\Psi} \sum_{k=0}^{\sigma(N)-1} \left(f_\psi(x^{(k)}) - \hat{h}_k\right)^2. \tag{4.4}$$

With available function approximation f_{ψ^*} for h_η, the advantage function (3.16) can be estimated as

$$\hat{A}_\eta(x^{(k)}, a^{(k)}) := g(x^{(k)}) - \widehat{\mu_\eta^T g} + \sum_{y\in\mathcal{X}} P\left(y|x^{(k)}, a^{(k)}\right) f_{\psi^*}(y) - f_{\psi^*}(x^{(k)}). \tag{4.5}$$

Given an episode of N regenerative cycles generated under policy π_η one can compute the advantage function estimates $\hat{A}_\eta(x^{(k,q)}, a^{(k,q)})$ by (4.5):

$$D^{(0:\sigma(N)-1)} := \Big\{ \left(x^{(0)}, a^{(0)}, \hat{A}_\eta(x^{(0)}, a^{(0)})\right), \left(x^{(1)}, a^{(1)}, \hat{A}_\eta(x^{(1)}, a^{(1)})\right), \cdots, \\ \left(x^{(\sigma(N)-1)}, a^{(\sigma(N)-1)}, \hat{A}_\eta(x^{(\sigma(N)-1)}, a^{(\sigma(N)-1)})\right) \Big\}.$$

The loss function (3.18) is estimated as a sample average over the cycles:

$$\hat{L}\left(\theta, \eta, D^{(0:\sigma(N)-1)}\right) = \sum_{k=0}^{\sigma(N)-1} \max\Big[\frac{\pi_\theta\left(a^{(k)}|x^{(k)}\right)}{\pi_\eta\left(a^{(k)}|x^{(k)}\right)} \hat{A}_\eta(x^{(k)}, a^{(k)}), \\ \text{clip}\left(\frac{\pi_\theta\left(a^{(k)}|x^{(k)}\right)}{\pi_\eta\left(a^{(k)}|x^{(k)}\right)}, 1-\epsilon, 1+\epsilon\right) \hat{A}_\eta(x^{(k)}, a^{(k)})\Big]. \tag{4.6}$$

If Q episodes can be simulated in parallel, each of $q = 1, .., Q$ (parallel) actors collect an episode

$$\left\{x^{(0,q)}, a^{(0,q)}, x^{(1,q)}, a^{(1,q)}, \cdots, x^{(k,q)}, a^{(k,q)}, \cdots, x^{(\sigma^q(N)-1,q)}, a^{(\sigma^q(N)-1,q)}\right\}$$

with N regenerative cycles, where $\sigma^q(N)$ 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 $\sum_{q=1}^{Q} \sigma^q(N)$ data-points:

$$D^{(0:\sigma^q(N)-1)_{q=1}^Q} = \left\{\left(x^{(0,q)}, a^{(0,q)}, \hat{A}_{0,q}\right), \cdots, \left(x^{(\sigma^q(N)-1,q)}, a^{(\sigma^q(N)-1,q)}, \hat{A}_{\sigma^q(N)-1,q}\right)\right\}_{q=1}^{Q}.$$

Optimization of the loss function yields a new policy for the next iteration.

Algorithm 1: Base proximal policy optimization algorithm for long-run average cost problems

Result: policy π_{θ_I}

Initialize policy π_{θ_0} ;

for *policy iteration* $i = 0, 1, ..., I-1$ **do**

  for *actor* $q = 1, 2, ..., Q$ **do**

    Run policy π_{θ_i} until it reaches Nth regeneration time on $\sigma^q(N)$ step: collect an episode $\left\{x^{(0,q)}, a^{(0,q)}, x^{(1,q)}, a^{(1,q)}, \cdots, x^{(\sigma^q(N)-1,q)}, a^{(\sigma^q(N)-1,q)}, x^{(\sigma^q(N),q)}\right\}$;

  end

  Compute average cost estimate $\widehat{\mu_{\theta_i}^T g}$ by (4.2) (utilizing Q episodes) ;

  Compute $\hat{h}_{k,q}$, estimate of $h_{\theta_i}(x^{(k,q)})$, by (4.3) for each $q = 1, .., Q$, $k = 0, .., \sigma^q(N) - 1$;

  Update $\psi_i := \psi$, where $\psi \in \Psi$ minimizes $\sum_{q=1}^{Q} \sum_{k=0}^{\sigma^q(N)-1} \left(f_\psi(x^{(k,q)}) - \hat{h}_{k,q}\right)^2$ following (4.4) ;

  Estimate advantage functions $\hat{A}_{\theta_i}\left(x^{(k,q)}, a^{(k,q)}\right)$ using (4.5) for each $q = 1, .., Q$, $k = 0, .., \sigma^q(N) - 1$:

$$D^{(1:Q),(0:\sigma^q(N)-1)} = \left\{\left(x^{(0,q)}, a^{(0,q)}, \hat{A}_{0,q}\right), \cdots, \left(x^{(\sigma^q(N)-1,q)}, a^{(\sigma^q(N)-1,q)}, \hat{A}_{\sigma^q(N)-1,q}\right)\right\}_{q=1}^{Q}.$$

  Minimize surrogate objective function w.r.t. $\theta \in \Theta$:

$$\hat{L}\left(\theta, \theta_i, D^{(1:Q),(0:\sigma^q(N)-1)}\right) = \sum_{q=1}^{Q} \sum_{k=0}^{\sigma^q(N)-1} \max\Big[\frac{\pi_\theta(a^{(k,q)}|x^{(k,q)})}{\pi_{\theta_i}(a^{(k,q)}|x^{(k,q)})} \hat{A}_{\theta_i}(x^{(k,q)}, a^{(k,q)}),$$
$$\text{clip}\left(\frac{\pi_\theta(a^{(k,q)}|x^{(k,q)})}{\pi_{\theta_i}(a^{(k,q)}|x^{(k,q)})}, 1-\epsilon, 1+\epsilon\right) \hat{A}_{\theta_i}(x^{(k,q)}, a^{(k,q)})\Big]$$

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

end

In practice a naive (standard) Monte estimator (4.3) fails to improve in PPO policy iteration algorithm 1 due the large variance and cannot provide a reliable estimate of $h_\eta(x)$. In the next several sections, we develop progressively a sequence of estimators. When apply them into the PPO algorithm, their performance improves. We end this section with the following two remarks.

Remark 3. *For any state $x \in \mathcal{X}$ the one-replication estimate (4.3) 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 an one-replication estimate is computed when state x is visited for the first time within a cycle and next visits at state x within the same cycle are ignored. For more details about every-visit and first-visit Monte-Carlo methods see [76, Section 5.1].*

Remark 4. *In regression problem (4.4), each data point $(x^{(k)}, \hat{h}_k)$ is used in the quadratic loss function, despite that many of the $x^{(k)}$'s represent the same state. One could restrict that* only distinct *$x^{(k)}$'s are used in the loss function, with corresponding $\hat{h}_k$'s properly averaged. It turns out that this new optimization problem yields the same optimal solution as the one in (4.4). The equivalence of the optimization problems follows from the fact that for an arbitrary sequence of real numbers $a_1, .., a_n \in \mathbb{R}$:*

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

4.2 Approximating martingale-process method

Estimator (4.3) 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 only at the regeneration state. In this section we discuss how one can decrease the variance by reducing the magnitude of summands in (4.3) if an approximation ζ of the solution to Poisson's equation h_η is available.

Let us assume an episode $\left\{x^{(0)}, a^{(1)}, x^{(1)}, a^{(2)}, \cdots, x^{(K-1)}, a^{(K-1)}, x^{(\sigma(N))}\right\}$ has been generated under policy π_η. From the definition of a solution to Poisson's equation (3.3):

$$g(x^{(k)}) - \mu_\eta^T g = h_\eta(x^{(k)}) - \sum_{y \in \mathcal{X}} P_\eta\left(y|x^{(k)}\right) h_\eta(y) \text{ for each state } x^{(k)} \text{ in the simulated episode.}$$

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

$$g(x^{(k)}) - \widehat{\mu_\eta^T g} \quad \text{and} \quad \zeta(x^{(k)}) - \sum_{y \in X} P_\eta\left(y|x^{(k)}\right) \zeta(y)$$

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 [35], see also [3].

Following [35, Proposition 7], for some approximation ζ such that $\mu_\eta^T \zeta < \infty$ and $\zeta(x^*) = 0$, we consider the martingale process starting from an arbitrary state $x^{(k)}$ until the first regeneration time:

$$M_{\sigma_k}(x^{(k)}) = \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k - 1} \left[\sum_{y \in \mathcal{X}} P_\eta\left(y|x^{(t)}\right) \zeta(y) - \zeta(x^{(t)}) \right], \tag{4.7}$$

where $\sigma_k = \min\left\{t > k \mid x^{(t)} = x^*\right\}$ is the first time when the regeneration state x^* is visited after time k. The martingale process (4.7) has zero expectation $\mathbb{E}M_n = 0$ for all $n \geq 0$ and is used as a control variate to define a new estimator. Adding M_{σ_k} to estimator (4.3) we get the AMP estimator of the solution to Poisson's equation:

$$\hat{h}_\eta^{AMP(\zeta)}(x^{(k)}) := \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k - 1} \left(g(x^{(t)}) - \widehat{\mu_\eta^T g} + \sum_{y \in \mathcal{X}} P_\eta\left(y|x^{(t)}\right) \zeta(y) - \zeta(x^{(t)}) \right). \tag{4.8}$$

Let assume that estimation of the average cost is accurate, i.e. $\widehat{\mu_\eta^T g} = \mu_\eta^T g$. In this case estimator (4.8) has zero variance if the approximation is exact $\zeta = h_\eta$.

Now we want to replace the standard regenerative estimator (4.3) that is used in line 8 of Algorithm 1 by AMP estimator (4.8). As the approximation ζ needed in (4.8), we use $f_{\psi_{i-1}}$ that approximates a solution to the Poisson's equation corresponding to previous policy $\pi_{\theta_{i-1}}$. 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 (4.8).

Algorithm 2: Proximal policy optimization with AMP method

Result: policy π_{θ_I}

Initialize policy π_{θ_0} and value function $f_{\psi_{-1}} \equiv 0$ approximators ;

for *policy iteration* $i = 0, 1, ..., I-1$ **do**

  for *actor* $q = 1, 2, ..., Q$ **do**

    Run policy π_{θ_i} until it reaches Nth regeneration time on $\sigma^q(N)$ step: collect an episode $\left\{x^{(0,q)}, a^{(0,q)}, x^{(1,q)}, a^{(1,q)}, \cdots, x^{(\sigma^q(N)-1,q)}, a^{(\sigma^q(N)-1,q)}, x^{(\sigma^q(N),q)}\right\}$;

  end

  Compute average cost estimate $\widehat{\mu_{\theta_i}^T g}$ by (4.2);

  Compute $\hat{h}_{k,q}^{AMP(f_{\psi_{i-1}})}$, estimate of $h_{\theta_i}(x^{(k,q)})$, by (4.8) for each $q = 1, .., Q$, $k = 0, .., \sigma^q(N) - 1$;

  Update $\psi_i := \psi$, where $\psi \in \Psi$ minimizes $\sum_{q=1}^{Q} \sum_{k=0}^{\sigma^q(N)-1} \left(f_\psi(x^{(k,q)}) - \hat{h}_{k,q}^{AMP(f_{\psi_{i-1}})}\right)^2$ following (4.4);

  Estimate advantage functions $\hat{A}_{\theta_i}(x^{(k,q)}, a^{(k,q)})$ using (4.5) for each $q = 1, .., Q$, $k = 0, ..., \sigma^q(N) - 1$:

$$D^{(0:\sigma^q(N)-1)_{q=1}^Q} = \left\{\left(x^{(0,q)}, a^{(0,q)}, \hat{A}_{0,q}\right), \cdots, \left(x^{(\sigma^q(N)-1,q)}, a^{(\sigma^q(N)-1,q)}, \hat{A}_{\sigma^q(N)-1,q}\right)\right\}_{q=1}^{Q}.$$

  Minimize surrogate objective function w.r.t. $\theta \in \Theta$:

$$\hat{L}\left(\theta, \theta_i, D^{(0:\sigma^q(N)-1)_{q=1}^Q}\right) = \sum_{q=1}^{Q} \sum_{k=0}^{\sigma^q(N)-1} \max\left[\frac{\pi_\theta(a^{(k,q)}|x^{(k,q)})}{\pi_{\theta_i}(a^{(k,q)}|x^{(k,q)})} \hat{A}_{\theta_i}(x^{(k,q)}, a^{(k,q)}),\right.$$
$$\left. \text{clip}\left(\frac{\pi_\theta(a^{(k,q)}|x^{(k,q)})}{\pi_{\theta_i}(a^{(k,q)}|x^{(k,q)})}, 1-\epsilon, 1+\epsilon\right) \hat{A}_{\theta_i}(x^{(k,q)}, a^{(k,q)})\right]$$

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

end

4.3 Variance reduction through discount

In the previous section we applied AMP method to reduce the variance of the summands in (4.3). Unless an approximation ζ is exact, each term in the summation in (4.8) is random with nonzero variance. When the expected length of a regeneration cycle is large the cumulative variance of estimator (4.8) still can be devastating.

The common approach to overcome this issue is to introduce a forgetting factor $\gamma \in (0, 1)$ to discount the future relative costs [39, 8, 54, 41, 77, 70].

