Intervention-Assisted Online Deep Reinforcement Learning for Stochastic Queuing Network Optimization

Intervention-Assisted Policy Gradient Methods for Online Stochastic Queuing Network Optimization

Jerrod Wigmore
jwigmore@mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Brooke Shrader
brooke.shrader@ll.mit.edu
MIT Lincoln Laboratory
Lexington, Massachusetts, USA

Eytan Modiano
modiano@mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

ABSTRACT

Deep Reinforcement Learning (DRL) offers a powerful approach to training neural network control policies for stochastic queuing networks (SQN). However, traditional DRL methods rely on offline simulations or static datasets, limiting their real-world application in SQN control. This work proposes Online Deep Reinforcement Learning-based Controls (ODRLC) as an alternative, where an intelligent agent interacts directly with a real environment and learns an optimal control policy from these online interactions. SQNs present a challenge for ODRLC due to the unbounded nature of the queues within the network resulting in an unbounded state-space. An unbounded state-space is particularly challenging for neural network policies as neural networks are notoriously poor at extrapolating to unseen states. To address this challenge, we propose an intervention-assisted framework that leverages strategic interventions from known stable policies to ensure the queue sizes remain bounded. This framework combines the learning power of neural networks with the guaranteed stability of classical control policies for SQNs. We introduce a method to design these intervention-assisted policies to ensure strong stability of the network. Furthermore, we extend foundational DRL theorems for intervention-assisted policies and develop two practical algorithms specifically for ODRLC of SQNs. Finally, we demonstrate through experiments that our proposed algorithms outperform both classical control approaches and prior ODRLC algorithms.

CCS CONCEPTS

• **Networks → Network algorithms**; • **Theory of computation → Reinforcement learning**.

KEYWORDS

Deep Reinforcement Learning, Queuing Networks, Lyapunov Stability, Online Learning and Control

ACM Reference Format:
Jerrod Wigmore, Brooke Shrader, and Eytan Modiano. 2024. Intervention-Assisted Policy Gradient Methods for Online Stochastic Queuing Network Optimization. In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '24), October 14–17, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641512.3686383

1 INTRODUCTION

In the field of Deep Reinforcement Learning (DRL), agents are often trained using offline simulated environments prior to being deployed on the real-world environment. DRL is a promising technique for training stochastic network control agents. However, the traditional simulation-based training paradigm has two major pitfalls. If the true network dynamics are not able to be accurately captured in simulation, then the agent trained on the simulation dynamics may perform poorly on the real-network. This is often referred to as the sim-to-real gap [13, 21]. Additionally, the policies of agents are often overfit to the training environments, and thus struggle to generalize to unseen environments in their deployment. [3, 27, 28]. In the context of SQN control, an agent would have to be trained on all possible dynamics of a particular network if the true network dynamics are not known with certainty. To overcome these limitations, we propose an Online Deep Reinforcement Learning-based Controls (ODRLC) paradigm for training SQN control agents. In ODRLC, an intelligent agent directly interacts with a real-world environment and learns to optimize its policy through these online interactions. This approach ensures the agent's policy is optimized for the true environment and does not require access to simulations prior to deployment.

Applying ODRLC to SQN control tasks presents significant challenges due to the unbounded state space. The infinite buffer model is commonly used in network control because of its analytical simplicity and the reality that network buffers are often extremely large. However, neural networks (NNs), which are use for policy and/or value function approximation in DRL, struggle with extrapolating or generalizing to unseen inputs [5, 11, 25]. In most control tasks, this poor generalization is not a major issue, as the state space is usually bounded, and sufficient exploration during training allows the agent to encounter nearly all possible states, minimizing the need for generalization.

However, in the context of an unbounded state space, especially with purely online training, this issue becomes critical. When an agent encounters an unseen state, it tends to take suboptimal actions

This work was supported by NSF grants CNS-2148183 and CNS-2106268.

MOBIHOC '24, October 14-17, 2024, Athens, Greece

ACM ISBN 979-8-4007-0521-2/24/10
https://doi.org/10.1145/3641512.3686383

due to the NN's poor extrapolation capabilities. These suboptimal actions increase cumulative costs and drive the agent further into unexplored regions, perpetuating a catastrophic feedback loop. This cycle, which we term as the *extrapolation loop* of unbounded state spaces, can be difficult to break, leading to continuously escalating costs as the agent interacts with the environment. In offline simulation-based training, this loop can be mitigated by periodically resetting the environment's state. However, in ODRLC, such resets may be infeasible or costly. Therefore, ensuring the environment's state remains within a finite region of the state space is crucial and closely related to the concept of strong stability often desired in SQN control algorithms.

This work addresses the challenge of ODRLC for SQN control tasks with unbounded state spaces. We propose a novel intervention-assisted agent framework that leverages a known stable policy to guarantee network stability while incorporating a NN policy for exploration and policy improvement. We prove these intervention-assisted policies are strongly stable, enabling their use for ODRLC. We extend key DRL theorems to the intervention-assisted setting, and introduce two practical ODRLC algorithms for SQN control. Our experiments show that these algorithms outperform existing SQN control and DRL-based methods in the ODRLC setting.

1.1 Related Works

Despite the wide applicability of queuing network models to various domains such as communication networks, manufacturing, and transportation, and their rich historical context in the controls literature, the integration of DRL for SQN controls remains a relatively underexplored avenue. The authors of [2] leverage DRL to optmize for delay in SQN control tasks that are similar to those studied in this paper, however their methods are not developed for the ODRLC setting. In [12], the authors use Deep Deterministic Policy Gradient to learn queuing network control policies via offline environments that provide explicit guarantees on the end-to-end delay of the policy. Each of these aforementioned works uses the standard offline simulation-based training paradigm of DRL and thus the algorithms do not extend well into the ODRLC setting.

The ODRLC setting is most similar to *continuing* or *average reward* Reinforcement Learning. In [29], the authors provides a novel policy improvement theorem for the average-reward case, which is fundamental in the development of trust-region methods including PPO. Ma et. al propose a unified policy improvement theorem that combines both the average reward and discounted reward settings in addition to addressing the Average Value Constraint problem that arises in average reward DRL [8]. The theoretical results in both [8, 29] hinge on the assumption that the state-space is finite and thus don't apply to environments with unbounded state-spaces such as queuing networks. In [11], the authors develop a Lyapunov-inspired reward shaping approach that encourages agents to learn a stable policy for online DRL over unbounded state-spaces.

Safe-DRL is a branch of DRL that incorporates interventions during training to maximize a reward function while adhering to safety constraints. For example, [23] employs human interventions in robotic navigation, while [22] uses automatic advantage-based interventions to enforce safety constraints in DRL algorithms designed for unconstrained tasks. These methods align with our approach, emphasizing the role of external guidance in ensuring stability and safety during training. Safe-DRL can also be formulated using Constrained Markov Decision Processes (MDPs), which introduce constraint functions alongside traditional MDPs [7]. However, satisfying these constraints in model-free DRL is challenging, as agents often violate them during exploration. Common techniques for addressing constrained MDPs in DRL include Lagrangian relaxation [17], projection-based optimization [26], and Lyapunov-based approaches [1]. While these methods share similarities with our intervention-assisted strategies, they address constrained optimization, whereas ODRLC for SQN control involves unconstrained optimization, with challenges arising from an unbounded state space and online learning requirements.

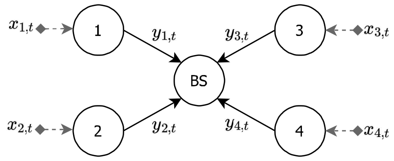

Figure 1: (SH2) An example of a single-hop wireless network. Packets arrive to each user according to user-dependent arrival distributions. All packets are destined for the base-station (BS). At each time step, the central controller chooses from one of the four links to activate.

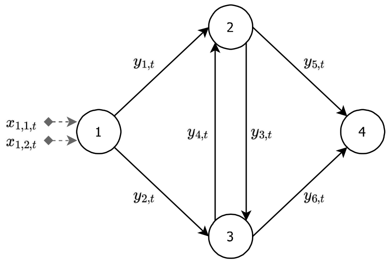

Figure 2: (MH1) An example of a multi-hop network. Packets from two different classes arrive to node 1 and all packets are destined for node 4. At each time-step, the central network controller must choose an action $\mathbf{a}_t$ which dictates how many packets from each class is transmitted over each directed link.