Let

$$r_\eta(x^*) := (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t g(x^{(t)}) \mid x^{(0)} = x^*\right] \tag{4.9}$$

be a *present discounted value* at state x^*; the term "present discounted value" is proposed in [79, Section 11.2]. We define the *regenerative discounted relative value function as*:

$$V_\eta^{(\gamma)}(x) := \mathbb{E}\left[\sum_{t=0}^{\sigma(x^*)-1} \gamma^t \left(g(x^{(t)}) - r_\eta(x^*)\right) \;\middle|\; x^{(0)} = x\right] \text{ for each } x \in \mathcal{X}, \tag{4.10}$$

where $x^{(k)}$ is the state of the Markov chain with transition matrix P_η at time k, x^* is the prespecified regeneration state, $\gamma \in (0, 1]$ is a discount factor. We note that $V_\eta^{(\gamma)}(x^*) = 0$ by definition. It follows from [67, Corollary 8.2.5.] that, under the drift condition, $r_\eta(x^*) \to \mu_\eta^T g$ as $\gamma \uparrow 1$. Furthermore, by Lemma 9, for each $x \in \mathcal{X}$

$$V_\eta^{(\gamma)}(x) \to h^{(x^*)}(x) \text{ as } \gamma \uparrow 1,$$

where $h^{(x^*)}$ is a solution to the Poisson equation given in (4.1).

For a fixed $\gamma \in (0,1)$ and state $x \in \mathcal{X}$ any unbiased estimator of $V_\eta^{(\gamma)}(x)$ will be a biased estimator of $h^{(x^*)}(x)$. It turns out that the discount conterparts of estimators (4.3) and (4.8) for $V_\eta^{(\gamma)}(x)$ have smaller variances than these two estimators for $h^{(x^*)}(x)$. This variance reduction can be explained intutively as follows. Introducing the discount factor γ can be interpered as a modification of the original transition dynamics; under the modified dynamics, any action produces a transition into a regeneration state with probability at least $1-\gamma$, thus shortening the length of a regenerative cycles; see Section B for details of the modification and formula B.3 there.

We define a *discounted* advantage function as

$$A_\eta^{(\gamma)}(x,a) := \mathop{\mathbb{E}}_{y\sim P(\cdot|x,a)} \left[g(x) - \mu_\eta^T g + V_\eta^{(\gamma)}(y) - V_\eta^{(\gamma)}(x) \right]. \tag{4.11}$$

For two policies π_θ and π_η, $\eta, \theta \in \Theta$, we define an approximation of $N_1(\theta,\eta)$ (3.15)

$$N_1^{(\gamma)}(\theta,\eta) := \mathop{\mathbb{E}}_{\substack{x\sim\mu_\eta \\ a\sim\pi_\theta(\cdot|x)}} \left[r_{\theta,\eta}(a|x) A_\eta^{(\gamma)}(x,a) \right]. \tag{4.12}$$

With available function approximation f_ψ of $V_\eta^{(\gamma)}$, the advantage function (4.11) can be estimated as (4.5).

We now present the discounted version of the APM estimator (4.8). Let ζ be an approximation of the discounted value function $V_\eta^{(\gamma)}$ such that $\mu_\eta^T \zeta < \infty$ and $\zeta(x^*) = 0$. We define the sequence $(M_\eta^{(n)} : n \geq 0)$ where

$$M_\eta^{(n)}(x) := \sum_{t=k+1}^{n} \gamma^{t-k} \left[\zeta(x^{(t)}) - \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t-1)}\right) \zeta(y) \right], \tag{4.13}$$

where $x = x^{(k)}$ and $x^{(t)}$ is a state of the Markov chain after t steps.

We define a one-replication of the AMP estimator for discounted value function:

$$\begin{aligned} \hat{V}_\eta^{AMP(\zeta),(\gamma)}(x^{(k)}) &:= \sum_{t=k}^{\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} \right) - M_\eta^{(\sigma_k-k)}(x^{(k)}) \qquad (4.14) \\ &= \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} + \gamma \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t)}\right) \zeta(y) - \zeta(x^{(t)}) \right) - \gamma^{\sigma_k} \xi(x^*) \\ &= \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} + \gamma \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t)}\right) \zeta(y) - \zeta(x^{(t)}) \right), \end{aligned}$$

where $\widehat{r_\eta(x^*)}$ is an estimation of $r(x^*)$, $\sigma_k = \min\left\{ t > k \mid x^{(t)} = x^* \right\}$ is the first time when the regeneration state x^* is visited after time k.

The AMP estimator (4.18) does not introduce any bias subtracting M_η from $\hat{V}_\eta^{(\gamma)}$ since $\mathbb{E}M_\eta^{(n)} = 0$ for any $n > 0$ by [35]. Function $V_\eta^{(\gamma)}$ is a solution of the following equation, see Lemma 8:

$$g(x) - r_\eta(x^*) + \gamma \sum_{y\in\mathcal{X}} P_\eta(y|x) h(y) - h(x) = 0 \quad \text{for each } x \in \mathcal{X}. \tag{4.15}$$

Therefore, similarly to (4.8), estimator (4.18) has zero variance if approximation is exact $\widehat{r_\eta(x^*)} = r_\eta(x^*)$ and $\zeta = V_\eta$, see Poisson's equation (4.15).

Further variance reduction is possible via *T-step truncation* [76, Section 6]. Consider an estimate of the value function (4.10) at a state $x \in \mathcal{X}$ as a sum of discounted costs before time T, where $T < \sigma(x^*)$, and discounted costs after time T:

$$\hat{V}^{(\gamma)}(x) = \sum_{t=0}^{T-1} \gamma^t \left(g(x^{(t)}) - \widehat{r(x^*)} \right) + \gamma^T \sum_{t=0}^{\sigma(x^*)-1} \gamma^t \left(g(x^{(T+t)}) - \widehat{r(x^*)} \right), \tag{4.16}$$

where $x^{(0)} = x$, $x^{(t)}$ is a state of the Markov chain after t steps and $\sum_{t=0}^{\sigma(x^*)-1} \gamma^t g(x^{(T+t)})$ is a standard one-replication estimation of the value function at state $x^{(T)}$. Instead of estimating the value at state $x^{(T)}$ by a random roll-out (second term in (4.16)), one can use the value of deterministic approximation function ζ at state $x^{(T)}$. The T-step truncation reduces variance of the standard estimator but introduces bias unless the approximation is exact $\zeta(x^{(T)}) = V^{(\gamma)}(x^{(T)})$.

A *T-truncated* version of the AMP estimator is

$$\begin{aligned}
\hat{V}_k^{AMP(\zeta),(\gamma,T)} :&= \sum_{t=k}^{T\wedge\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} \right) + \gamma^{T\wedge\sigma_k} \zeta(x^{(T\wedge\sigma_k)}) - M_\eta^{(T\wedge\sigma_k)}(x^{(k)}) \qquad (4.17)\\
&= \sum_{t=k}^{T\wedge\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} \right) - \sum_{t=k+1}^{T\wedge\sigma_k} \gamma^{t-k} \left(\zeta(x^{(t)}) - \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t-1)}\right) \zeta(y) \right)\\
&\quad + \gamma^{T\wedge\sigma_k} \zeta(x^{(k+T)})\\
&= \zeta(x^{(k)}) + \sum_{t=k}^{T\wedge\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} + \gamma \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t)}\right) \zeta(y) - \zeta(x^{(t)}) \right),
\end{aligned}$$

where $T \wedge \sigma_k = \min(T, \sigma_k)$. We note that if the value function approximation and present discounted value approximation are exact, estimator (4.17) is unbiased for $V^{(\gamma)}(x^{(k)})$ and has zero variance. The T-truncated estimator can be generalized by taking the number of summands T to follow geometrical distribution with parameter $\lambda < 1$ as in TD(λ) method [76, Section 12], [70, Section 3]:

$$\begin{aligned}
\hat{V}_k^{AMP(\zeta),(\gamma,\lambda)} :&= \mathbb{E}_{T\sim Geom(1-\lambda)} \hat{V}_k^{AMP(\zeta),(\gamma,T)} \qquad (4.18)\\
&= (1-\lambda)\Big(\hat{V}_k^{AMP(\zeta),(\gamma,1)} + \lambda \hat{V}_k^{AMP(\zeta),(\gamma,2)} + \lambda^2 \hat{V}_k^{AMP(\zeta),(\gamma,3)} + \cdots\\
&\qquad + \lambda^{\sigma_k} V_k^{AMP(\zeta),(\gamma,\sigma_k)} + \lambda^{\sigma_k+1} V_k^{AMP(\zeta),(\gamma,\sigma_k)} + \cdots \Big)\\
&= \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k-1} (\gamma\lambda)^{t-k} \left(g(x^{(t)}) - \widehat{r_\eta(x^*)} + \gamma \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t)}\right) \zeta(y) - \zeta(x^{(t)}) \right).
\end{aligned}$$

The regenerative cycles can be very long. In practice we want to control/predict time and memory amount allocated for the algorithm execution. Therefore, the simulated episodes should have finite length. We use the following estimation for first N timesteps if an episode with finite length $N + L$ is generated:

$$\hat{V}_k^{AMP(\zeta),(\gamma,\lambda,L)} := \zeta(x^{(k)}) + \sum_{t=0}^{L\wedge\sigma_k-1} (\gamma\lambda)^t \left(g(x^{(k+t)}) - \widehat{r_\eta(x^*)} + \gamma \sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(k+t)}\right) \zeta(y) - \zeta(x^{(k+t)}) \right), \qquad (4.19)$$

where $k = 0, ..., N-1$, and L is large enough integer. We note that if an episode has finite length $N+L$, regeneration σ_k may have not been observed in the generated episode, i.e. $\sigma_k > N + L$. In this case summation in (4.19) is done up to L.

We provide the proximal policy optimization algorithm where each of $q = 1, ..., Q$ parallel actors simulate an episode with length $N + L$: $\left\{x^{(0,q)}, a^{(0,q)}, x^{(1,q)}, a^{(1,q)}, \cdots, x^{(N+L-1,q)}, a^{(N+L-1,q)}\right\}$. As the approximation ζ needed in (4.19), we use $f_{\psi_{i-1}}$ that approximates a regenerative discounted value

function corresponding to previous policy $\pi_{\theta_{i-1}}$, see Algorithm 3.

Algorithm 3: Proximal policy optimization with discounting

Result: policy π_{θ_I}

Initialize policy π_{θ_0} and value function $f_{\psi_{-1}} \equiv 0$ approximators ;

for *policy iteration* $i = 0, 1, ..., I-1$ **do**

  for *actor* $q = 1, 2, ..., Q$ **do**

    Run policy π_{θ_i} for $N + L$ timesteps: collect an episode $\left\{x^{(0,q)}, a^{(0,q)}, x^{(1,q)}, a^{(1,q)}, ..., x^{(N+L-1,q)}, a^{(N+L-1,q)}\right\}$;

  end

  Estimate average cost $\widehat{\mu_{\theta_i}^T g}$ by (4.2), present discounted value $\widehat{r_{\theta_i}(x^*)}$ by (4.20) below;

  Compute $\hat{V}_{k,q}^{AMP(f_{\psi_{i-1}}),(\gamma,\lambda)}$ estimates by (4.19) for each $q = 1, ..., Q$, $k = 0, .., N-1$;

  Update $\psi_i := \psi$, where $\psi \in \Psi$ minimizes $\sum_{q=1}^{Q} \sum_{k=0}^{N-1} \left(f_\psi \left(x^{(k,q)} \right) - \hat{V}_{k,q}^{AMP(f_{\psi_{i-1}}),(\gamma,\lambda)} \right)^2$ following (4.4) ;

  Estimate advantage functions $\hat{A}_{\theta_i}\left(x^{(k,q)}, a^{(k,q)}\right)$ using (4.5) for each $q = 1, .., Q$, $k = 0, .., N-1$:

$$D^{(0:N-1)_{q=1}^{Q}} = \left\{ \left(x^{(0,q)}, a^{(0,q)}, \hat{A}_{\theta_i}^{(\gamma)}(x^{(0,q)}, a^{(0,q)}) \right), \cdots, \left(x^{(N-1,q)}, a^{(N-1,q)}, \hat{A}_{\theta_i}^{(\gamma)}(x^{(N-1,q)}, a^{(N-1,q)}) \right) \right\}_{q=1}^{Q}$$

  Minimize surrogate objective function w.r.t. $\theta \in \Theta$:

$$\hat{L}^{(\gamma)}\left(\theta, \theta_i, D^{(0:N-1)_{q=1}^{Q}}\right) = \sum_{q=1}^{Q} \sum_{k=0}^{N-1} \max \Big[\frac{\pi_\theta\left(a^{(k,q)}|x^{(k,q)}\right)}{\pi_{\theta_i}\left(a^{(k,q)}|x^{(k,q)}\right)} \hat{A}_{\theta_i}^{(\gamma)}\left(x^{(k,q)}, a^{(k,q)}\right), \text{clip}\left(\frac{\pi_\theta\left(a^{(k,q)}|x^{(k,q)}\right)}{\pi_{\theta_i}\left(a^{(k,q)}|x^{(k,q)}\right)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_{\theta_i}^{(\gamma)}\left(x^{(k,q)}, a^{(k,q)}\right) \Big];$$

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

end

In Algorithm 3, assume that state x^* have been visited N_q times in the qth generated episode, when $q = 1, .., Q$ parallel actors are available. The present discounted value $r(x^*)$ can be estimated by

$$\widehat{r(x^*)} = (1-\gamma) \sum_{q=1}^{Q} \sum_{n=1}^{N_q} \sum_{k=\sigma_n}^{\sigma^q(n)+L} \gamma^{k-\sigma^q(n)} g(x^{(k,q)}), \tag{4.20}$$

where $\sigma^q(n)$ is the nth time when state x^* is visited in the qth episode, L is a large enough integer. If many parallel actors are available, we recommend to start episodes from state x^* to ensure that state x^* appears in the generated episodes sufficient number of times.

Remark 5. *One can use the following discounted value function as an approximation of the solution of Poisson's equation h_η:*

$$J^{(\gamma)}(x) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(g(x^{(k)}) - \mu^T g \right) \mid x^{(0)} = x \right] \text{ for each } x \in \mathcal{X}. \tag{4.21}$$

We note that discounted value function (4.21) and regenerative discounted value function (4.10) are solutions of the same Poisson's equation, see Lemma B.2. Therefore, the bias of advantage function estimator (4.11) does not change when h_η in (3.16) is replaced either by $J^{(\gamma)}$ or by $V^{(\gamma)}$. The variance of regenerative discounted value function estimator (4.19) can be potentially smaller than the variance of analogous $J^{AMP(\xi),(\gamma,\lambda,L)}$ estimator

$$\hat{J}_k^{AMP(\zeta),(\gamma,\lambda,L)} := \zeta(x^{(k)}) + \sum_{j=0}^{L-1} (\gamma\lambda)^j \left(g(x^{(k+j)}) - \widehat{\mu_\eta^T g} + \gamma \sum_{y \in \mathcal{X}} P_\eta\left(y|x^{(k+j)}\right) \zeta(y) - \zeta(x^{(k+j)}) \right). \tag{4.22}$$

The upper bound of summation in (4.19) is $\min(\sigma_k, L) - 1$ *hence this summation can include less summands than summation in (4.22) if the regeneration happens frequently. In Section D of Appendix we present a numerical experiment for the criss-cross network which implies that a particular choice between* $J^{AMP(\xi),(\gamma,\lambda,L)}$ *and* $V^{AMP(\xi),(\gamma,\lambda,L)}$ *estimators can affect the PPO algorithm convergence rate to the optimal policy.*

We end this subsection by commenting on the connection of the AMP estimators proposed in this paper with the GAE estimator prossed in [70]. This connection provides an alternative motivation of introducing the GAE estimator. An accurate estimation of the value function $V_\eta^{(\gamma)}$ by the standard estimator might require a large number of state-action pairs samples from the current policy π_η [76, Section 13], [70, 38]. In this paper we apply the approximating martingale-process (AMP) method to propose the discounted AMP estimator in (4.19) that has the smaller variance than that of the estimators (4.16). The major drawback of the AMP method is a necessity of the knowledge of transition probabilities $P(y|x,a)$ so that the expected values

$$\mathop{\mathbb{E}}_{\substack{a\sim\pi_\eta(\cdot|x)\\ y\sim P(\cdot|x,a)}} \zeta(y) = \sum_{y\in\mathcal{X}} \sum_{a\in\mathcal{A}} P(y|x,a)\pi_\eta(a|x)\zeta(y)$$

can be computed exactly for each $(x,\ a,\ y) \in \mathcal{X} \times \mathcal{A} \times \mathcal{X}$.

One may want to relax this requirement replacing each expected value $\mathop{\mathbb{E}}_{\substack{a\sim\pi_\eta\left(\cdot|x^{(t)}\right)\\ y\sim P(\cdot|x,a)}} \zeta(y)$ by its *one-replication estimate* $\zeta\left(x^{(t+1)}\right)$, where $\left(x^{(t)}, x^{(t+1)}\right)$ are two sequential states from an episode

$$\left(x^{(0)}, \cdots, x^{(t)}, x^{(t+1)}, \cdots\right)$$

generated under policy π_η.

In Section 4.3 we propose the AMP estimator of the regenerative discounted value function (4.18):

$$\hat{V}_k^{AMP(\zeta),(\gamma,\lambda)} = \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k-1} (\gamma\lambda)^{t-k} \left(g(x^{(t)}) - \widehat{r(x^*)} + \gamma \sum_{y\in\mathcal{X}} P_\eta(y|x^{(t)})\zeta(y) - \zeta(x^{(t)}) \right). \tag{4.23}$$

Replacing expectations $\sum_{y\in\mathcal{X}} P_\eta\left(y|x^{(t)}\right)\zeta(y)$ by the one-replication estimates $\zeta\left(x^{(t+1)}\right)$ one obtains the following estimator for the value function:

$$\hat{V}_k^{GAE(\zeta),(\gamma,\lambda)} := \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k-1} (\gamma\lambda)^{t-k} \left(g(x^{(k+t)}) - \widehat{r(x^*)} + \gamma\zeta(x^{(t+1)}) - \zeta(x^{(t)}) \right), \tag{4.24}$$

which is already known in the literature as a part of the general advantage estimation (GAE) method [70]. We note that the advantage function is estimated as $\hat{A}_k^{GAE(\zeta),(\gamma,\lambda)} := \hat{V}_k^{GAE(\zeta),(\gamma,\lambda)} - \zeta\left(x^{(k)}\right)$ in [70].

We note that when $\lambda = 1$ estimator (4.18) is transformed into the standard estimator and does not depend on ξ:

$$\begin{aligned}\hat{V}_\eta^{(\zeta),(\gamma)}(x^{(k)}) :&= \zeta(x^{(k)}) + \sum_{t=k}^{\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r(x^*)} + \gamma\zeta(x^{(t+1)}) - \zeta(x^{(t)}) \right) \\ &= \sum_{t=k}^{\sigma_k-1} \gamma^{t-k} \left(g(x^{(t)}) - \widehat{r(x^*)} \right).\end{aligned}$$

Thus, the martingale-based control variate has no effect on the GAE estimator when $\lambda = 1$, but can produce significant variance reduction in the AMP estimator avoiding addition of the bias; see Section 5.1 for numerical experiments.

5 Experimental results for multiclass queueing networks

In this section we evaluate performance of the proposed proximal policy optimization Algorithms 1, 2 and 3 for multiclass queueing 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 E 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 $I = 200$ policy iterations. Each iteration the algorithm uses $Q = 50$ actors to simulate data in parallel. See Section F in the Appendix for more details on the experimental setup and hyperparameters.

5.1 Criss-cross network

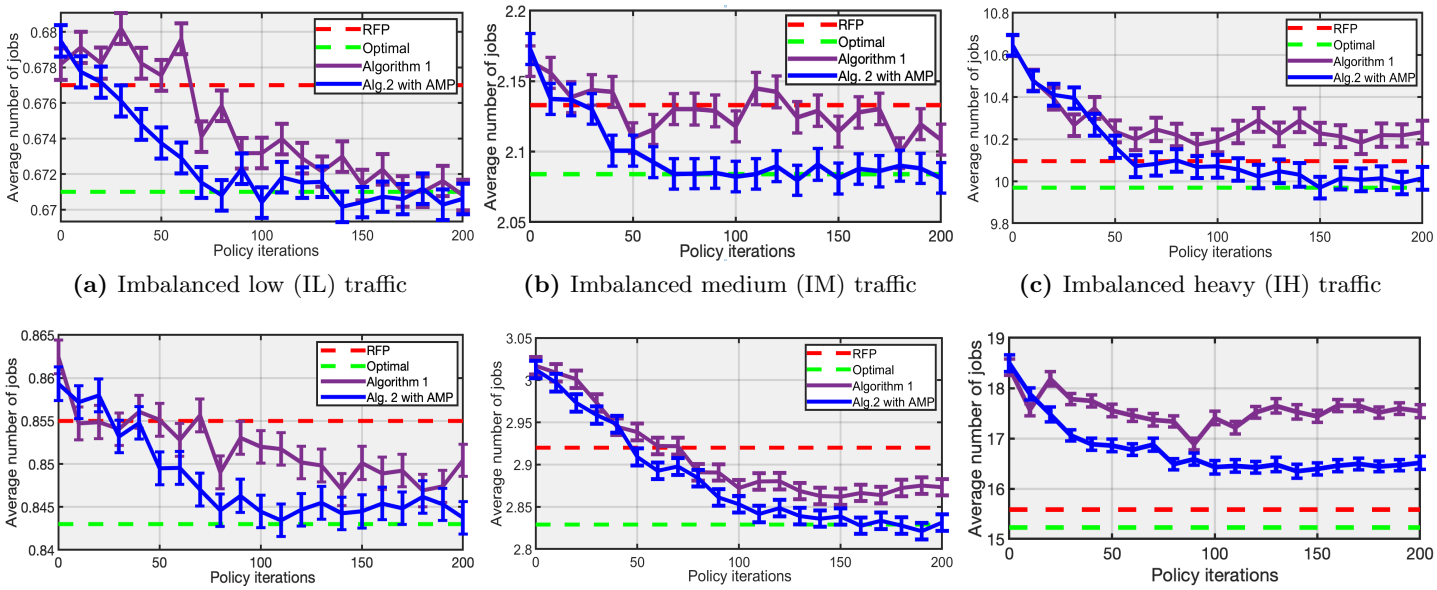
We first study PPO algorithm and compare its base version Algorithm 1 and its modification Algorithm 2 that incorporates the AMP method. We check the robustness of the algorithms for the criss-cross system with various load (traffic) intensity regimes, including I.L. (imbalanced light), B.L. (balanced 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 in any of these traffic regimes is stable under any work-conserving policy [24].

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 [65] and denoted by TP in the table. In the fourth column we list the performance of a threshold policy proposed in [34]. In the fifth and sixth columns we list the performance of fluid (FP) and robust fluid (RFP) policies from [12].

In the last column of Table 2 we provide simulation results of PPO policy π_{θ_I} obtained from Algorithm 2. (We will add more details of the simulation in the last two paragraphs of this section.) 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. We also report the half width of the 95% confidence intervals (CIs) of the performance of the resulting policy in the last column.

The policy NN parameters θ_0 are initialized using standard Xavier initialization [26]. 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 2 we save policy parameters $\{\theta_i\}_{i=0,10,\ldots,200}$ every 10th iteration over the course of learning. After the algorithm terminates we independently simulate policies $\{\pi_{\theta_i} : i = 0, 10, ..., 200\}$ (in parallel) 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 5×10^7 (5×10^6, 10^6) regenerative cycles correspondingly. 95%−confidence intervals are computed using the strongly consistent estimator of asymptotic variance, see details in [4, 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) with CIs
I.L.	0.671	0.678	0.679	0.678	0.677	0.671 ± 0.001
B.L.	0.843	0.856	0.857	0.857	0.855	0.844 ± 0.004
I.M.	2.084	2.117	2.129	2.162	2.133	2.084 ± 0.011
B.M.	2.829	2.895	2.895	2.965	2.920	2.833 ± 0.010
I.H.	9.970	10.13	10.15	10.398	10.096	10.014 ± 0.055
B.H.	15.228	15.5	15.5	18.430	15.585	16.513 ± 0.140

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 $\pi_{\theta_{200}}$ whose long-run average performance is 15.353 ± 0.138 jobs, which is lower than the RFP performance in [12]. See the comparison

of the learning curves of Algorithm 2 and Algorithm 3 in Figure 3. Figure 3 shows clearly that Algorithm 3 (by using discount factors) achieves much better the variance reduction than Algorithm 2.

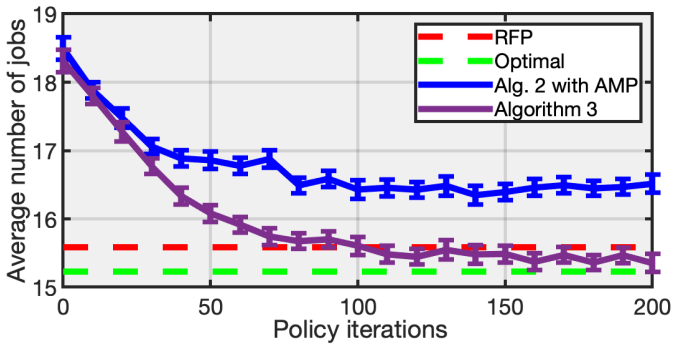


Figure 3: Comparison of learning curves from Algorithm 2 and Algorithm 3 on the criss-cross network with balanced heavy regime.

Recall that in the uniformization procedure, the transition matrix $\tilde{P}$ in (2.5) allows "fictitious" transitions when $\tilde{P}(x|x) > 0$ for some state $x \in \mathcal{X}$. The PPO algorithm approximately solves the discrete-time MDP (2.12), which allows a decision at every state transition, including fictitious ones. The algorithm produces randomized stationary Markovian policies. In evaluating the performance of any such policies, we actually simulate a DTMC $\{x^{(k)} : k = 0, 1, 2, \ldots\}$ operating under the policy, estimating the corresponding long-run average cost as in (2.12). There are two versions of DTMCs depending on how often the randomized policy is sampled to determine the next action. In version 1, the policy is re-sampled only when a *real* state transition occurs. Thus, whenever a fictitious state transition occurs, no new action is determined and server priorities remain unchanged in this version. In version 2, the policy is re-sampled at the *every* transition. Unfortunately, there is no guarantee that these two versions of DTMCs yield the same long-run average cost, see [14, Example 2.2] for a counterexample.

When the randomized stationary Markovian policy is optimal for the discrete-time MDP (2.12), under an additional mild condition, these two versions of DTMCs yield the same long-run average cost; see [14, Theorem 3.6]. Whenever simulation is used to estimate the performance of a randomized policy in this paper, we use version 1 of the DTMC. The reason for this choice is that the long-run average for this version of DTMC is the same as the continuous-time long-run average cost in (2.11), and the latter performance has been used as benchmarks in literature. Although the final randomized stationary Markovian policy from our PPO algorithm is not expected to be optimal, our numerical experiments demonstrate that the performance of these two versions is statistically identical. See, for example, Table 3 for the performance of these two versions in the criss-cross network. Version 1 column is identical to the last column of Table 2.