2 PRELIMINARIES

2.1 Stochastic Queuing Network Model

In this paper, we focus on the objective of delay minimization for general discrete time SQNs with Markovian dynamics. Under these models, the delay minimization task is well modeled by a Markov decision process. The networks under consideration consist of nodes connected by directed links, with each node hosting one or more queues equipped with unbounded buffers that store undelivered packets. Let $\mathbf{q}_t = \{q_{i,t}\}_{\forall i}$ denote the vector of all queue backlogs within the network, and let $\bar{q}_t = \sum_i q_{i,t}$ denote the *network backlog* which is the sum of backlogs across all queues within the network at the beginning of time t. The delay minimization objective is equivalent to minimizing the long-term average queue backlog [6]. Thus our objective function can be given as:

$$\min \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \bar{q}_t \tag{1}$$

For the SQN models considered, two random processes – stochastic packet arrivals and stochastic link capacities – govern the dynamics, along with the actions taken by a central network controller or agent. All packets belong to one of $K \geq 1$ traffic classes, where each class has an associated packet arrival distribution, arrival node, and a destination node. For any class k, $x_{k,t}$ packets arrives at time t, where $x_{k,t}$ is drawn i.i.d. from a finite discrete distribution $\mathbb{P}(x_k)$. The notation $\mathbf{x}_t = \{x_{k,t}\}_{\forall k}$ denotes the vector of all arrivals in time t.

We denoted the capacity of link m at time step t by $y_{m,t}$, which is also referred to as link m's link state. At the beginning of each time step $y_{m,t}$ is sampled i.i.d. from a finite discrete distribution $\mathbb{P}(y_{m,t})$. We use $\mathbf{y}_t = \{y_{m,t}\}_{\forall m}$ to denote the set of all link states over all M links at time t. We assume that arrival and link state distributions are mutually independent and independent of the overall network state. The network state at time t is captured by $\mathbf{s}_t = (\mathbf{q}_t, \mathbf{y}_t)$, encompassing both the queue states $\mathbf{q}_t$ and link states $\mathbf{y}_t$. At the start of each time step, the central network controller observes $\mathbf{s}_t$ and selects an action $\mathbf{a}_t$ from its policy π. The set of allowable actions depend on the specific network instance and its current state. An action $\mathbf{a}_t$ is a vector that specifies the amount of packets to transmit over each link for each class. The central controller aims to efficiently route all packets to their destinations by choosing which packets are transmitted on each link during each time step. A packet leaves the network once it arrives to its destination node. In Section 5, we test our algorithms on the following SQN control tasks:

2.1.1 Single-Hop Wireless Network Scheduling Task. For single-hop wireless scheduling problems, the network instance is described by a set of K user nodes, a base-station, and a single link between each user and the base-station. There is a traffic class associated with each user, and the base-station serves as the destination node for all user's traffic. To model wireless interference constraints, only a single link may be activated by the central controller in each time step. When the central controller selects user k's link at time step t, the number of packets that are transmitted to the base station is $a_{k,t} = \min\{q_{k,t}, y_{k,t}\}$. This constraint reflects that user k can only transmit the number of packets in its queue $q_{k,t}$, and cannot exceed the link's capacity $y_{k,t}$. In the reinforcement learning setting, the central controller aims to learn a state-dependent scheduling policy π that minimizes the long-term average backlog.

2.1.2 Multi-hop Network Control Task. For multi-hop networks, the network instance is described by a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denote the set of nodes and $\mathcal{E}$ denotes the set of directed links between nodes. Figure 2 shows an example a topology of a multi-hop network. Each of the K packet classes have an fixed source node and destination node. Each node maintains K queues, one for each class of traffic. At each time-step, the control policy observes the network state $\mathbf{s}_t = (\mathbf{q}_t, \mathbf{y}_t)$, and selects an action $\mathbf{a}_t = \{a_{m,k,t}\}_{\forall m,k}$ where where $a_{m,k,t} \geq 0$ is the number of class k packets to be transmitted on link m in time-step t. This action $\mathbf{a}_t$ must satisfy the following constraint:

$$\sum_k a_{m,k,t} \leq y_{m,t}, \quad \forall m = 1, ..., M \tag{2}$$

This link-capacity constraint means the total number of packets transmitted over each link must be less than the total capacity of the link. The central controller's decision $\mathbf{a}_t$ encompasses both a routing and scheduling decision. It determines not only the path each packet takes but also the order at which each class of traffic is transmitted over each link in every time step. Link activation constraints may also be included to model interference in wireless multi-hop networks, but we do not add this constraint for the experiments in Section 5

2.2 Markov Decision Process

Each SQN control tasks can be formulated as an average-cost Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, c, \rho_0)$ where:

(1) $\mathcal{S}$ represents the state-space, comprised of all possible states $\mathbf{s} = (\mathbf{q}, \mathbf{y})$. As its assumed each buffer within the network is unbounded, the state-space $\mathcal{S}$ is also unbounded.
(2) $\mathcal{A}$ denotes the action-space, which is comprised is the set of feasible control decisions and depends on the task. Additionally, we assume there is a set of valid actions $\mathcal{A}(\mathbf{s})$ for each $\mathbf{s} \in \mathcal{S}$ that is known by the central controller for each SQN control task.
(3) $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the probability of transitioning to state $\mathbf{s}'$ from state $\mathbf{s}$ after applying action $\mathbf{a}$. This transition probability captures the inherent uncertainty stemming from stochastic packet arrivals and stochastic link states.
(4) $c(\mathbf{s}_t)$ is the cost function. For delay minimization tasks, this equates to $c(\mathbf{s}_t) = \bar{q}_t$.
(5) $\rho_0(\mathbf{s}) = P(\mathbf{s}_0 = \mathbf{s})$ is the initial state distribution. For Section 5, we assume that at the beginning of each task, all queues are empty and each link state is sampled randomly from its respective distribution. However, none of our theoretical results depend on this assumption.

For an SQN control task, the central controller takes actions according to its policy π. We assume π is stochastic and $\pi(\mathbf{a}|\mathbf{s})$ denotes the probability of taking action $\mathbf{a}$ in state $\mathbf{s}$. We use $\pi(\cdot|\mathbf{s})$ to denote the distribution over all valid actions $\mathcal{A}(\mathbf{s})$ in state $\mathbf{s}$. In the ODRLC setting, π generates a single long trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, c_0, \mathbf{s}_1, \mathbf{a}_1, ...)$ where $\mathbf{s}_0 \sim \rho_0$, $\mathbf{a}_t \sim \pi(\cdot|\mathbf{s})$, $c_t = c(\mathbf{s}_t)$, and $\mathbf{s}_{t+1} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)$. Unlike the traditional offline simulation setting, the state cannot be externally reset. The policy π is updated at fixed intervals of length T_e. The aim is to learn a policy π to solve the following average-cost minimization problem:

$$\min_{\pi\in\Pi} \eta(\pi) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{\infty} \mathbb{E}_\pi[c(\mathbf{s}_t)] \tag{3}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expectation under policy π. The policy space Π denotes the set of all valid policies. We restrict Π to only include stationary Markovian polices. This means each $\pi \in \Pi$ makes decisions solely based on the current state, $\mathbf{s}_t$, and is independent of the time step t.

Any stationary Markov policy π induces a Markov chain over the states with a state-transition distribution $P_\pi(\mathbf{s}'|\mathbf{s})$. When the state Markov Chain is positive recurrent we have the equivalence $\eta(\pi) = \mathbb{E}_{\mathbf{s}\sim d(\pi)}[c(\mathbf{s})]$ where $d(\pi)$ is the steady-state distribution of the Markov chain induced by π. Note that for $\eta(\pi)$ to be finite, the state Markov chain must be positive recurrent. When $\eta(\pi)$ is finite, the following value functions are well defined:

$$V^\pi(\mathbf{s}) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} c(\mathbf{s}_t) - \eta(\pi)|\mathbf{s}_0 = \mathbf{s}\right] \tag{4}$$

$$Q^\pi(\mathbf{s},\mathbf{a}) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} c(\mathbf{s}_t) - \eta(\pi)|\mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right] \tag{5}$$

$$A^\pi(\mathbf{s},\mathbf{a}) = Q^\pi(\mathbf{s},\mathbf{a}) - V^\pi(\mathbf{s}) \tag{6}$$

2.3 Lyapunov Stability

Each queue backlog $q_{i,k}(t)$ must remain finite over any trajectory $\tau \sim \pi$, in order for the limit in equation (3) to be finite. This requirement is strongly related to the following notion of stability:

Definition 2.1 (Strong Stability [10]). *A discrete time process $\{q_t\}$ is strongly stable under transition function P if for any initial state q_0 the following condition is satisfied:*

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_P[|q_t|] < \infty \tag{7}$$

If the queue state Markov chain $\{\mathbf{q}_t\}$ under P_π is strongly stable for each queue in the network, the corresponding policy π is called a strongly stable policy. Strong stability is an important property for queuing networks that ensures the state Markov chain $\{\mathbf{q}_t\}$ is positive recurrent with unique steady-state distribution $d(\pi)$ that is independent of the initial state. Additionally, strong-stability implies that the number of packets in each buffer remains finite which is essential in the ODRLC setting as it ensures finite packet delay throughout the learning process.

Lyapunov Optimization is a technique for ensuring stability of dynamical systems through the use of Lyapunov functions. A Lyapunov function $\Phi : \mathcal{S} \mapsto \mathbb{R}^+$ maps state vectors to non-negative scalars which quantify the "energy" of each state. Specifically, for SQNs, $\Phi(\mathbf{s}_t)$ is typically defined to grow large as the queue sizes grow large. Stability is achieved by taking actions that cause the Lyapunov drift defined as $\mathbb{E}_\pi[\Delta(\mathbf{s}_t)] = \underset{\mathbf{s}_{t+1}\sim P_\pi(\cdot|\mathbf{s})}{\mathbb{E}}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|\mathbf{s}_t]$ to be negative when queue sizes grow too large. For the SQN models considered in this work, the Lyapunov function is solely a function of the queue state $\Phi(\mathbf{s}_t) = \Phi(\mathbf{q}_t)$ as the link-states are not influenced by control decisions. The following Lyapunov drift condition can be used to guarantee stability properties of classical network control algorithms:

Theorem 1. *The policy π is strongly stable if there exists a Lyapunov function $\Phi : \mathcal{S} \mapsto [0,\infty]$, a finite region of the state space $\mathcal{S}_1 \subset \mathcal{S}$ and a finite constant B such that:*

$$\underset{\mathbf{s}_{t+1}\sim P_\pi(\cdot|\mathbf{s}_t)}{\mathbb{E}}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|\mathbf{s}_t] \le -(1+\bar{q}_t) + B\mathbb{1}_{\mathcal{S}_1}(\mathbf{s}_t)\ \forall \mathbf{s}_t \in \mathcal{S}$$

where,

$$\mathbb{1}_{\mathcal{S}_1}(\mathbf{s}_t) = \begin{cases} 1, & \mathbf{s}_t \in \mathcal{S}_1, \\ 0, & \textit{otherwise} \end{cases}$$

This theorem is a modified version of the (V3) Lyapunov drift condition with $f = 1 + \bar{q}_t$ [9, Ch. 14]. Examples of strongly stable policies include the MaxWeight scheduling and Backpressure routing policies. [9, 20]. MaxWeight assigns weights $(q_{i,t} \times y_{i,t})$ to each node-link pair, and activates the link with the largest weight in each time step. Backpressure dynamically routes traffic based on congestion gradients without prespecified paths in Multihop networks. Both aim to to minimize bounds on the expected Lyapunov drift at each time step. While the throughput benefits of these algorithms are well established, they may suffer from poor delay performance as shown Section 5.

2.4 Policy Gradient Methods

In this work we focus on policy gradient methods, a class of DRL algorithms designed to directly optimize an agent's policy. We assume a NN is used to represent the policy for a particular task. We refer to these policy-NNs as actor networks. We denote a policy as π_θ, where θ represents the weights of the actor network. For parametric policies, the minimum cost objective can be expressed as $\min_{\theta\in\Theta} \eta(\pi_\theta)$. This minimization is over the possible policy parameters Θ, where Θ is determined by the actor network's architecture. Policy gradient methods perform this minimization iteratively: first estimating the gradient of $\eta(\pi_\theta)$ with respect to the actor network's parameters, θ, and then performing gradient descent. The analytical form of the gradient $\nabla_\theta\eta(\pi_\theta)$ is provided by the classical policy gradient theorem [19]:

$$\nabla_\theta\eta(\pi_\theta) = \mathbb{E}_{\substack{\mathbf{s}\sim d(\pi_\theta)\\ \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}}\left[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\nabla_\theta \log\pi_\theta(\mathbf{a}|\mathbf{s})\right] \tag{8}$$

where $d(\pi_\theta)$ is the stationary-distribution of the Markov chain induced by the policy π_θ. Implementations that utilize automatic differentiation software work by constructing a loss function whose gradient approximates the analytical gradient:

$$L_{PG}(\pi_\theta) = \frac{1}{T}\sum_{t=0}^{T-1} \hat{A}^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)\log\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \tag{9}$$

where $\hat{A}^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)$ is an estimate of the advantage function with respect to the policy π_θ. Utilizing the advantage function estimate is often preferred to an estimate of the state-action value $\hat{Q}^{\pi_\theta}(\mathbf{s}_t,\mathbf{a}_t)$ as it offers a lower-variance estimate of the gradient [4]. A significant limitation of policy gradient algorithms is their one-time use of each state-action pair $(\mathbf{s}_t,\mathbf{a}_t) \in \tau$. Re-using trajectories for multiple gradient updates often leads to destructively large policy updates which cause "performance collapse" [16].

Trust region methods, a subset of policy gradient methods, are designed to address the challenge of making the largest possible steps towards performance improvements upon each update to the policy without risking performance collapse. The theoretical foundation for these methods are bounds on the relative performance $\delta(\pi'_\theta,\pi_\theta) = \eta(\pi'_\theta) - \eta(\pi_\theta)$ between two policies. These bounds take the following form:

$$\delta(\pi'_\theta,\pi_\theta) \le \mathbb{E}_{\substack{\mathbf{s}\sim d(\pi_\theta)\\ \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}}\left[\frac{\pi'_\theta(\mathbf{a}|\mathbf{s})}{\pi_\theta(\mathbf{a}|\mathbf{s})}A^{\pi_\theta}(\mathbf{s},\mathbf{a})\right] + D(\pi'_\theta,\pi_\theta) \tag{10}$$

where $D(\pi'_\theta,\pi_\theta)$ is proportional to a measure of dissimilarity between the policies π'_θ and π_θ. Practical algorithms define surrogate objectives that such that minimizing the surrogate objective corresponds to minimizing this upper bound. These surrogate objectives are typically defined as to minimize the first term while constraining the KL-divergence between π'_θ and π_θ. However, like policy-gradient methods, this optimization is performed via gradient updates on the policy parameters θ, and thus effective algorithms must determine a suitable step-size to ensure $\delta(\pi'_\theta, \pi_\theta) < 0$. Trust-region Policy Optimization (TRPO) approaches this by framing the problem as a constrained minimization, solved approximately via conjugate gradient methods [14]. Proximal Policy Optimization (PPO), conversely, employs a clipped surrogate objective to deter substantial policy shifts, optimizing the bound while limiting large changes between successive policies [16].

3 INTERVENTION-ASSISTED POLICY FRAMEWORK

This section introduces the intervention-assisted policy framework for online training of SQN control agents. This framework addresses two critical questions: (1) how to design a policy that guarantees strong stability in the ODRLC setting? (2) how to update this policy based on the online interactions with the SQN environment? The proofs for each theorem in this section are omitted for brevity but can be found in the technical report [24]. The following assumption is required for this framework:

Assumption 1. *The agent has access to a known strongly stable policy* π_0

This assumption is not restrictive for SQN controls. Classical SQN control algorithms such as the MaxWeight or Backpressure policies can serve as π_0 for single-hop and multi-hop problems respectively [9]. Its crucial to recognize that strong stability does not imply optimality, where the optimal policy is defined as $\pi^* = \text{argmin}_\pi(\eta(\pi))$. We restrict our attention to SQN environments that may be stabilized. In which case, the optimal policy π^* is strongly stable.