Load regime	Version 1 performance with CIs	Version 2 performance with CIs
I.L.	0.671 ± 0.001	0.671 ± 0.001
B.L.	0.844 ± 0.004	0.844 ± 0.004
I.M.	2.084 ± 0.011	2.085 ± 0.011
B.M.	2.833 ± 0.010	2.832 ± 0.010
I.H.	10.014 ± 0.055	9.998 ± 0.054
B.H.	16.513 ± 0.140	16.480 ± 0.137

Table 3: Average number of jobs per unit time in the discrete-time MDP model of the criss-cross network under PPO polices.

5.2 Extended six-class queueing network

In the next experiment we consider the family of extended six-class networks from [12] 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: $\lambda_1 = \lambda_3 = 9/140$, 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 $\rho = 0.9$.

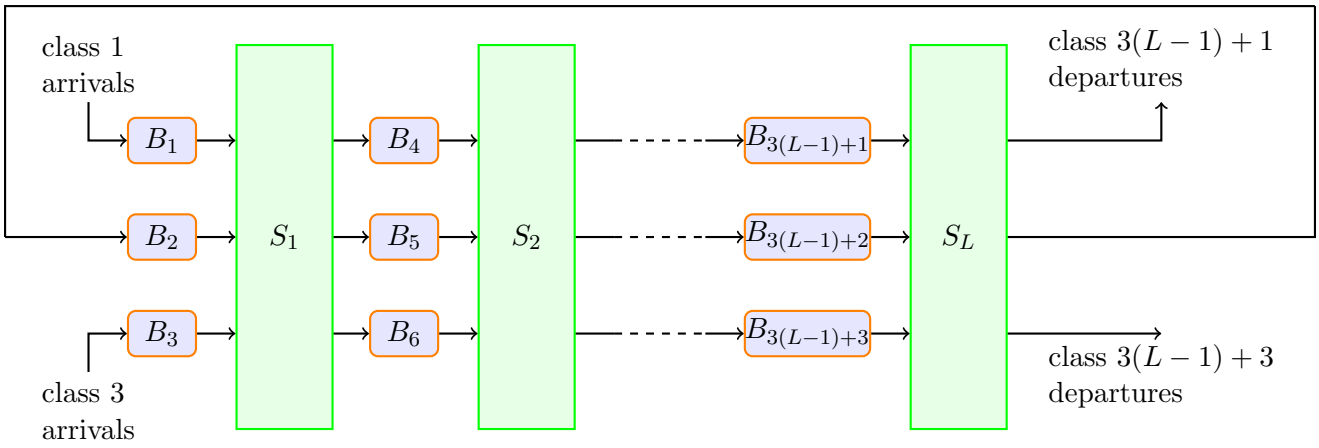


Figure 4: The extended six-class network.

Num. of classes 3L	LBFS	FCFS	FP	RFP	PPO (Algorithm 3) with CIs
6	15.749	40.173	15.422	15.286	14.130 ± 0.208
9	25.257	71.518	26.140	24.917	23.269 ± 0.251
12	34.660	114.860	38.085	36.857	32.171 ± 0.556
15	45.110	157.556	45.962	43.628	39.300 ± 0.612
18	55.724	203.418	56.857	52.980	51.472 ± 0.973
21	65.980	251.657	64.713	59.051	55.124 ± 1.807

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

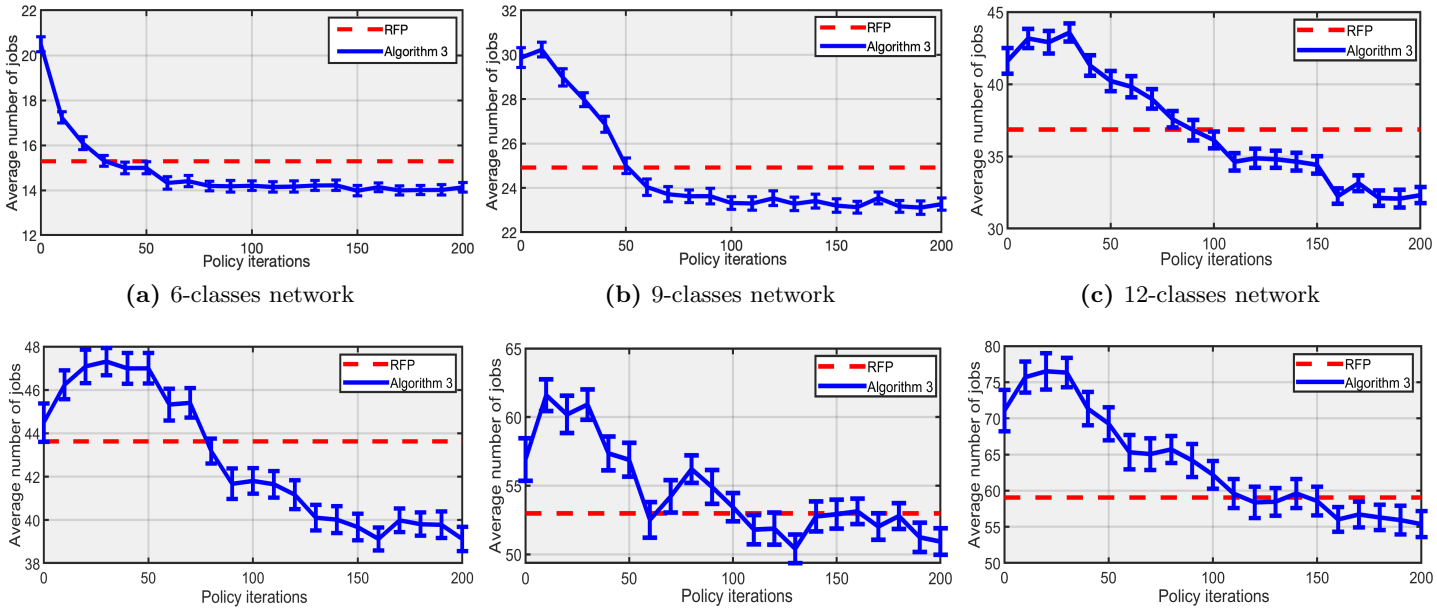
Table 4 provides the performance of the PPO policy and compares it with other heuristic methods for the extended six-class queueing networks. In the experiments we change the size of the network to test the robustness of the PPO policies. The size of the networks varies from 6 to 21 classes. In all experiments 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 [12]. We note that the displayed performance of robust fluid policy in Table 4 is the performance of *the best* robust fluid policy which corresponds to the best choice of policy parameters that is different for each extended network. LBFS refers to the last-buffer first-serve policy, where the priority at a server is given to jobs with highest index. FCFS refers to the first-come first-serve policy, where the 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 might yield an unstable policy for extended six-class networks. To overcome this issue, we fix a stable, randomized policy for the discrete-time MDP. We simulate a long episode of this MDP operating under the randomized policy. At each timestep we save the state at the time and the corresponding probability distribution over the actions. We use this simulated data set to train the policy NN π_{θ_0}. In our numerical experiments, we use the *proportionally randomized (PR)* policy as the initial stable policy. When the network operates under the PR policy, whenever an arrival or service completion event happens at station k, a nonempty buffer j receives a priority over other classes at the station with probability

$$\frac{x_j}{\sum\limits_{i \in \mathcal{B}(k)} x_i}, \tag{5.1}$$

where $\mathcal{B}(k)$ is a set of buffers associated to server k and $x = (x_1, ..., x_J)$ is the jobcount at the time of the event after the job arrivals and departures have been accounted for. The priority stays fixed until the next arrival or service completion event happens. The PR policy is maximally stable for open MQNs, meaning that if the system is unstable under PR policy there is no other policy that can stabilize it; see Section C in Appendix.

In each plot in Figure 5, we save policy NN parameters $\{\theta_i\}_{i=0,10,...,200}$ every 10th policy iteration. For each saved policy NN, we conduct a separate long simulation of the queueing network operating under the policy for accurate performance evaluation by providing a 95% confidence interval of the long-run average cost. For any of the six queueing networks, when the load is high, the regeneration is rare. Thus, we adopt the *batch means* method to estimate the confidence interval [36, Section 6]. For each policy from the set $\{\pi_{\theta_i} : i = 0, 10, ..., 200\}$, we simulate an episode starting from an empty state $x = (0, .., 0)$ until 5×10^6 arrival events happened. Then we estimate average performance of the policy based on this episode. To compute the confidence interval from the episode, we split the episode into 50 sub-episodes (batches), see also [63]. Pretending that the obtained 50 mean estimates are i.i.d. we compute 95%−confidence intervals shown on the plots in Figure 5.



(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: Performance of Algorithm 3 on six queueing networks

At the end of Section 4.3, the relationship of the GAE estimator (4.24) and the AME estimator (4.19) was discussed. In Figure 6 we compare the learning curves of PPO algorithm 3 on the 6-classes network empirically showing the advantage of using the AMP method over GAE method. The learning curve for the GAE estimator is obtained by replacing the value function estimation in line 6 of Algorithm 3 with the GAE estimator (4.24).

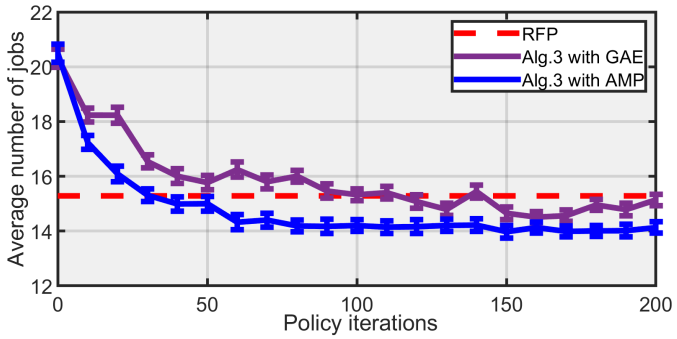


Figure 6: Learning curves from Algorithm 3 for the 6-class network. *The purple (blue) solid line* shows the performance of PPO policies obtained from Algorithm 3 in which the value function estimations are computed by the AMP method (4.18) (by the GAE method (4.24), correspondingly); *red dash line* –performance of the robust fluid policy (RFP).

5.3 Parallel servers network

In this section, we demonstrate that PPO Algorithm 3 is also effective for a stochastic processing network that is outside model class of multiclass queueing networks with little manual configuration of neural networks and hyper parameters. Consider a processing network system depicted in Figure 7 that has two independent Poisson input arrival flows, two servers, exponential service times and linear holding costs. This system is known as the *N-model network* and it first appeared in [30].

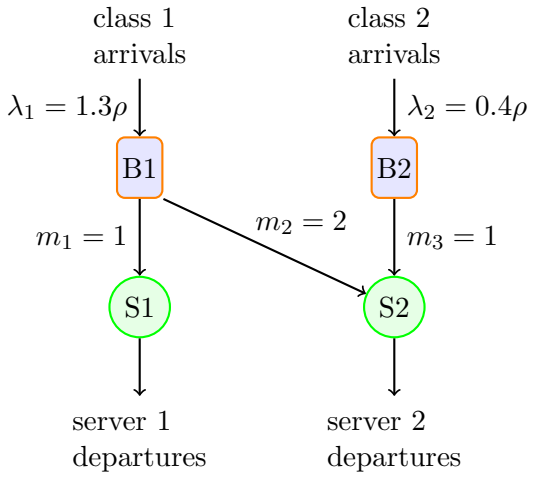


Figure 7: N-model network

Jobs of class i arrive according to a Poisson process at an average rate of λ_i such that $\lambda_1 = 1.3\rho$ and $\lambda_2 = 0.4\rho$ per unit time, where $\rho = 0.95$ is a parameter that specifies the traffic intensity. Each job requires a single service before it departs, and class 1 can be processed by either server 1 or server 2, whereas class 2 can be processed only by server 2. The jobs are processed by three different activities, as follows:

$$\begin{aligned}&\text{activity 1} = \text{processing of class 1 jobs by server 1},\\ &\text{activity 2} = \text{processing of class 1 jobs by server 2},\\ &\text{activity 3} = \text{processing of class 2 jobs by server 2}.\end{aligned}$$

We assume that the service at both servers is preemptive and work-conserving, particularly we assume that activity 2 happens only if there is at least one class 1 job in the system.

The service times for activity i are exponentially distributed with mean m_i, where $m_1 = m_3 = 1$ and $m_2 = 2$. The holding costs are continuously incurred at a rate of h_j for each class j job that

remains within the system, with the specific numerical values $h_1 = 3$ and $h_2 = 1$. We note that all model parameter values correspond to those in [30].

We define $x = (x_1, x_2) \in \mathcal{X}$ as a system state, where x_j is number of class j jobs at the system. We use uniformization to convert the continuous-time control problem to a discrete-time control problem. Under control $a = 1$ (class 1 has preemption high priority for the second server) the transition probabilities are given by

$$
\begin{aligned}
&P\left((x_1+1, x_2)|(x_1, x_2)\right) = \frac{\lambda_1}{\lambda_1+\lambda_2+\mu_1+\mu_2+\mu_3},\\
&P\left((x_1, x_2+1)|(x_1, x_2)\right) = \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_1+\mu_2+\mu_3},\\
&P\left((x_1-1, x_2)|(x_1, x_2), a=1\right) = \frac{\mu_1\mathbb{I}_{\{x_1>0\}}+\mu_2\mathbb{I}_{\{x_1>1\}}}{\lambda_1+\lambda_2+\mu_1+\mu_2+\mu_3},\\
&P\left((x_1, x_2-1)|(x_1, x_2), a=1\right) = \frac{\mu_3\mathbb{I}_{\{x_2>0, x_1\leq 1\}}}{\lambda_1+\lambda_2+\mu_1+\mu_2+\mu_3},\\
&P\left((x_1, x_2)|(x_1, x_2)\right) = 1 - P\left((x_1+1, x_2)|(x_1, x_2)\right) -\\
&\quad - P\left((x_1, x_2+1)|(x_1, x_2)\right) - P\left((x_1-1, x_2)|(x_1, x_2)\right) - P\left((x_1, x_2)|(x_1, x_2), 1\right)
\end{aligned}
$$

where $\mu_i = 1/m_i$, $i = 1, 2, 3$.