3.1 Intervention Assisted Policy

The intervention-assisted framework is based on the partitioning the state-space $\mathcal{S}$ into two disjoint regions: a bounded "learning region" $\mathcal{S}_\theta$ and and unbounded "intervention region" $\mathcal{S}_0$. When the current state $\mathbf{s}_t$ falls within the learning region $\mathcal{S}_\theta$, the agent samples an action $\mathbf{a}_t$ from the actor policy π_θ. Conversely, if $\mathbf{s}_t \in \mathcal{S}_0$, the agent samples an action $\mathbf{a}_t$ from the known strongly stable policy π_0. Section 4.2 details a practical method of choosing this partitioning to ensure sample efficient learning. The intervention-assisted policy π_I is formulated as follows:

$$\pi_I(\cdot|\mathbf{s}) = I(\mathbf{s})\pi_0(\cdot|\mathbf{s}) + (1 - I(\mathbf{s}))\pi_\theta(\cdot|\mathbf{s}) \tag{11}$$

where

$$I(\mathbf{s}) = \begin{cases} 1, & \mathbf{s} \in \mathcal{S}_0, \\ 0, & \mathbf{s} \in \mathcal{S}_\theta \end{cases} \tag{12}$$

indicating that it utilizes policy $\pi_0(\cdot|\mathbf{s})$ when the state belongs to the set $\mathcal{S}_0$, and $\pi_\theta(\cdot|\mathbf{s})$ for states in $\mathcal{S}_\theta$. We leverage on-policy policy gradient methods where the intervention policy π_I is used by the agent to generate a trajectory $\tau \sim \pi_I$. Each trajectory is a sequence of states, intervention indicators, actions, and costs $\tau = (\mathbf{s}_0, I_0, \mathbf{a}_0, c_0, \mathbf{s}_1, ...)$ where $I_t \in \{0, 1\}$ indicates if an intervention occurred at time-step t.

3.1.1 Guaranteeing Stability of Intervention-Assisted Policies. This section details how the intervention-assisted policy π_I ensures strong stability using a Lyapunov optimization framework. Strong stability is vital from an SQN control perspective as it ensure packet delay is finite which is necessary for any policy deployed on a real-network. From a learning perspective, strong-stability guarantees a steady-state distribution $d(\pi_I)$, which is is necessary for well-defined policy gradient updates. The proofs of stability for intervention-assisted policies rely on the following lemma:

Lemma 1. *Under the assumption that all arrivals are finite, if the Lyapunov function $\Phi(\cdot)$ is bounded for each $\mathbf{s} \in \mathcal{S}$, then there exists a constant $B_i > 0$ for any bounded subset $\mathcal{S}_i \subset \mathcal{S}$, such that*

$$\mathbb{E}_\pi[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|\mathbf{s}_t \in \mathcal{S}_i] < B_i \tag{13}$$

for any policy π.

Lemma 1 means that given any finite region $\mathcal{S}_i \subset \mathcal{S}$, the maximum conditional drift is bounded above. Intuitively this is true because the max conditional drift is achieved by idling for any $\mathbf{s}_t$, and since arrivals are bounded, the max conditional drift is bounded.

The following theorem provides details how to ensure an intervention-assisted policy π_I is strongly stable.

Theorem 2. *Let $\mathcal{S}_\theta$ denote the learning region and $\mathcal{S}_0 = \mathcal{S} \setminus \mathcal{S}_\theta$ denote the intervention region for an intervention assisted policy π_I. If $\mathcal{S}_\theta$ is finite and π_0 satisfies Theorem 1 for some $\Phi(\cdot)$, B, and $\mathcal{S}_1$, then the following Lyapunov drift condition is satisfied:*

$$\mathbb{E}_{P_{\pi_I}}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|\mathbf{s}_t] \leq -(1+\bar{q}_t) + B_{1,\theta}\mathbf{1}_{\mathcal{S}_{1,\theta}}(\mathbf{s}_t) \ \forall \mathbf{s}_t \in \mathcal{S} \tag{14}$$

for a constant $B_{1,\theta} < \infty$ and the region $\mathcal{S}_{1,\theta} = \mathcal{S}_1 \cup \mathcal{S}_\theta$ where $\cup$ denotes the union between two sets.

Corollary 1. *If the conditions of Theorem 2 are satisfied, the intervention-assisted policy π_I is strongly stable.*

Thus to achieve strong-stability given a strongly stable π_0, we only need to ensure that the learning region $\mathcal{S}_\theta$ is finite. These conditions also mitigate the extrapolation burden on the actor network π_θ. Since the possible state inputs into π_θ is confined to the finite region $\mathcal{S}_\theta$.

3.2 Intervention Assisted Policy Gradients

This section extends the classical policy gradient theorem [19] to derive the analytical form of the gradient of the intervention-assisted performance objective $\nabla_\theta \eta(\pi_I)$.

Theorem 3. *Given a strongly stable intervention-assisted policy $\pi_I(\cdot|\mathbf{s}) = I(\mathbf{s})\pi_0(\cdot|\mathbf{s}) + (1 - I(\mathbf{s}))\pi_\theta(\cdot|\mathbf{s})$, and average-cost objective $\eta(\pi_I)$, the policy gradient is:*

$$\nabla_\theta \eta(\pi_I) = \mathop{\mathbb{E}}_{\substack{\mathbf{s}\sim d(\pi_I) \\ \mathbf{a}\sim\pi_I(\cdot|\mathbf{s})}}[(1 - I(\mathbf{s})Q^{\pi_I}(\mathbf{s}, \mathbf{a})\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s})] \tag{15}$$

where $d(\pi_I)$ is the steady-state distribution induced by π_I, and Q^{π_I} is the state-action value function with respect to policy π_I.

Equation (15) bears a strong resemblance to the original policy gradient theorem given in equation (9) albeit with a few key distinctions. First, the expectation for the intervention-assisted policy gradient is with respect to the steady-state and action distributions induced by the intervention-assisted policy π_I. Additionally, the intervention-assisted policy gradient depends on the state-action value function $Q^{\pi_I}(\mathbf{s}, \mathbf{a})$ which captures the state-action values with respect to the entire intervention-assisted policy instead of just π_θ . Like equation (8), the intervention-assisted policy gradient depends on $\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s})$, but note that the $(1 - I(\mathbf{s}))$ term blocks direct contributions to the overall gradient from any states where an intervention occurred. The overall performance of π_I, including the contributions from π_0 during interventions, still effects the gradient through the state-action value function $Q^{\pi_I}(\mathbf{s}, \mathbf{a})$ and the dependence on the steady-state distribution $d(\pi_I)$. Like in the non-intervention assisted case, equation (15) only theoretically supports a single update-step per trajectory generated.

3.3 Intervention-Assisted Policy Improvement Bounds

This section establishes bounds of the form equation (10) for intervention-assisted policies. These bounds allow us to extend the trust-region methods for intervention-assisted policies.