Under control $a = 2$ (class 2 has high priority), the only change of transition probabilities are

$$
\begin{aligned}
&P\left((x_1-1, x_2)|(x_1, x_2), a=2\right) = \frac{\mu_1\mathbb{I}_{\{x_1>0\}}+\mu_2\mathbb{I}_{\{x_1>1, x_2=0\}}}{\lambda_1+\lambda_2+\mu_1+\mu_2+\mu_3},\\
&P\left((x_1, x_2-1)|(x_1, x_2), a=2\right) = \frac{\mu_3\mathbb{I}_{\{x_2>0\}}}{\lambda_1+\lambda_2+\mu_1+\mu_2+\mu_3}.
\end{aligned}
$$

The cost-to-go function is defined as $g(x) := h_1x_1 + h_2x_2 = 3x_1 + x_2$. The objective is to find policy $\pi_\theta, \theta \in \Theta$ that minimizes long-run average holding costs

$$
\lim_{N\to\infty}\frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1} g\left(x^{(k)}\right)\right],
$$

where $x^{(k)}$ is the system state after k timesteps.

We use the PPO Algorithm 3 to find a near-optimal policy. Along with a learning curve from Algorithm 3 we show the performance the best threshold policy with $T = 11$ and optimal policy, see Figure 8. The threshold policy has been proposed in [9]. Server 2 operating under the threshold policy gives priority for class 1 job if number of class 1 jobs in the system is larger than a fixed threshold T.

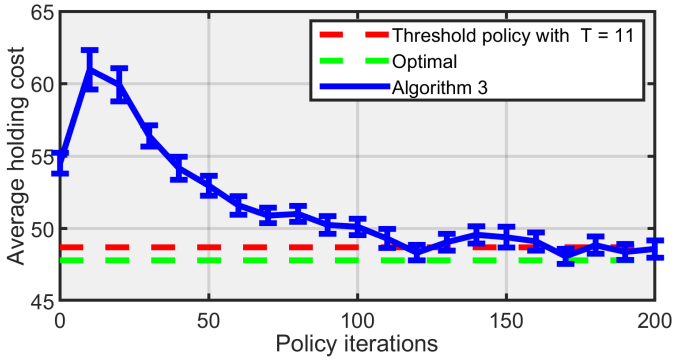
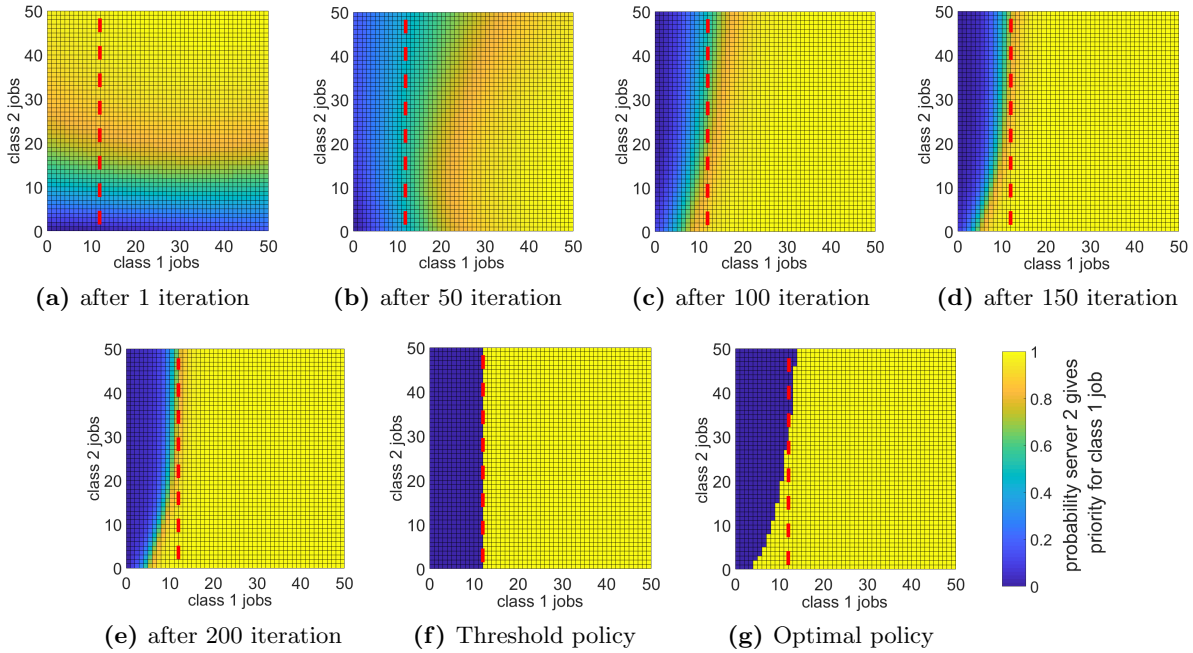


Figure 8: Learning curves from Algorithm 3 for the N-model network. *The blue solid line* shows the performance of PPO policies obtained from Algorithm 3; *red dash line* –performance of the threshold policy with $T = 11$; *greed dash line* –performance of the optimal policy.

In Figure 9 we show the control of randomized PPO policies obtained after 1, 50, 100, 150, and 200 algorithm iterations. For each policy we depict the probability distribution over two possible actions for states $x \in \mathcal{X}$ such that $0 \leq x_j \leq 50$, $j = 1, 2$. The PPO policies are compared with the optimal and threshold policies.



(a) after 1 iteration **(b)** after 50 iteration **(c)** after 100 iteration **(d)** after 150 iteration

(e) after 200 iteration **(f)** Threshold policy **(g)** Optimal policy

Figure 9: Evolution of PPO policies over the curse of learning and its comparison with threshold and optimal policies. The probability that server 2 gives priority to class 1 jobs is shown by a color gradient for system states that have less than 50 jobs in each buffer: *yellow color* - server 2 gives priority for class 1 jobs, *blue color* - server 2 gives priority for class 2 jobs. The *red dash line* represents the threshold policy with $T = 11$.

6 Conclusion

In this study we have presented a method of optimizing the long-run average performance in queueing network control problems. For long-run average cost objective, we provide the theoretical justification for extending the PPO algorithm in [71] for Markov decision problems with infinite state space and unbounded costs.

A key step in a successful implement ion of the PPO algorithm for long-run average cost MDP problem is to have an effective Monte Carlo estimate of the relative value function in each policy iteration. We have numerically observed that (i) introducing a proper discount factor has the most impact in variance reduction of an estimator, even though the discount factor introduces biases; (ii) introducing control variate via approximating martingale-process (AMP) has a significant variance reduction, and (iii) exploring regenerative estimators has also a significant variance reduction when the system load varies from light to moderate. When transition probabilities are known, we demonstrate that the AMP estimation is preferred over GAE estimation proposed in [69].

Our numerical results show that PPO Algorithm 3 produces almost-optimal performance for the criss-cross network and accomplish long-run average performance within 1% from the optimal, see Table 2. In large-size networks our algorithm leads to effective scheduling policies that outperform or perform comparable to known alternatives. In the extended six-class queueing network PPO polices outperform the robust fluid policies on average by 10%, see Table 4. In all our experiments we have used fixed set of hyperparameters. The algorithm can be applied for a wide class of processing networks control problems as described in [23]. As an example, we provide the numerical experiment for the N-model network.

Complexity of the queueing control optimization problems highly depends not only on the network topology, but on the traffic intensity. Therefore, the complexity of any reinforcement learning algorithms

grows as the network traffic intensity increases. We believe that this feature makes multiclass queueing networks potentially good benchmark problems to test reinforcement learning methods.

Acknowledgement

This research is supported in part by the National Science Foundation Grant CMMI-1537795.

A Proofs of the theorems in Section 3

We believe the claim of Lemma 7 should be a well-known mathematical fact, but we have not found its proof in any textbook. For completeness, we present it here.

Lemma 7. *Let $\mathcal{M}_{\mathcal{X},\mathcal{X}}$ be a set of all matrices on the countable space $\mathcal{X}\times\mathcal{X}$. The operator norm $\|\cdot\|_{\mathcal{V}}$ on $\mathcal{M}_{\mathcal{X}\times\mathcal{X}}$ is equivalent to the operator norm induced from vector norm $\|\cdot\|_{1,\mathcal{V}}$ and to the operator norm induced from vector norm $\|\cdot\|_{\infty,\mathcal{V}}$ in the following sense:*

$$\|T\|_{\mathcal{V}} = \sup_{\nu:\|\nu\|_{1,\mathcal{V}}=1} \|\nu T\|_{1,\mathcal{V}} = \sup_{h:\|h\|_{\infty,\mathcal{V}}=1} \|Th\|_{\infty,\mathcal{V}} \quad \text{for any } T\in\mathcal{M}_{\mathcal{X}\times\mathcal{X}},$$

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

Furthermore, for any vectors ν_1,ν_2 on $\mathcal{X}$ and matrices $T_1,T_2\in\mathcal{M}_{\mathcal{X}\times\mathcal{X}}$ the following inequalities hold:

$$\|\nu_1^T T\nu_2\|_{\mathcal{V}} \le \|\nu_1\|_{1,\mathcal{V}}\|T\|_{\mathcal{V}}\|\nu_2\|_{\infty,\mathcal{V}} \tag{A.1}$$

and

$$\|T_1T_2\|_{\mathcal{V}} \le \|T_1\|_{\mathcal{V}}\|T_2\|_{\mathcal{V}}. \tag{A.2}$$

Proof. First, we show that $\sup_{h:\|h\|_{1,\mathcal{V}}=1} \|Th\|_{1,\mathcal{V}} = \|T\|_{\mathcal{V}}$. On the one hand,

$$\begin{aligned}
\sup_{\nu:\|\nu\|_{1,\mathcal{V}}=1} \|\nu T\|_{1,\mathcal{V}} &= \sup_{\nu:\mathcal{X}\to\mathbb{R}} \frac{1}{\|\nu\|_{1,\mathcal{V}}}\|\nu T\|_{1,\mathcal{V}} \\
&\ge \sup_{\nu\in\{e_x\}} \frac{1}{\|\nu\|_{1,\mathcal{V}}}\|\nu T\|_{1,\mathcal{V}} \\
&= \sup_{x\in\mathcal{X}} \frac{1}{\mathcal{V}(x)} \sum_{y\in X} \mathcal{V}(y)|T(x,y)| = \|T\|_{\mathcal{V}},
\end{aligned}$$

where in second step we choose a set $\{e_x\}$ of unit vectors $e_x=(0,...,0,1,0,...)$, which xth coordinate is 1 and rest ones are 0s.

On the other hand,

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

Similarly one can show the equivalency of $\sup_{h:\|h\|_{\infty,\mathcal{V}}=1} \|Th\|_{\infty,\mathcal{V}}$ and $\|T\|_{\mathcal{V}}$ norms.

Inequalities (A.1) and (A.2) follow from properties of a linear operator norm. □

Proof of Lemma 3. We define vector $h := Z\big(g - (\mu^T g)e\big)$. Matrix Z has a finite $\mathcal{V}-$norm, therefore the inverse matrix of Z is unique and equal to $I - P + \Pi$. Then, by definition vector h satisfies

$$(I - P + \Pi)h = g - (\mu^T g)e. \tag{A.3}$$

Multiplying both sides of (A.3) by μ we get $\mu^T h = 0$ (and $\Pi h = 0$). Hence, vector h is a solution of Poisson's equation (3.3) such that $\mu^T h = 0$. It follows from Lemma 2 that $h = h^{(f)}$. □

Proof of Lemma 4. We denote

$$U_{\theta,\eta} := (P_\theta - P_\eta)Z_\eta \tag{A.4}$$

and define matrix $H_{\theta,\eta}$ as

$$H_{\theta,\eta} := \sum_{k=0}^{\infty} U_{\theta,\eta}^k \tag{A.5}$$

Convergence in $\mathcal{V}$-weighted norm in definition (A.5) follows from assumption $\|U_{\theta,\eta}\|_{\mathcal{V}} < 1$.

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

$$\mu_\theta^T = \mu_\eta^T H_{\theta,\eta} \tag{A.6}$$

Let $\nu^T := \mu_\eta^T H_{\theta,\eta}$. We use $e = (1, 1, ..., 1, ..)^T$ to denote the unit vector. We first verify that $\nu^T e = \sum\limits_{x\in\mathcal{X}} \nu(x) = 1$. We note that $Z_\eta e = e$. Then

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

Next we verify that $\nu^T P_\theta = \nu^T$. We prove it by first assuming that

$$P_\theta - P_\eta + U_{\theta,\eta}P_\eta = U_{\theta,\eta} \tag{A.7}$$

holds. Indeed,

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

It remains to prove (A.7). Indeed,

$$\begin{aligned}
P_\theta - P_\eta + U_{\theta,\eta}P_\eta &= P_\theta - P_\eta + (P_\theta - P_\eta)Z_\eta P_\eta \\
&= (P_\theta - P_\eta)(I + Z_\eta P_\eta) \\
&= (P_\theta - P_\eta)(I - \Pi_\eta + Z_\eta P_\eta) \\
&= (P_\theta - P_\eta)(I - Z_\eta \Pi_\eta + Z_\eta P_\eta) \\
&= (P_\theta - P_\eta)Z_\eta \\
&= U_{\theta,\eta},
\end{aligned}$$

where the second equality follows from

$$\Big((P_\theta - P_\eta)Z_\eta\Big)P_\eta = (P_\theta - P_\eta)\Big(Z_\eta P_\eta\Big),$$

which holds by [43, Corollary 1.9], third eqaulity holds due to $(P_\theta - P_\eta)\Pi_\eta = 0$, forth equality holds from the fact that $Z_\eta\Pi_\eta = \Pi_\eta$ and the fifth equality follows from $I = Z_\eta(I + \Pi_\eta - P_\eta) = Z_\eta + Z_\eta\Pi_\eta - Z_\eta P_\eta$.

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

□

Proof of Theorem 1. Under assumption $\|U_{\theta,\eta}\|_\mathcal{V} = D_{\theta,\eta} < 1$ operator $H_{\theta,\eta} := \sum\limits_{k=0}^{\infty} U_{\theta,\eta}^k$ is well-defined and

$$\|H_{\theta,\eta}\|_\mathcal{V} \le \frac{1}{1 - D_{\theta,\eta}}. \tag{A.8}$$

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 (A.6). We get $\mu_\theta^T\mathcal{V} = \mu_\eta H_{\theta,\eta}\mathcal{V} \le \|H_{\theta,\eta}\|_\mathcal{V}(\mu_\eta^T\mathcal{V}) < \infty$.

The following equation has been proved for the finite state space in [72, Theorem 2], see also [27]:

$$Z_\theta = Z_\eta H_{\theta,\eta} - \Pi_\eta H_{\theta,\eta}[P_\theta - P_\eta]Z_\eta^2 H_{\theta,\eta}. \tag{A.9}$$

We have shown that all matrices in (A.9) are well-defined and one can directly follow the proof of [72, Theorem 2] to generalize the statement for the countable state space.

The long-run average costs difference is equal to

$$\begin{aligned}
\mu_\theta^T g - \mu_\eta^T g &= \mu_\theta^T g + \mu_\theta^T\left((P_\theta - I)h_\eta\right) - \eta_\eta \\
&= \mu_\theta^T(g + P_\theta h_\eta - h_\eta) - \eta_\eta \\
&= \mu_\eta^T(g - \mu_\eta^T g e + P_\theta h_\eta - h_\eta) + (\mu_\theta^T - \mu_\eta^T)(g + P_\theta h_\eta - h_\eta) \\
&= \mu_\eta^T(g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta) + (\mu_\theta^T - \mu_\eta^T)(g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta).
\end{aligned}$$

Now we are ready to bound the last term:

$$\begin{aligned}
\left|(\mu_\theta^T - \mu_\eta^T)(g - \eta_\eta e + P_\theta h_\eta - h_\eta)\right| &\le \|\mu_\theta - \mu_\eta\|_{1,\mathcal{V}}\ \|g - (\mu_\eta^T g)e + P_\theta h_\eta - h_\eta\|_{\infty,\mathcal{V}} \\
&= \|\mu_\theta - \mu_\eta\|_{1,\mathcal{V}}\ \|(P_\theta - P_\eta)h_\eta\|_{\infty,\mathcal{V}} \\
&= \|\mu_\theta - \mu_\eta\|_{1,\mathcal{V}}\ \|(P_\theta - P_\eta)Z_\eta(g - (\mu_\eta^T g)e)\|_{\infty,\mathcal{V}} \\
&\le \|\mu_\theta - \mu_\eta\|_{1,\mathcal{V}}\ \|(P_\theta - P_\eta)Z_\eta\|_\mathcal{V}\ \|g - (\mu_\eta^T g)e\|_{\infty,\mathcal{V}} \\
&= D_{\theta,\eta}\|\mu_\theta - \mu_\eta\|_{1,\mathcal{V}}\ \|g - (\mu_\eta^T g)e\|_{\infty,\mathcal{V}} \\
&= D_{\theta,\eta}\|\mu_\eta^T(P_\eta - P_\theta)Z_\theta\|_{1,\mathcal{V}}\ \|g - (\mu_\eta^T g)e\|_{\infty,\mathcal{V}},
\end{aligned}$$

where the first inequality follows from Lemma 7, second equality follows from Lemma 3 and the last equality is held due to

$$\mu_\theta^T - \mu_\eta^T = \mu_\eta^T(P_\eta - P_\theta)Z_\theta$$

from (A.6).

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

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

where the first equality follows from $Z_\eta(I + \Pi_\eta - P_\eta) = I$, the second equality follows from (A.9), the second inequality is held due to Lemma 7 and the last inequality follows from (3.8), (A.8).

□

Proof of Lemma 5.

$$
\begin{aligned}
\|(P_\theta - P_\eta)Z_\eta\|_{\mathcal{V}} &\le \|P_\theta - P_\eta\|_{\mathcal{V}} \|Z_\eta\|_{\mathcal{V}} \\
&= \|Z_\eta\|_{\mathcal{V}} \sup_{x\in\mathcal{X}} \frac{1}{\mathcal{V}(x)} \sum_{y\in\mathcal{X}} |P_\theta - P_\eta|_{x,y} \mathcal{V}(y) \\
&= \|Z_\eta\|_{\mathcal{V}} \sup_{x\in\mathcal{X}} \frac{1}{\mathcal{V}(x)} \sum_{y\in\mathcal{X}} |\sum_{a\in\mathcal{A}} P(y|x,a)\pi_\theta(a|x) - \sum_{a\in\mathcal{A}} P(y|x,a)\pi_\eta(a|x)| \mathcal{V}(y) \\
&\le \|Z_\eta\|_{\mathcal{V}} \sup_{x\in\mathcal{X}} \frac{1}{\mathcal{V}(x)} \sum_{y\in\mathcal{X}} \sum_{a\in\mathcal{A}} P(y|x,a)|\pi_\theta(a|x) - \pi_\eta(a|x)| \mathcal{V}(y) \\
&= \|Z_\eta\|_{\mathcal{V}} \sup_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} |\pi_\theta(a|x) - \pi_\eta(a|x)| \frac{\sum\limits_{y\in\mathcal{X}} P(y|x,a)\mathcal{V}(y)}{\mathcal{V}(x)} \\
&= \|Z_\eta\|_{\mathcal{V}} \sup_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} \Big|\frac{\pi_\theta(a|x)}{\pi_\eta(a|x)} - 1\Big| G(x,a)
\end{aligned}
$$

□

B Proofs of the theorems in Section 4

We consider the Poisson's equation for a Markov chain with the transition kernel P, stationary distribution μ and cost function $g : \mathcal{X} \to \mathbb{R}$:

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

which admits a solution

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

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

In the large-size systems regenerative cycles can be long. We propose to change the original dynamics and increase probability of transition to the regenerative state x^* from each state $x \in \mathcal{X}$.

Let $P(y|x)$ be an original transition probability from state x to state y, for each $x, y \in \mathcal{X}$. We consider a new Markov reward process with cost function g and a modified transition kernel $\tilde{P}$:

$$
\begin{cases}
\tilde{P}(y|x) = \gamma P(y|x) & \text{for } y \neq x^*, \\
\tilde{P}(x^*|x) = \gamma P(x^*|x) + (1-\gamma),
\end{cases}
\tag{B.1}
$$

for each $x \in \mathcal{X}$.

We have modified the transition kernel so that the probability of transition to the regenerative state x^* is at least $1-\gamma$ from any state.

The Poisson's equation for the modified problem is equal to:

$$
g(x) - \tilde{\mu}^T g + \sum_{y\in\mathcal{X}} \tilde{P}(y|x)\tilde{h}(y) - \tilde{h}(x) = 0, \text{ for each } x \in \mathcal{X},
\tag{B.2}
$$

where $\tilde{\mu}$ is the stationary distribution of the Markov chain $\tilde{P}$.

Equation (B.2) admits a solution

$$
\tilde{h}^{(x^*)}(x) := \mathbb{E}\left[\sum_{k=0}^{\tilde{\sigma}(x^*)-1} \Big(g(x^{(k)}) - \tilde{\mu}^T g\Big) \mid x^{(0)} = x\right] \text{ for each } x \in \mathcal{X},
\tag{B.3}
$$

where $x^{(k)}$ is the state of the Markov chain with transition matrix $\tilde{P}$ after k timesteps, $\tilde{\sigma}(x^*) = \min\big\{k > 0 \mid x^{(k)} = x^*\big\}$. According to the new dynamics the regeneration happens more frequently and solution (B.3) can be estimated by less replications of the regenerative simulation.

Lemma 8. *Consider the Poisson's equation for the Markov chain with the transition kernel $\tilde{P}$ defined by (B.1), stationary distribution $\tilde{\mu}$ and cost function $g : \mathcal{X} \to \mathbb{R}$:*

$$g(x) - \tilde{\mu}^T g + \sum_{y \in \mathcal{X}} \tilde{P}(y|x)\tilde{h}(y) - \tilde{h}(x) = 0, \text{ for each } x \in \mathcal{X}. \tag{B.4}$$

Equation (B.4) admits solutions:

$$J^{(\gamma)}(x) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(g(x^{(k)}) - \mu^T g\right) \mid x^{(0)} = x\right] \text{ for each } x \in \mathcal{X},$$

and

$$V^{(\gamma)}(x) := \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) \mid x^{(0)} = x\right] \text{ for each } x \in \mathcal{X},$$

where $x^{(k)}$ is the state of the Markov chain with transition matrix P after k timesteps.

Proof. We substitute the definition of $\tilde{P}$ (B.1) and rewrite equation (B.4) as

$$g(x) - (\tilde{\mu}^T g - (1-\gamma)\tilde{h}(x^*)) + \gamma \sum_{y \in \mathcal{X}} P(y|x)\tilde{h}(y) - \tilde{h}(x) = 0, \text{ for each } x \in \mathcal{X}. \tag{B.5}$$

Equation (B.5) admits infinitely many solutions, but one can specify a unique solution fixing $\tilde{h}(x^*)$. Next, we consider two options.

First, let $\tilde{h}(x^*) = \frac{1}{1-\gamma}(\mu^T g - \tilde{\mu}^T g)$. Then the Poisson's equation (B.5) becomes

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

and admits solution

$$J^{(\gamma)}(x) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(g(x^{(k)}) - \mu^T g\right) \mid x^{(0)} = x\right] \text{ for each } x \in \mathcal{X}.$$

Now, let $\tilde{h}(x^*) = 0$ in equation (B.5). We note that $\tilde{\mu}^T g = r(x^*)$, where $r(x^*) = (1-\gamma)\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k g\left(x^{(k)}\right) \mid x^{(0)} = x^*\right]$ is a present discounted value at x^*.

We get the Poisson's equation (4.15)

$$g(x) - r(x^*) + \gamma \sum_{y \in \mathcal{X}} P(y|x)\tilde{h}(y) - \tilde{h}(x) = 0, \text{ for each } x \in \mathcal{X},$$

which admits solution (4.10)

$$V^{(\gamma)}(x) := \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) | x^{(0)} = x\right] \text{ for each } x \in \mathcal{X}.$$

Indeed,

$$\begin{aligned}
V^{(\gamma)}(x) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) | x^{(0)} = x\right] \\
&= \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) | x^{(0)} = x\right] + \mathbb{E}\left[\gamma^{\sigma(x^*)} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) | x^{(0)} = x^*\right]\right] \\
&= \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) | x^{(0)} = x\right].
\end{aligned}$$

□

Lemma 9. *Consider irreducible, aperiodic Markov chain with transition matrix P that satisfies drift condition (3.1). Let*

$$V^{(\gamma)}(x) := \mathbb{E}\left[\sum_{k=0}^{\sigma(x^*)-1} \gamma^k \left(g(x^{(k)}) - r(x^*)\right) \mid x^{(0)} = x\right] \text{ for each } x \in \mathcal{X}$$

be the corresponding regenerative discounted relative value function for one-step cost function g, such that $|g(x)| < \mathcal{V}(x)$ for each $x \in \mathcal{X}$ and regenerative state $x^* \in \mathcal{X}$. Let $h^{(x^*)}$ be a solution of the Poisson's equation (3.3) defined by (4.1).

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

$$\left|V^{(\gamma)}(x) - h^{(x^*)}(x)\right| \le \frac{rR(1-\gamma)}{(1-r)(1-\gamma r)}(\mathcal{V}(x) + \mathcal{V}(x^*)) \text{ for each } x \in \mathcal{X}.$$

Proof. By Lemma 8 function $V^{(\gamma)}$ is a solution of Poisson's equation (B.2). Consider the discounted value function $J^{(\gamma)} = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(g\left(x^{(k)}\right) - \mu^T g\right) \mid x^{(0)} = x\right]$ that is another solution of Poisson's equation (B.2).

Then, for an arbitrary $x \in \mathcal{X}$,

$$\left|V^{(\gamma)}(x) - h^{(x^*)}(x)\right| \le \left|J^{(\gamma)}(x) - h^{(f)}(x)\right| + \left|V^{(\gamma)}(x) - h^{(x^*)}(x) - \left(J^{(\gamma)}(x) - h^{(f)}(x)\right)\right|, \tag{B.6}$$

where $h^{(f)}$ is the fundamental solution of Poisson's equation (3.3).

First, let us bound $\left|J^{(\gamma)}(x) - h^{(f)}(x)\right|$:

$$\begin{aligned}
|J^{(\gamma)}(x) - h(x)| &\le \sum_{t=0}^{\infty} |\gamma^t - 1| \left|\sum_{y \in \mathcal{X}} P^t(y|x)(g(y) - \mu^T g)\right| \\
&\le R\mathcal{V}(x) \sum_{t=0}^{\infty} (1-\gamma^t) r^t \\
&= R\mathcal{V}(x) r \frac{1-\gamma}{(1-r)(1-\gamma r)},
\end{aligned}$$

where the second inequality follows from (3.2).