Theorem 4. *Consider two different strongly stable intervention assisted policies π_I' and π_I that utilize the same learning region $\mathcal{S}_\theta$ and intervention policy π_0 and only differ in their actor policies π_θ' and π_θ respectively. The performance difference is bounded as:*

$$\delta(\pi_I', \pi_I) \leq \mathop{\mathbb{E}}_{\substack{\mathbf{s}\sim d(\pi_I) \\ \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}} \left[A^{\pi_I}(\mathbf{s}, \mathbf{a}) \mathcal{R}_\theta^{\theta'}(\mathbf{a}|\mathbf{s}) \mid \mathbf{s} \in \mathcal{S}_\theta\right] + \mathop{\mathbb{E}}_{\substack{\mathbf{s}\sim d(\pi_I) \\ \mathbf{a}\sim\pi_0(\cdot|\mathbf{s})}} \left[A^{\pi_I}(\mathbf{s}, \mathbf{a}) \mid \mathbf{s} \in \mathcal{S}_0\right] + \mathcal{D}(d(\pi_I'), d(\pi_I)) \tag{16}$$

where $R_\theta^{\theta'}(\mathbf{a}|\mathbf{s}) = \frac{\pi_{\theta'}(\mathbf{a}|\mathbf{s})}{\pi_\theta(\mathbf{a}|\mathbf{s})}$ and:

$$\mathcal{D}(d(\pi_I'), d(\pi_I)) = 2 \sup_{\mathbf{s}\in\mathcal{S}} \left| \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi_I'(\cdot|\mathbf{s})} [A^{\pi_I}(\mathbf{s}, \mathbf{a})] \right| D_{\text{TV}}\left(d(\pi_I') \| d(\pi_I)\right)$$

Equation (16) resembles analogous bounds for non-intervention-assisted methods shown in equation (10), but with some key distinctions. The first difference is in conditioning. In equation (16), the first term is only considered when the state falls within the learning region $\mathcal{S}_\theta$. Similar to equation (15), it depends on the intervention assisted policy through the advantage function $A^{\pi_I}(\mathbf{s}, \mathbf{a})$ and on the actor policies through $R_\theta^{\theta'}(\mathbf{a}|\mathbf{s})$. To minimize this term, the ratio $R_\theta^{\theta'}(\mathbf{a}|\mathbf{s})$ should be maximized (minimized) when $A^{\pi_I}(\mathbf{s}, \mathbf{a})$ is negative (positive). The next major difference is that equation (10) lacks the second term present in equation (16). This term accounts for the performance of the intervention assisted policy π_I in the region $\mathcal{S}_0$. This term goes to zero as the actor policy π_θ learns to keep the state within $\mathcal{S}_\theta$. The last term in equation (16) is a measure of dissimilar between the steady-state distributions induced by π_I' and π_I. This term is strictly positive, meaning to minimize the bound, the difference between policies should be minimized.

4 ALGORITHMS

Building on the theoretical foundations of the previous sections, this section presents two practical algorithms for online training of intervention-assisted policies. These algorithms follow the same structure of on-policy actor-critic reinforcement learning algorithms [18]. Both algorithms follow a two-phase approach, consisting of a policy rollout phase and policy update phase, repeated across multiple training episodes $e = 1, 2, ..., E$:

(1) **Policy Rollout Phase**: The current policy $\pi_I^{(e)}$ interacts with the environment and generates a trajectory $\tau^{(e)}$.
(2) **Policy Update Phase** The trajectory $\tau^{(e)}$ is used in computing gradient updates. The trajectory is re-used in U update epochs to provide a sequence of updated policies

$$(\pi_I^{(e,0)}, \pi_I^{(e,1)}, ...\pi_I^{(e,U)})$$

where $\pi_I^{(e,u)}$ for $u > 0$ refers to the intervention assisted policy after the uth update epoch.

Here $\pi_I^{(e,0)}$ corresponds to the original policy $\pi_I^{(e)}$ that generated the trajectory $\tau^{(e)}$ in policy rollout phase. After all U updates, the most recently updated policy $\pi_I^{(e,U)}$ becomes starting policy for the next episode $(e + 1)$.

Each of the following algorithms differ only in their loss functions. For each algorithm, a trajectory $\tau^{(e)}$ is generated during the rollout phase of episode e, and the policy parameters are updated U times via stochastic gradient descent:

$$\theta^{(e,u+1)} = \theta^{(e,u)} - \alpha \nabla_\theta \mathcal{L}_{pol}(\pi_I^{(e,u)}, \tau^{(e)}) \tag{17}$$

where α is the learning rate and $L_{pol}(\pi_I^{(e,u)}, \tau^{(e)})$ is the policy loss function which is a function of the current policy $\pi_I^{(e,u)}$ and the trajectory $\tau^{(e)}$.

The first algorithm, the Intervention-Assisted Policy Gradient (IA-PG) algorithm, is an extension of the Vanilla Policy Gradient (VPG) algorithm[1] to the intervention-assisted setting. The IA-PG algorithm utilizes the following loss function:

$$\mathcal{L}_{PG}\left(\pi_I^{(e,u)}, \tau^{(e)}\right) = \frac{1}{T} \sum_{t=0}^{T-1} (1 - I_t) \hat{A}_t^{\pi_I^{(e)}} \log\pi_\theta^{(e,u)}(\mathbf{a}_t|\mathbf{s}_t) \tag{18}$$

where $A_t^{\pi_I^{(e)}} = A^{\pi_I^{(e)}}(\mathbf{s}_t, \mathbf{a}_t)$. Similar to the non-intervention assisted case, this loss function is designed to have a gradient that approximates the analytical gradient presented in equation (15). However, to reduce variance, IA-PG employs the advantage function estimate $\hat{A}^{\pi_I^{(e)}}$ instead of the state-action value estimate $\hat{Q}^{\pi_I^{(e)}}$. It's important to note that this loss function does not incorporate any constraints on the policy updates. This lack of restriction can potentially lead to performance degradation issues.

Our next algorithm, the Intervention-Assisted PPO (IA-PPO) algorithm, is designed to prevent such performance degradation. The IA-PPO algorithm builds upon the bound presented in theorem 4 to derive a loss function that resembles the original PPO algorithm [16]. Recall that iteratively minimizing the the right-hand side of

[1] Vanilla Policy Gradient — Spinning Up documentation (openai.com)

equation (16) leads to a monotonically improving sequence of policies with respect to the average cost objective. Since only the actor network parameters θ change between updates, the bound in equation (16) suggests solving the following optimization problem for updates:

$$\theta^{(e,u+1)} = \underset{\theta}{\text{argmin}} \quad \mathbb{E}\left[((1 - I(\mathbf{s}))A^{\pi_I^{(e)}}(\mathbf{s},\mathbf{a})\mathcal{R}_{(e)}^{(e,u)}(\mathbf{a}|\mathbf{s})\right] + \mathcal{D}(d(\pi_I^{(e,u)}), d(\pi_I^{(e)})) \tag{19}$$

where the expectation is over $\mathbf{s} \sim d(\pi_I^{(e)})$ and $\mathbf{a} \sim \pi_I^{(e)}(\cdot|\mathbf{s})$, and $\mathcal{R}_{(e)}^{(e,u)}(\mathbf{a}|\mathbf{s}) = \frac{\pi_\theta^{(e,u)}(\mathbf{a}|\mathbf{s})}{\pi_\theta^{(e)}(\mathbf{a}|\mathbf{s})}$.

Notice, equation (19) omits the second term in equation (16) in as it does not depend on the variable θ. Minimizing the first term of equation (19) encourages maximizing the ratio $\mathcal{R}_{(e)}^{(e,u)}(\mathbf{a}|\mathbf{s})$ for negative advantages, and encourages minimizing $\mathcal{R}_{(e)}^{(e,u)}(\mathbf{a}|\mathbf{s})$ for positive advantages. However, the second term penalizes large deviations between policies. As π_0 and $I(\mathbf{s})$ remain unchanged between updates, the primary factor influencing this term is the difference between the actor policies between updates. To this end, the IA-PPO algorithm uses the following clipped loss:

$$\mathcal{L}_{clip}(\pi_I^{(e,u)}, \tau^{(e)}) = \frac{1}{T}\sum_{t=0}^{T-1}(1 - I(\mathbf{s}_t)) \max\{A_t^{\pi_I^{(e)}} R_{(e)}^{(e,u)}(\mathbf{a}_t|\mathbf{s}_t), \text{clip}(\epsilon, \hat{A}_t^{\pi_I^{(e)}}) \tag{20}$$

where $\epsilon \in (0, 1)$ is a hyperparameter and

$$\text{clip}(\epsilon, A) = \begin{cases} (1+\epsilon)A, & A \geq 0 \\ (1-\epsilon)A, & A < 0 \end{cases}$$

This clipped loss function creates more conservative updates by attempting to limit the divergence of policies between updates while still increasing (decreasing) the likelihood of actions that decrease (increase) the advantage. This focus on conservative updates is even more critical in online training compared to simulation-based training. Online training relies on a single sample path generated from the previous trajectory's end state. This limitation leads to inherently noisier and potentially more biased advantage function estimates compared to settings where multiple trajectories are generated from various starting states (simulation-based training). The clipped loss function helps to mitigate the impact of this noise and bias on policy updates.

4.1 Pseudocode

Algorithm 1 provides an outline of the IA-PG and IA-PPO algorithms as actor-critic style algorithms. The only difference between the two algorithm is the computation of $\mathcal{L}_{pol}$ in line 12, as the IA-PG algorithm uses equation (18) while the IA-PPO uses equation (20) for the policy loss. The algorithm as written assumes that $\mathcal{S}_\theta$ and $\mathcal{S}_0$ have been pre-determined. The next section details how these regions were selected for the results shown in Section 5. In the update phase, the algorithm operates without requiring knowledge of the underlying MDP, such as the transition and cost functions. Instead, it relies solely on the collected transitions from the previous trajectory τ, making it a model-free approach. The advantage function is estimated using an average cost variant of the Generalized Advantage Estimation (GAE) algorithm [15]. This GAE algorithm utilizes a separate NN, a "critic" network, to for value estimation. The details of critic network and advantage function estimator are given in [24].

Algorithm 1 Intervention-Assisted PG/PPO Algorithm

```
1:  for each epoch $e = 1, E$ do
2:      # Policy Rollout Phase
3:      Initialize an empty trajectory buffer $\tau$
4:      for each step $t = 0, 1..., T_e - 1$ do
5:          Observe state $\mathbf{s}_t$ and compute $I_t = \mathbf{1}(\mathbf{s}_t \in \mathcal{S}_0)$
6:          Sample action $\mathbf{a}_t \sim \pi_I(\cdot|\mathbf{s}_t)$
7:          Execute action $\mathbf{a}_t$, observe cost $c_t$ and next state $\mathbf{s}_{t+1}$
8:          Store transition $(\mathbf{s}_t, I_t, , \mathbf{a}_t, c_t, \mathbf{s}_{t+1})$ in $\tau$
9:      # Update Phase
10:     Estimate advantages $\hat{A}^{\pi_I^{(e)}}(s_t, a_t)\ \forall\ (a_t, s_t) \in \tau$
11:     for each update epoch $u = 1, U$ do
12:         Compute policy loss $\mathcal{L}_{pol}$
13:         Compute value loss $\mathcal{L}_{val}$
14:         Update policy parameters: $\theta \longleftarrow \theta - \alpha\nabla_\theta L_{pol}$
15:         Update critic parameters function: $\phi \longleftarrow \phi - \alpha\nabla_\phi L_{val}$
```

4.2 Learning Region Selection

To achieve sample efficient learning, the finite learning region $\mathcal{S}_\theta$ should be specified to minimize the amount of interventions. To this end, we can leverage Theorem 1 to ensure that interventions not only stabilize the network, but also push the network state back towards the non-intervention region $\mathcal{S}_\theta$ in expectation. Given a strongly stable intervention policy π_0, according to Theorem 1, there exists a bounded sub-region $\mathcal{S}_1 \in \mathcal{S}$ such that all states $\mathbf{s} \notin \mathcal{S}_1$ have negative expected drift, or more specifically:

$$\mathbb{E}_{\pi_0}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t) \mid \mathbf{s}_t \notin \mathcal{S}_1] \leq -(\bar{q}_t + 1) \tag{21}$$

Setting $\mathcal{S}_\theta = \mathcal{S}_1$ would ensure that each intervention results in negative expected drift, effectively pushing the state Markov chain back to $\mathcal{S}_\theta$ once it leaves. If $\mathcal{S}_1$ is not known beforehand, it can be estimated by producing a trajectory using only π_0. In practice, it may be very difficult to estimate $\mathcal{S}_1$ exactly as it requires learning the relationship between high-dimensional state-space and the expected drift. To address this challenge, we aim to learn a superset $\mathcal{S}_g \supseteq \mathcal{S}_1$ where $\mathcal{S}_g$ can be estimated using a lower-dimensional representation of the states. To this end, we use the following corollary:

Corollary 2. *Given an strongly stable policy π and a convex function $g : \mathcal{S} \mapsto [0, \infty)$, we can bound the expected drift conditioned on $g(\mathbf{s}_t)\ \forall\ \mathbf{s}_t$ as:*

$$\mathbb{E}_{P_\pi}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|g(\mathbf{s}_t)] \leq -(1 + \bar{q}_t) + B_g\mathbf{1}_{\mathcal{S}_g}(\mathbf{s}_t) \tag{22}$$

where $\mathcal{S}_g = \{\mathbf{s}' \in \mathcal{S} : g(\mathbf{s}') \leq \max_{\mathbf{s}\in\mathcal{S}_1} g(\mathbf{s})\}$ and B_g is a constant.

This corollary ensures that if $\mathcal{S}_g$ is known, the expected drift for $\mathbf{s} \notin \mathcal{S}_g$ is negative. Letting $g(\mathbf{s}_t) = \bar{q}_t$ means $\mathcal{S}_g$ is defined based off the network backlog and we only need to estimate a $\bar{q}^*$ such that:

$$\mathbb{E}_{P_\pi}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|\bar{q}_t, \bar{q}_t > \bar{q}^*] \leq -(1 + \bar{q}_t) \tag{23}$$

This quantity $\bar{q}_t^*$ is easier much easier to estimate compared to the exact region $\mathcal{S}_1$. Once $\bar{q}_t^*$ is estimated, the intervention criteria can be defined as:

$$I(\mathbf{s}_t) = \begin{cases} 0, & \bar{q}_t \leq \bar{q}^* \\ 1, & \bar{q}_t > \bar{q}^* \end{cases} \tag{24}$$

Under this criteria, $\mathcal{S}_\theta$ remains bounded thus the intervention assisted policy is strongly stable given that π_0 is strongly stable, and the expected drift given $I(\mathbf{s}_t) = 1$ is negative. Note that $\bar{q}_t$ only contains partial information about the high-dimensional state $\mathbf{s}_t = (\mathbf{q}_t, \mathbf{y}_t)$ as it neglects all the information on the link states $\mathbf{y}_t$ in addition to averaging the information over the queue state $\mathbf{q}_t$. As a result, $\mathcal{S}_\theta$ isn't minimal in the sense that it can contains some states such that $\mathbb{E}_{P_{\pi_0}}[\Phi(\mathbf{s}_{t_1}) - \Phi(\mathbf{s}_t)|\mathbf{s}_t] < 0$, but in practice we have found this backlog based intervention criteria a good strategy for sample efficient learning as long as we use a pessimistic estimate of $\bar{q}^*$. To form this pessimistic estimate, we use π_0 to collect a trajectory τ_0, and estimate the backlog at which the drift is negative from this trajecory i.e. $\hat{q}^* = \min\{\bar{q} : \Delta_{\pi_0}(\bar{q}) < \omega\}$ where the hyperparameter $\omega < 0$ is a constant and $\Delta_{\pi_0}(\bar{q}_t) = \mathbb{E}_{\pi_0}[\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)|\bar{q}_t]$ is the expected drift given the current backlog $\bar{q}_t$. For the experiments in section 5, we collect τ_0 by only using π_0 until the state-Markov chain converges. The full details of this procedure can be found in [24].

5 EXPERIMENTS

We conducted a series of experiments to evaluate the IA-PG and IA-PPO algorithms. The following SQN environments were used in the experiments:

(1) **SH1**: A two user ($K = 2$) single-hop wireless network.
(2) **SH2**: A four user ($K = 4$) single-hop wireless network. The topology is shown in Figure 1.
(3) **MH1**: A multihop environment with two classes ($K = 2$), six links ($M = 6$), and four nodes ($N = 4$). The topology is shown in Figure 2.
(4) **MH2**: A multihop environment with four classes ($K = 4$), thirteen links ($M = 13$), and eight nodes ($N = 8$). The topology can be found in Figure 4 of [24].

The arrival and service distributions for each SQN environment can be found in the technical report [24].