We note that $V^{(\gamma)}$ and $J^{(\gamma)}$ are solutions of Poisson's equation (B.2), hence

$$J^{(\gamma)}(x) - V^{(\gamma)}(x) = J^{(\gamma)}(x^*) - V^{(\gamma)}(x^*) = \frac{1}{1-\gamma}(\mu^T g - r(x^*)) \text{ for each } x \in \mathcal{X}.$$

Similarly, $h^{(f)}(x) - h^{(x^*)}(x) = h^{(f)}(x^*)$ for each $x \in \mathcal{X}$.

Now we will bound the last term in inequality (B.6):

$$\begin{aligned}
&\left|V^{(\gamma)}(x) - h^{(x^*)}(x) - \left(J^{(\gamma)}(x) - h^{(f)}(x)\right)\right| \\
&\qquad = \left|h^{(f)}(x^*) - \frac{1}{1-\gamma}(r(x^*) - \mu^T g)\right| \\
&\qquad = \left|\sum_{t=0}^{\infty} \sum_{y \in \mathcal{X}} P^t(y|x^*)(g(y) - \mu^T g) - \sum_{t=0}^{\infty} \sum_{y \in \mathcal{X}} \gamma^t P^t(y|x^*)(g(y) - \mu^T g)\right| \\
&\qquad = \left|\sum_{t=0}^{\infty} \sum_{y \in \mathcal{X}} (1-\gamma^t) P^t(y|x^*)(g(y) - \mu^T g)\right| \\
&\qquad \le \sum_{t=0}^{\infty} |1-\gamma^t| \left|\sum_{y \in \mathcal{X}} P^t(y|x^*)(g(y) - \mu^T g)\right| \\
&\qquad \le R\mathcal{V}(x^*) r \frac{1-\gamma}{(1-r)(1-\gamma r)},
\end{aligned}$$

where the last inequality is held due to (3.2). □

C Maximal Stability of Proportionally Randomized Policy

In this section we argue that a discrete-time MDP obtained by uniformization of the multiclass queueing network SMDP model is stable under proportionally randomized (PR) policy if the load conditions (2.9) are satisfied. We illustrate the proof for criss-cross queueing network. Let $x \in \mathbb{Z}_+^3$ be a state for the discrete-time MDP. The proportionally randomized policy π is given by

$$\pi(x) = \begin{cases} \left(\frac{x_1}{x_1+x_3}, 1, \frac{x_3}{x_1+x_3}\right) & \text{if } x_2 \geq 1 \text{ and } x_1 + x_3 \geq 1, \\ \left(\frac{x_1}{x_1+x_3}, 0, \frac{x_3}{x_1+x_3}\right) & \text{if } x_2 = 0 \text{ and } x_1 + x_3 \geq 1, \\ (0,1,0) & \text{if } x_2 \geq 1 \text{ and } x_1 + x_3 = 0, \\ (0,0,0) & \text{if } x_2 = 0 \text{ and } x_1 + x_3 = 0. \end{cases}$$

Recall the transition probabilities $\tilde{P}$ defined by (2.5). The discrete-time MDP operating under policy π is a DTMC. Next we specify its transition matrix. For $x \in \mathbb{Z}_+^3$ with $x_1 \geq 1$, $x_3 \geq 1$, and $x_2 \geq 1$,

$$P(y|x) = \pi_1(x)\tilde{P}\big(y|x,(1,2)\big) + \pi_3(x)\tilde{P}\big(y|x,(3,2)\big) \quad \text{for each } y \in \mathbb{Z}_+^3.$$

For $x \in \mathbb{Z}_+^3$ on the boundary with $x_1 \geq 1$, $x_3 \geq 1$, and $x_2 = 0$,

$$P(y|x) = \pi_1(x)\tilde{P}\big(y|x,(1,0)\big) + \pi_3(x)\tilde{P}\big(y|x,(3,0)\big) \quad \text{for each } y \in \mathbb{Z}_+^3.$$

For $x \in \mathbb{Z}_+^3$ on the boundary with $x_1 = 0$, $x_3 \geq 1$, and $x_2 \geq 1$,

$$P(y|x) = \tilde{P}\big(y|x,(3,2)\big) \quad \text{for each } y \in \mathbb{Z}_+^3.$$

The transition probabilities for other boundary cases can be written down similarly. One can verify that

$$P\big((x_1+1,x_2,x_3)|x\big) = \frac{\lambda_1}{B}, \quad P\big((x_1,x_2,x_3+1)|x\big) = \frac{\lambda_3}{B}, \tag{C.1}$$

$$P\big((x_1-1,x_2+1,x_3)|x\big) = \frac{\mu_1}{B}\frac{x_1}{x_1+x_3} \quad \text{if } x_2 \geq 1, \tag{C.2}$$

$$P\big((x_1,x_2,x_3-1)|x\big) = \frac{\mu_3}{B}\frac{x_3}{x_1+x_3} \quad \text{if } x_3 \geq 1, \tag{C.3}$$

$$P\big((x_1,x_2-1,x_3)|x\big) = \frac{\mu_2}{B} \quad \text{if } x_2 \geq 1, \tag{C.4}$$

$$P(x|x) = 1 - \sum_{y \neq x} P(y|x), \tag{C.5}$$

where $B = \lambda_1 + \lambda_3 + \mu_1 + \mu_2 + \mu_3$. This transition matrix P is irreducible. Now consider the continuous-time criss-cross network operating under the head-of-line proportional-processor-sharing (HLPPS) policy that is defined in [16]. Under the HLPPS policy, the jobcount process $\{Z(t), t \geq 0\}$ is a CTMC. Under the load condition (2.1), [16] proves that the CTMC is positive recurrent. One can verify that the transition probabilities in (C.1)-(C.5) are identical to the ones for a uniformized DTMC of this CTMC. Therefore, the DTMC corresponding the transition probabilities (C.1)-(C.5) is positive recurrent, proving the stability of the discrete-time MDP operating under the proportionally randomized policy.

D Additional experimental results

In Remark 5 we discussed two possible biased estimators of the solution to Poisson's equation. In this section we compare the performance of the PPO algorithm with these two estimators. Two versions of line 7 of Algorithm 3 are considered: first version uses regenerative discounted value function (VF) estimator (4.19), second version uses discounted value function estimator (4.22). We apply two versions of the PPO algorithm for the criss-cross system operating under the balanced medium (B.M.) load regime. The queueing network parameter setting is identical to the one detailed in Section 5.1, except that the quadratic cost function $g(x) = x_1^2 + x_2^2 + x_3^2$ replaces the linear cost function that is used to minimize the long-run average cost, where x_i is a number of jobs in buffer i, $i = 1, 2, 3$.

The policy NN parameters θ_0 are initialized using Xavier initialization. We take the empty system state $x^* = (0,0,0)$ as a regeneration state. Each episode in each iteration starts at the regenerative

state and runs for 6000 time steps. The one-replication estimates of a value function (either regenerative discounted VF or discounted VF) are computed for the first $N = 5000$ steps at each episode. In this experiment we simulated $Q = 20$ episodes in parallel. The values of the rest hyperparameters (not mentioned yet) are the same as in Table 6.

In Figure 10 we compare the learning curves of PPO algorithm 3 empirically showing the advantage of using the regenerative discounted VF estimator over the discounted VF estimator when the system regeneration happens frequently.

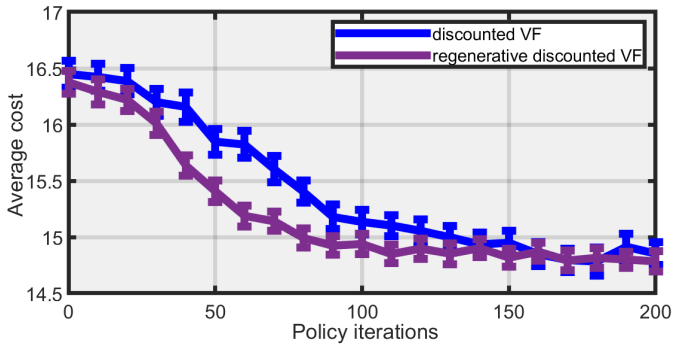


Figure 10: Learning curves from Algorithm 3 for the criss-cross network with B.M. load and quadratic cost function. *The purple (blue) solid line* shows the performance of PPO policies obtained from Algorithm 3 in which the solution to the Poisson's equation is estimated by the discounted VF estimator (4.22) (by the regenerative discounted VF estimator (4.19), correspondingly).

E Neural Network structure

In the experiments we parameterized the RL policy with a neural network. We use θ to denote the vector of weights and biases of the neural network. For a fixed parameter θ, the neural network outputs deterministically distribution $\pi_\theta(\cdot|x)$ over the action space for each state $x \in \mathcal{X}$. Therefore, the resulting policy π_θ is a randomized policy as explained at the end of Section 2.

To represent the policy we use a fully-connected feed-forward neural network with one input layer, three hidden layers with tanh activation functions, and one output layer. The input layer has J units, one for each job class, the first hidden layer consists of $10 \times J$ units, the third hidden layer has $10 \times L$, where L is number of stations in the queueing system. Number of units in the second hidden layer is a geometric mean of units in the first and third hidden layers, that is $10 \times \sqrt{LJ}$.

We use $z_j^{(k)}$ to denote the variable in the jth unit of hidden layer k, $k = 1, 2, 3$. Thus, our feed-forward neural network has the following representations.

$$z_j^{(1)} = h\Big(\sum_{i=1}^{J} A_{ji}^{(1)} x_i + b_j^{(1)}\Big), \quad j = 1, \dots, 10J,$$

$$z_j^{(2)} = h\Big(\sum_{i=1}^{10J} A_{ji}^{(2)} z_i^{(1)} + b_j^{(2)}\Big), \quad j = 1, \dots, 10\sqrt{LJ},$$

$$z_j^{(3)} = h\Big(\sum_{i=1}^{10\sqrt{LJ}} A_{ji}^{(3)} z_i^{(2)} + b_j^{(3)}\Big), \quad j = 1, \dots, 10\sqrt{L},$$

where $h : \mathbb{R} \to \mathbb{R}$ is the activation function given by $h(y) = \tanh(y)$ for each $y \in \mathbb{R}$.

We denote $p = (p_j)$ as the output vector, which is given by

$$p_j = \sum_{i=1}^{10\sqrt{L}} A_{ji}^{(4)} z_i^{(3)} + b_j^{(4)}, \quad j \in \mathcal{J},$$

and the output vector p is normalized via a *softmax function* into L probability distributions as follows

$$\pi_\theta(j|x) = \frac{\exp(p_j)}{\sum_{i \in \mathcal{L}(\ell)} \exp(p_i)} \quad \text{for each } j \in \mathcal{B}(\ell),\ \ell \in \mathcal{L}, \tag{E.1}$$

where the sets $\mathcal{J}$, $\mathcal{L}$, and $\mathcal{B}(\ell)$ are defined in Section 2.

The neural network parameter θ is the vector of weights A's and biases b's

$$\theta = \left(A^{(1)}, b^{(1)}, A^{(2)}, b^{(2)}, A^{(3)}, b^{(3)}, A^{(4)}, b^{(4)}\right),$$

which has dimension

$$K = \left(J \times 10J + 10J\right) + \left(10J \times 10\sqrt{LJ} + 10\sqrt{LJ}\right) + \left(10\sqrt{LJ} \times 10L + 10L\right) + \left(10L \times J + J\right).$$

For example, when $J = 21$ and $L = 7$, this dimension is approximately equal to 30203.

To represent the value function we use a neural network whose architecture is almost identical to the policy neural network except the following changes: the third hidden layer has 10 units. The number of units in the second hidden layer is $10 \times \sqrt{J}$. The output layer contains one unit with a linear activation function, which means that

$$V(x) = \sum_{i=1}^{10} A_i^{(4)} z_i^{(3)} + b^{(4)}.$$

For N-model processing network, considered in Section 5.3, the structure of policy and value NNs is the same as for MQNs described above, except the meaning of set $\mathcal{B}(\ell)$ in (E.1). Consider station $\ell \in \mathcal{L}$ of a processing network. The set $\mathcal{B}(\ell)$ includes a job class if and only if there is an activity such that station ℓ can process jobs from this class.

F Implementation Details and Experiment Parameters

We use Tensorflow v1.13.1 [1] to build a training routine of the neural networks and Ray package v0.6.6 [62] to maintain parallel simulation of actors. All experiments have been proceeded on a 2.7 GHz 96-core processor with 1510 GB of RAM.

The value and policy function parameters are optimized to minimize the corresponding loss functions (4.4), (4.6) by the Adaptive Moment Estimation (Adam) method [44]. The Adam method is an algorithm for mini-batch gradient-based optimization. Assume that N datapoints have been generated $D^{(1:N)} = \left\{(x^{(j)}, a^{(j)}, \hat{A}_j)\right\}_{j=1}^{N}$ to update the policy NN or $D^{(1:N)} = \left\{(x^{(j)}, \hat{V}_j)\right\}_{j=1}^{N}$ to update the value NN. The Adam algorithm runs for E epochs. The number of epochs is the number of complete passes through the entire dataset. In the beginning of a new epoch e the entire dataset is randomly reshuffled and divided into the batches with size m. Then each batch (indexed by n) is passed to the learning algorithm and the parameters of the neural networks $\theta = (\theta^1, ..., \theta^i, ..., \theta^K)$ are updated at the end of every such step according to

$$\theta_{n+1}^i = \theta_n^i - \varsigma \frac{1}{\sqrt{\hat{H}_n^i} + \varrho} \hat{G}_n^i,$$

where ς is a learning rate, $\hat{G}_n$ and $\hat{H}_n$ are moving average estimates of the first and second moments of the gradient respectively, $\varrho << 1$ is a constant.