We evaluate the performance of all learning algorithms against the MaxWeight algorithm for single-hop network instances and Backpressure for the multi-hop network instances. In addition to these classic network control algorithms, we evaluate the performance of the following DRL algorithms developed for average-reward tasks:

(1) **Average Cost PPO (AC-PPO)**[8]: an average-cost variant of the original PPO algorithm that does not leverage interventions.
(2) **Stability then Optimality PPO (STOP-PPO)** [11]: an average reward policy gradient algorithm designed for environments with unbounded state-spaces. STOP-PPO utilizes reward shaping to first train the agent to learn how to stabilize the queuing network before learning how to optimize the queuing network. Our variant differs from the original as it includes the PPO clipping mechanism in the policy loss function and utilizes the Average Value Constraint method to control the bias of the critic network.

5.1 ODRLC Experiment Procedure

The following online-training process akin to an ODRLC setting for all algorithms (IA-PG, IA-PPO, AC-PPO, and STOP-PPO). The agent interacts continuously with the SQN environment from $t = 0$ until a long-time horizon T_{end}. The performance of the agent is monitored over the entire long trajectory. We measure the following two metrics: the time-averaged backlog $\bar{q}_t^{(t)} = \frac{1}{t}\sum_{h=0}^{t-1}\bar{q}_h$ and $T_{MA} = 10,000$ step moving average $q_t^{(MA)} = \frac{1}{T_{MA}}\sum_{h=t-T_{MA}}^{t-1}\bar{q}_t$. The moving average captured shorter-term performance, while the time-averaged metric assessed performance up to the current time step.

The experiment time horizon T_{end} was divided into distinct episodes of length T_e. For the IA-PG and IA-PPO algorithms, the first E_0 episodes only the intervention policy π_0 was used. These trajectories $(\tau^{(0)}, \tau^{(1)}, ...\tau^{(E_0)}$ were then used to estimate $\bar{q}_t^*$ and determine the learning $\mathcal{S}_\theta$ and intervention regions $\mathcal{S}_0$ according to the learning region estimation algorithm given in [24]. The performance of π_0 was measured and included in analysis of the intervention-assisted algorithm's performance. After episode E_0, the full intervention-assisted policy $\pi_I^{(e)}$ is used to generate all future trajectories. After each trajectory $\tau^{(e)} \sim \pi_I^{(e)}$ was generated, the actor network was updated U times using the corresponding policy loss function. The AC-PPO and STOP PPO algorithms the same training procedure, minus the initial learning region estimation phase meaning their actor policy π_θ generates all trajectories starting from $t = 0$. For all algorithms, the environment state is never reset. Additionally, for the IA-PG, IA-PPO, and AC-PPO algorithms, the cost shaping function $r'(\mathbf{s}_t) = \frac{-1}{1+\bar{q}_t}$ was used. This cost shaping function ensures that the scales of costs are similar for different environments even if the backlog of the the respective optimal policies differ substantially, which allowed us to use the same learning rate for all environments as the magnitude of the gradients were comparable. We also used the symmetric natural log state transformation for all DRL algorithms to decrease the magnitude of divergence between inputs to the actor and critic networks [11].

All experiments were repeated five times for each algorithm using the same random seeds. This ensured identical arrival processes and link states across corresponding algorithms in each environment. All algorithms employed the Average Value Constrained Critic, with advantages estimated using an average-cost variant of the Generalized Advantage Estimation (GAE) algorithm. For consistency, identical hyperparameters were used across all environments for each algorithm if they shared hyperparameters. A detailed description of hyperparameters and network architectures can be found in the technical report [24].

5.2 Results

5.2.1 Intervention-less DRL Baselines. We start by demonstrating how the intervention-less DRL algorithms struggle to stabilize the queues resulting in very poor performance on most environments. The online-performance of the AC-PPO and STOP-PPO algorithms are shown in Figure 3 which can be found on the last page. For the SH2, MH1, and MH2 environments, neither the AC-PPO nor the STOP-PPO algorithm can can stabilize the queuing network resulting in the networks queue backlog growing without bounds. For the SH1 environment, the AC-PPO algorithm was able to stabilize the queuing network for each seed while the STOP-PPO algorithm only stabilized the queuing network in three of the five seeds. The variation in performance between the SH1 network and the other network scenarios is best explained by examining the performance of a randomized policy on each network scenario. Only in the SH1 network scenario does the randomized policy stabilize the queuing network. The randomized policy performance is a good indicator of whether or not an intervention-less policy can work in the ODRLC setting as an untrained agent's initial policy is typically close to a randomized policy. If this initial randomized policy is stable and the policy updates are conservative enough, as enabled by PPO-style updates, then its possible for the agent to avoid the extrapolation loop. However, its evident that an intervention-less approach to ODRLC will fail on many SQN control tasks due to the unbounded extrapolation loop of ODRLC in unbounded state-spaces.

5.2.2 Intervention-Assisted Algorithms. Now that we have established the necessity of intervention-assisted methods for online-training of queueing network control algorithms we demonstrate that the IA-PG and IA-PPO algorithms can learn a better policy than classical network control algorithms online. These results are shown in Figure 4. Since the intervention-less DRL approaches failed in a majority of the environments, we focus on the comparison between the intervention-assisted algorithms and the classical network control algorithms as a baseline. In all environments, the time-averaged backlog of the intervention-assisted algorithms outperforms the non-learning baseline. It is also evident that the IA-PPO algorithm is more sample-efficient than the IA-PG algorithm. This is best seen by the average rate at which $\bar{q}_t^{(T_{MA})}$ drops below the time-averaged backlog of the non-learning baseline in each environment. It can also be seen that the moving average backlog $\bar{q}_t^{MA}$ of the IA-PPO algorithm is less noisy than that of IA-PG, especially for the SH2 and MH2 results. The SH2 and MH2 environments were also the more challenging environments as it took approximately 300,000 timesteps before $\bar{q}_t^{(MA)}$ of the IA-PPO algorithm was less than that of the MaxWeight/Backpressure policies, wheare is took closer to 100,000 timesteps to accomplish the same in the SH1 and MH1 environments. These environments had a higher-dimensional state-space compared to the SH1 and MH1 environments. Also, it can be inferred that the effective state-space in which the agent's encountered is much larger as seen by the maximum of $q_t^{(MA)}$ encountered over the experiments length.

6 CONCLUSION

In conclusion, this work introduces a novel intervention-assisted policy gradient approach for enabling Online Deep Reinforcement Learning Controls (ODRLC) in stochastic queuing networks. Our methods, IA-PG and IA-PPO, merge classical control's stability with neural networks' adaptability, showing superior queue stability and network optimization in real-time over traditional methods. Experiments confirm our framework's effectiveness, overcoming unbounded queue challenges and setting a theoretical groundwork for future DRL applications in complex systems. Future efforts will refine intervention mechanisms, explore scalability, and extend our framework to other domains with similar issues. This research paves the way for integrating traditional control and modern machine learning for advanced system optimization and control.

REFERENCES

[1] Yinlam Chow, Ofir Nachum, Aleksandra Faust, M. Ghavamzadeh, and Edgar A. Duéñez-Guzmán. 2019. Lyapunov-Based Safe Policy Optimization for Continuous Control. *ArXiv* (Jan. 2019).

[2] J. G. Dai and Mark Gluzman. 2022. Queueing Network Controls via Deep Reinforcement Learning. *Stochastic Systems* 12, 1 (March 2022), 30–67.

[3] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. 2020. Generalization and Regularization in DQN. arXiv:1810.00123

[4] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. 2004. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research* 5, Nov (2004), 1471–1530.

[5] P.J. Haley and D. Soloway. 1992. Extrapolation Limitations of Multilayer Feedforward Neural Networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, Vol. 4. 25–30 vol.4.

[6] John D. C. Little. 1961. A Proof for the Queuing Formula: L= λ W. *Operations Research* 9, 3 (1961), 383–387. jstor:167570

[7] Yongshuai Liu, Avishai Halev, and Xin Liu. 2021. Policy Learning with Constraints in Model-free Reinforcement Learning: A Survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Montreal, Canada, 4508–4515. https://doi.org/10.24963/ijcai.2021/614

[8] Xiaoteng Ma, Xiaohang Tang, Li Xia, Jun Yang, and Qianchuan Zhao. 2021. Average-Reward Reinforcement Learning with Trust Region Methods. *arXiv* (2021).

[9] Sean Meyn and Richard L. Tweedie. 2009. *Markov Chains and Stochastic Stability* (2 ed.). Cambridge University Press, Cambridge.

[10] Michael Neely. 2010. Stochastic Network Optimization with Application to Communication and Queueing Systems. *Synthesis Lectures on Communication Networks* 3, 1 (2010), 1–211.

[11] Brahma S. Pavse, Yudong Chen, Qiaomin Xie, and Josiah P. Hanna. 2023. Tackling Unbounded State Spaces in Continuing Task Reinforcement Learning. arXiv:2306.01896

[12] Majid Raeis, Ali Tizghadam, and Alberto Leon-Garcia. 2021. Queue-Learning: A Reinforcement Learning Approach for Providing Quality of Service. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 1 (May 2021), 461–468.

[13] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. 2021. Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access* 9 (2021), 153171–153187.

[14] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017. Trust Region Policy Optimization. arXiv:1502.05477

[15] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-Dimensional Continuous Control Using Generalized Advantage Estimation. arXiv:1506.02438

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347

[17] Adam Stooke, Joshua Achiam, and Pieter Abbeel. 2020. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. https://doi.org/10.48550/arXiv.2007.03964 arXiv:2007.03964 [cs, math]

[18] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second edition ed.). The MIT Press, Cambridge, Massachusetts.

[19] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, Vol. 12. MIT Press.

[20] L. Tassiulas and A. Ephremides. 1992. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Trans. Automat. Control* 37, 12 (1992), 1936–1948.

[21] Eugene Valassakis, Zihan Ding, and Edward Johns. 2020. Crossing the Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5372–5379.

[22] Nolan Wagener, Byron Boots, and Ching-An Cheng. 2021. Safe Reinforcement Learning Using Advantage-Based Intervention. arXiv:2106.09110

[23] Fan Wang, Bo Zhou, Ke Chen, Tingxiang Fan, Xi Zhang, Jiangyong Li, Hao Tian, and Jia Pan. 2018. Intervention Aided Reinforcement Learning for Safe and Practical Policy Optimization in Navigation. In *Proceedings of The 2nd Conference*

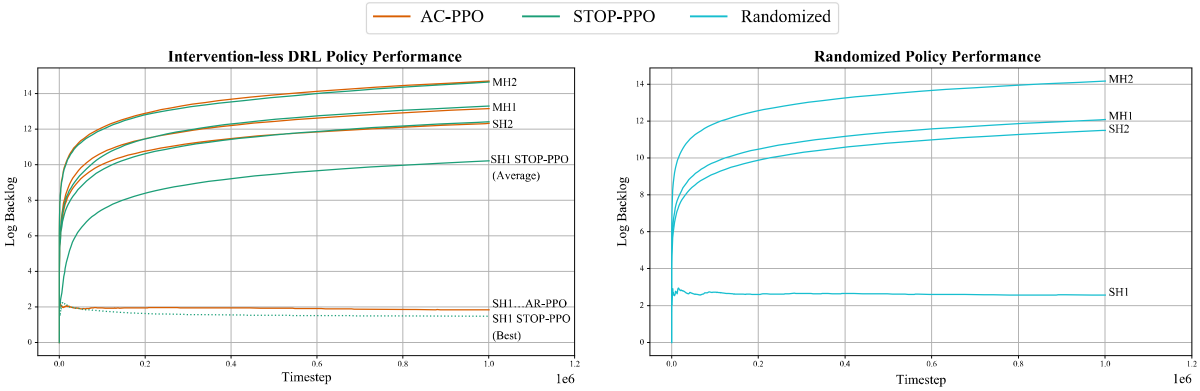

Figure 3: Performance of the AC-PPO, STOP-PPO, and randomized policy on each network scenario. The Y-axis represents the natural logarithm of the time-averaged backlog $\log(\bar{q}_t^{(t)})$. Each solid line represents an average over five seeds.

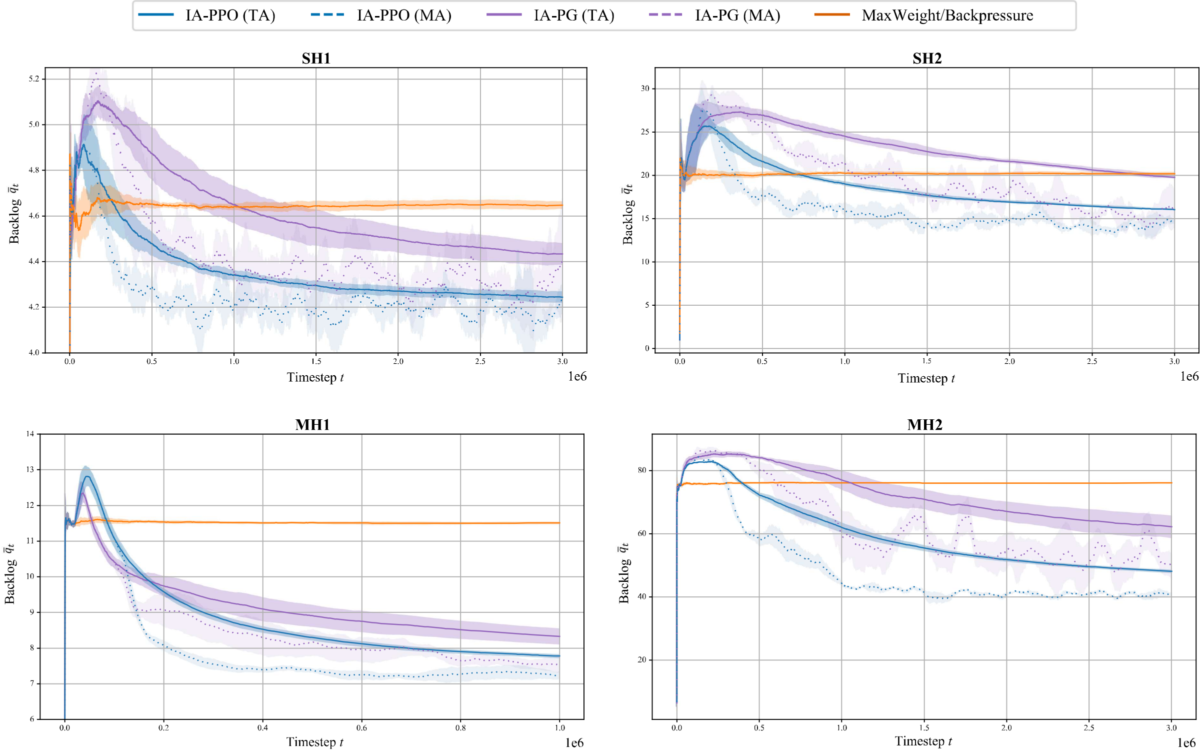

Figure 4: Performance of the IA-PG, IA-PPO, and MaxWeight/Backpressure algorithms on each environment. The Y-axis represents the actual queue backlog $\bar{q}_t$. Each line represents an average over five seeds. The solid lines correspond with the time-averaged backlog metrics $\bar{q}_t^{(t)}$ and the dashed lines correspond with a $T_{MA} = 10,000$ step moving average $\bar{q}_t^{(T_{MA})}$. The shaded regions correspond to the 95% confidence intervals for each performance metric.

on Robot Learning. PMLR, 410–421.

[24] Jerrod Wigmore, Brooke Shrader, and Eytan Modiano. 2024. Intervention-Assisted Policy Gradient Methods for Online Stochastic Queuing Network Optimization: Technical Report. arXiv:2404.04106

[25] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2021. How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks. arXiv:2009.11848

[26] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. 2020. Projection-Based Constrained Policy Optimization. https://doi.org/10.48550/arXiv.2010.03152 arXiv:2010.03152 [cs]

[27] Amy Zhang, Nicolas Ballas, and Joelle Pineau. 2018. A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning. arXiv:1806.07937

[28] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. 2018. A Study on Overfitting in Deep Reinforcement Learning. arXiv:1804.06893

[29] Yiming Zhang and Keith W. Ross. 2021. On-Policy Deep Reinforcement Learning for the Average-Reward Criterion. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 12535–12545.

accepted 28 August 2024