The batch is used to compute the gradient of a loss function $\hat{L}(\theta, D)$, for example (4.6) :

$$g_n = \frac{1}{m} \nabla_\theta L\left(\theta, D_e^{(nm:nm+m)}\right), \tag{F.1}$$

where $D_e^{(nm:nm+m)}$ denotes the data segment in the nth batch of size m at epoch e.

The moving averages G_0 and H_0 are initialized as vectors of zeros at the first epoch. Then the Adam method updates the moving average estimates and includes bias corrections to account for their initialization at the origin:

$$\begin{cases} G_n = \beta_1 G_{n-1} + (1 - \beta_1) g_{n-1}, \\ \hat{G}_n = \frac{G_n}{1 - \beta_1^n}, \end{cases}$$

where $\beta_1 > 0$ with β_1^n denoting β_1 to the power n.

Similarly, the second moments are computed by

$$\begin{cases} H_n = \beta_2 H_{n-1} + (1-\beta_2) g_{n-1}^2, \\ \hat{H}_n = \frac{H_n}{1-\beta_2^n}, \end{cases}$$

where g_n^2 means the elementwise square, $\beta_2 > 0$ with β_2^n denoting β_2 to the power n.

Each subsequent epoch continues the count over n and keeps update the moving average estimates G_n, H_n starting from their final values of the latest epoch.

In Table 5, Table 6 we summarize our choice of PPO hyperparameters used for the experiments in Section 5. In Table 7 we provide the estimate of the running time of Algorithm 3

Parameter	Value
Clipping parameter (ϵ)	$0.2 \times \max[\alpha, 0.01]$
Num. of regenerative cycles per actor (N)	5,000
Num. of actors (Q)	50
Adam parameters for policy NN	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varrho = 10^{-8}$, $\varsigma = 5 \cdot 10^{-4} \times \max[\alpha, 0.05]$
Adam parameters for policy NN	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varrho = 10^{-8}$, $\varsigma = 2.5 \cdot 10^{-4}$
Num. of epochs (E)	3
Minibatch size in Adam method (m)	2048

Table 5: PPO hyperparameters used in Algorithm 1 and 2 for the experiments in Section 5.1. Parameter α decreases linearly from 1 to 0 over the course of learning: $\alpha = (I-i)/I$ on the ith policy iteration, $i = 0, 1, ..., I-1$.

Parameter	Value
Clipping parameter (ϵ)	$0.2 \times \max[\alpha, 0.01]$
Horizon (N)	50,000
Num. of actors (Q)	50
Adam parameters for policy NN	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varrho = 10^{-8}$, $\varsigma = 5 \cdot 10^{-4} \times \max[\alpha, 0.05]$
Adam parameters for policy NN	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varrho = 10^{-8}$, $\varsigma = 2.5 \cdot 10^{-4}$
Discount factor (β)	0.998
GAE parameter (λ)	0.99
Num. of epochs (E)	3
Minibatch size in Adam method (m)	2048

Table 6: PPO hyperparameters used in Algorithm 3 for the experiments in Sections 5.2, 5.3. Parameter α decreases linearly from 1 to 0 over the course of learning: $\alpha = (I-i)/I$ on the ith policy iteration, $i = 0, 1, ..., I-1$.

Num. of classes $3L$	Time (minutes)
6	0.50
9	0.73
12	1.01
15	2.12
18	4.31
21	7.61

Table 7: Running time of *one policy iteration* of the RL algorithm for the extended six-class network of Figure 4.

In the Algorithm 3 we use finite length episodes to estimate the expectation of the loss function in line 10. For each episode we need to specify an initial state. We propose sampling the initiate states from the set of states generated during previous policy iteration. 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 we can sample Q states uniformly at random from the episodes generated under policy π_{i-1} and save them in memory. Then these state are used as initial states for Q episodes under π_i policy. For policy π_0 all Q episodes starts from state $x = (0, .., 0)$.

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). 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.

[2] Abbasi-Yadkori, Y., Bartlett, P., and Malek, A. (2014). 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.

[3] Andradóttir, S., Heyman, D. P., and Ott, T. J. (1993). Variance Reduction Through Smoothing and Control Variates for Markov Chain Simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 3(3):167–189.

[4] Asmussen, S. (2003). *Applied Probability and Queues*. Springer New York.

[5] Ata, B. and Kumar, S. (2005). Heavy traffic analysis of open processing networks with complete resource pooling: asymptotic optimality of discrete review policies. *Annals of Applied Probability*, 15:331–391.

[6] Avram, F., Bertsimas, D., and Ricard, M. (1995). Fluid models of sequencing problems in open queueing networks: an optimal control approach. In Kelly, F. and Williams, R. J., editors, *Stochastic Networks*, volume 71, page 237. Springer, New York.

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

[8] Baxter, J. and Bartlett, P. L. (2001). Infinite-Horizon Policy-Gradient Estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

[9] Bell, S. L. and Williams, R. J. (2001). Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. *The Annals of Applied Probability*, 11(3):608–649.

[10] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47(1):253–279.

[11] Bertsimas, D., Gamarnik, D., and Anatoliy Rikun, A. (2011). Performance analysis of queueing networks via robust optimization. *Operations Research*, 59(2):455–466.

[12] Bertsimas, D., Nasrabadi, E., and Paschalidis, I. C. (2015). Robust Fluid Processing Networks. *IEEE Transactions on Automatic Control*, 60(3):715–728.

[13] Bertsimas, D., Paschalidis, I. C., and Tsitsiklis, J. (1994). Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. *The Annals of Applied Probability*, 4:43–75.

[14] Beutler, F. J. and Ross, K. W. (1987). Uniformization for semi-Markov decision processes under stationary policies. *Journal of Applied Probability*, 24(3):644–656.

[15] Bhatnagar, S. and Lakshmanan, K. (2012). An Online Actor-Critic Algorithm with Function Approximation for Constrained Markov Decision Processes. *Journal of Optimization Theory and Applications*, 153(3):688–708.

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

[17] Bramson, M. (1998). State space collapse with application to heavy traffic limits for multiclass queueing networks. *Queueing Systems. Theory and Applications*, 30:89–140.

[18] Chen, H. and Yao, D. (1993). Dynamic scheduling of a multiclass fluid network. *Operations Research*, 41:1104–1115.

[19] Chen, R.-R. and Meyn, S. (1999). Value iteration and optimization of multiclass queueing networks. *Queueing Systems*, 32:65–97.

[20] Chen, W., Huang, D., Kulkarni, A. A., Unnikrishnan, J., Zhu, Q., Mehta, P., Meyn, S., and Wierman, A. (2009). Approximate dynamic programming using fluid and diffusion approximations with applications to power management. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3575–3580.

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

[22] Dai, J. and Shi, P. (2019). Inpatient overflow: An approximate dynamic programming approach. *Manufacturing & Service Operations Management*, 21(4):894–911.

[23] Dai, J. G. and Harrison, J. M. (2020). *Processing Networks: Fluid Models and Stability*. Cambridge University Press, Cambridge, UK.

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

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

[26] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

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

[28] Harrison, J. M. (1988). Brownian models of queueing networks with heterogeneous customer populations. In Fleming, W. and Lions, P. L., editors, *Stochastic Differential Systems, Stochastic Control Theory and Applications*, volume 10 of *The IMA Volumes in Mathematics and Its Applications*, pages 147–186. Springer, New York, NY.

[29] Harrison, J. M. (1996). The bigstep approach to flow management in stochastic processing networks. In F. P. Kelly, S. Z. and Ziedins, I., editors, *Stochastic Networks: Theory and Applications*, volume 4 of *Lecture Note Series*, pages 57–90. Oxford University Press.

[30] Harrison, J. M. (1998). Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies. *The Annals of Applied Probability*, 8(3):822–848.

[31] Harrison, J. M. (2000). Brownian models of open processing networks: canonical representation of workload. *Ann. Appl. Probab.*, 10(1):75–103. corrections: **13**, 390–393 (2003) and **16**, 1703-1732 (2006).

[32] Harrison, J. M. (2002). Stochastic networks and activity analysis. In Suhov, Y. M., editor, *Analytic Methods in Applied Probability: In memory of Fridrikh Karpelevich*, volume 207 of *American Mathematical Society Translations: Series 2*, pages 53–76, Providence, RI. American Mathematical Society.

[33] Harrison, J. M. and Nguyen, V. (1993). Brownian models of multiclass queueing networks: current status and open problems. *Queueing Systems Theory Appl.*, 13(1-3):5–40.

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

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

[36] Henderson, S. G. and Meyn, S. P. (1997). 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. ACM Press.

[37] Henderson, S. G., Meyn, S. P., and Tadić, V. B. (2003). Performance evaluation and policy selection in multiclass networks. *Discrete Event Dyn. Syst.*, 13(1-2):149–189. Special issue on learning, optimization and decision making.

[38] Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Adry, A. M. (2020). A Closer Look at Deep Policy Gradients. In *ICLR*.

[39] Jaakkola, T., P. Singh, S., and Jordan, M. I. (1994). Reinforcement learning algorithm for partially observable Markov decision problems. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, pages 345–352.

[40] Jiang, S., Liu, Y., and Tang, Y. (2017). A unified perturbation analysis framework for countable Markov chains. *Linear Algebra and Its Applications*, 529:413–440.

[41] Kakade, S. (2001). Optimizing Average Reward Using Discounted Rewards. In *COLT '01/EuroCOLT '01*, pages 605–615.

[42] Kakade, S. and Langford, J. (2002). Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann Publishers.

[43] Kemeny, J. G., Snell, J. L., and Knapp, A. W. (1976). *Denumerable Markov Chains*. Springer New York.

[44] Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *ICLR*.

[45] Konda, V. R. and Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166.

[46] Kumar, P. R. (1993). Re-entrant lines. *Queueing Systems Theory Appl.*, 13(1-3):87–110.

[47] Kumar, S. and Kumar, P. R. (1994). Performance bounds for queueing networks and scheduling policies. *IEEE Transactions on Automatic Control*, AC-39:1600–1611.

[48] Kumar, S. and Kumar, P. R. (1996). Fluctuation smoothing policies are stable for stochastic reentrant lines. *Discrete Event Dynamic Systems*, 6:361–370.

[49] Lehnert, L., Laroche, R., and van Seijen, H. (2018). On Value Function Representation of Long Horizon Problems. *Thirty-Second AAAI Conference on Artificial Intelligence*.

[50] Lu, S. C. H., Ramaswamy, D., and Kumar, P. R. (1994). Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. *IEEE Transactions on Semiconductor Manufacturing*, 7(3):374–388.

[51] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., and Kim, D. I. (2019). Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Communications Surveys and Tutorials*, 21(4):3133–3174.

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

[53] Maguluri, S., Srikant, R., and Ying, L. (2012). Stochastic models of load balancing and scheduling in cloud computing clusters. In *INFOCOM, 2012 Proceedings IEEE*, pages 702–710.

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

[55] Martins, L. F., Shreve, S. E., and Soner, H. M. (1996). Heavy traffic convergence of a controlled, multiclass queueing system. *SIAM Journal on Control and Optimization*, 34(6):2133–2171.

[56] McKeown, N., Mekkittikul, A., Anantharam, V., and Walrand, J. (1999). Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8):1260–1267.

[57] Meyn, S. (1997). Stability and optimization of queueing networks and their fluid models. In Yin, G. G. and Zhang, Q., editors, *Mathematics of Stochastic Manufacturing Systems*, pages 175–199. American Mathematical Society, Providence, RI.

[58] Meyn, S. (2007). *Control techniques for complex networks*. Cambridge University Press.

[59] Meyn, S. and Tweedie, R. L. (2009). *Markov Chains and Stochastic Stability*. Cambridge University Press, Cambridge, 2nd edition.

[60] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[61] Moallemi, C., Kumar, S., and Van Roy, B. (2008). Approximate and data-driven dynamic programming for queueing networks. Preprint.

[62] Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., and Stoica, I. (2018). Ray: A Distributed Framework for Emerging AI Applications.

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

[64] OpenAI (2019). Dota 2 with Large Scale Deep Reinforcement Learning.

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

[66] Perkins, J. R. and Kumar, P. R. (1989). Stable distributed real-time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Transactions on Automatic Control*, AC-34:139–148.

[67] Puterman, M. L. (2005). *Markov decision processes : discrete stochastic dynamic programming*. Wiley-Interscience.

[68] Ramirez-Hernandez, J. A. and Fernandez, E. (2007). 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.

[69] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust Region Policy Optimization. In *Proceeding ICML'15*, pages 1889–1897.

[70] Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. (2016). High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*.

[71] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms.

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

[73] Serfozo, R. F. (1979). Technical Note—An Equivalence Between Continuous and Discrete Time Markov Decision Processes. *Operations Research*, 27(3):616–620.

[74] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

[75] Srikant, R. and Ying, L. (2014). *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge University Press, Cambridge, UK.

[76] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning : an introduction*. MIT press, 2nd edition.

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

[78] Veatch, M. H. (2015). Approximate linear programming for networks: Average cost bounds. *Computers & Operations Research*, 63:32–45.

[79] Wagner, H. M. (1975). *Principles of Operations Research: with applications to managerial decisions*. Englewood Cliffs, N.J. : Prentice-Hall, 2nd edition.

[80] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016). Sample Efficient Actor-Critic with Experience Replay.

[81] Willams, R. J. (1998). Some recent developments for queueing networks. In Accardi, L. and Heyde, C. C., editors, *Probability Towards 2000*, pages 340–456. Springer.

[82] Wu, Y., Mansimov, E., Liao, S., Grosse, R., and Ba, J. (2017). 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.