Learning to Stabilize Online Reinforcement Learning in Unbounded State Spaces

Brahma S. Pavse [1] **Matthew Zurek** [1] **Yudong Chen** [1] **Qiaomin Xie** [1] **Josiah P. Hanna** [1]

Abstract

In many reinforcement learning (RL) applications, we want policies that reach desired states and then keep the controlled system within an acceptable region around the desired states over an indefinite period of time. This latter objective is called *stability* and is especially important when the state space is unbounded, such that the states can be arbitrarily far from each other and the agent can drift far away from the desired states. For example, in stochastic queuing networks, where queues of waiting jobs can grow without bound, the desired state is all-zero queue lengths. Here, a stable policy ensures queue lengths are finite while an optimal policy minimizes queue lengths. Since an optimal policy is also stable, one would expect that RL algorithms would implicitly give us stable policies. However, in this work, we find that deep RL algorithms that directly minimize the distance to the desired state during online training often result in unstable policies, i.e., policies that drift far away from the desired state. We attribute this instability to poor credit-assignment for destabilizing actions. We then introduce an approach based on two ideas: 1) a Lyapunov-based cost-shaping technique and 2) state transformations to the unbounded state space. We conduct an empirical study on various queuing networks and traffic signal control problems and find that our approach performs competitively against strong baselines with knowledge of the transition dynamics. Our code is available here: `https://github.com/Badger-RL/STOP`.

1. Introduction

Much of the recent progress in reinforcement learning (RL) has focused on obtaining optimal performance on tasks that can be completed in some amount of time. However, there are many real-world sequential decision-making problems in which we want the learning agent to operate optimally over an indefinite period of time. For example, in traffic intersection management, we want a policy that reaches and stays at the desired state of minimum waiting time of cars over an indefinite period of time. Such problems are also common in control of stochastic queue networks and industrial process control (Srikant & Ying, 2014; Neely, 2010). While it is important that the agent is optimal in these problems, it is simultaneously important that the learning agent is *stable*, meaning that its decisions keep the system in an acceptable region of the state space near the desired state (Meyn, 2022).

Throughout this paper, we use queuing as a motivating example for stochastic environments with unbounded state spaces. In stochastic queuing networks, a server policy must select from several queues to serve based on the current number of jobs waiting in each queue and the desired state is all-zero queue lengths. Ideally, the server keeps the average length of queues short so that no job must wait a long time to be served. Jobs arrive continuously and — if the server makes poor decisions (e.g., serving an empty queue) — then the average length of the queues can grow infinitely-long. In this case, an optimal policy minimizes the average queue length over time, while a stable policy ensures the queue lengths remain finite. By implication, an optimal policy is also a stable policy.

Since optimal behavior implies stable behavior, we would expect that directly trying to be optimal would implicitly give us stable behavior. However, in this work, we find that agents that do so actually learn unstable policies on multiple real-world-inspired domains. In particular, we empirically show that online deep RL agents that minimize the optimality cost objective of the domain drift far away from the desired state. We then propose an approach that specifically encourages the agent to be stable and optimal. More concretely, our contributions are as follows:

1. We demonstrate that solely minimizing the optimality cost objective inadequately discourages destabilizing actions, which leads to the agent taking those actions early in learning. This challenge is especially problematic in the: 1) online learning setting, where destabiliz-

[1]University of Wisconsin–Madison, USA. Correspondence to: Brahma S. Pavse <pavse@wisc.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

ing actions must be discouraged as quickly as possible to prevent the system's state from destabilizing, and 2) unbounded state space setting where the neural network's inability to effectively generalize across arbitrarily far unbounded samples prevents accurate credit assignment (Chollet, 2017).

2. We introduce STability and OPtimality (STOP), an approach that encourages stability and optimality. STOP is based on the combination of two techniques: 1) a Lyapunov-inspired cost shaping approach that explicitly encourages the agent to be stable and optimal and 2) state transformations that compress the unbounded state space to mitigate the burden of extreme generalization on neural networks. The first technique discourages destabilizing actions more than the true optimality cost function, even without knowing which actions are destabilizing. We provide intuition on how to choose an appropriate Lyapunov function by drawing upon results from queueing theory. The second technique improves the agent's ability to generalize across the unbounded state-space.
3. We then prove that our cost-shaping approach has the desired consistency property that it does not change the optimal policy. This result is analogous to the potential-based shaping result of Ng et al. (1999), but our result applies to the average-reward, unbounded state space setting, and relates consistency to stability.
4. We conduct a thorough empirical study and analyze the role of different components of STOP on the challenging real-world-inspired domains of queuing and traffic intersection management. We show that STOP enables learning of highly performant online RL policies and, in some cases, outperforms algorithms that have knowledge of the transition dynamics.

2. Related Work

Online RL. Our work focuses on online (or continuing) RL tasks in which interaction never terminates and performance is measured online (Sutton & Barto, 2018). This setting has been described with the autonomous RL (Sharma et al., 2021) or single-life RL (Chen et al., 2022) formalisms. In many practical set-ups, it is infeasible to reset the agent to a new initial state. For example, manually placing a robot in an initial state or removing all vehicles at a traffic intersection is costly or even inappropriate. Recent work has thus considered *reset-free* RL where the agent learns to reset itself (Eysenbach et al., 2018; Zhu et al., 2020; Gupta et al., 2021; Han et al., 2015; Sharma et al., 2022). However, these works still require a manual reset (Eysenbach et al., 2018), use policies learned on one task as a reset policy in another task (Gupta et al., 2021), or require access to demonstrations from a target policy (Sharma et al., 2022). Our work performs no resets and evaluates performance on a single infinitely-long, stationary task.

In online RL, the natural performance measure of an agent is average-reward (Sutton & Barto, 2018; Naik et al., 2019). Our work is different from prior average-reward RL work (Mahadevan, 1996; Schwartz, 1993; Wan et al., 2021; Wei et al., 2020; Zhang & Ross, 2021; Zhang et al., 2021) in that we focus on the challenge of learning stable behavior in an unbounded state space.

Stability and Unbounded State Spaces in RL. Stability is related to the notion of *safety* in RL (Garcıa & Fernández, 2015), but with crucial differences. Safety is typically defined as hard constraints on individual states and/or actions (Hans et al., 2008; Dalal et al., 2018; Dean et al., 2019; Koller et al., 2018), or soft constraints on the expected costs over the trajectory (Achiam et al., 2017; Chow et al., 2018; Yu et al., 2019). In contrast, stability concerns the long-run asymptotic behavior of the system, which cannot be immediately written as constraints over the states and actions or a budget for some cost.

Some work considers control-theoretic notions of stability (Vinogradska et al., 2016; Berkenkamp et al., 2017). While related, these results mostly consider systems with deterministic and partially unknown dynamics. For example, Westenbroek et al. (2022) propose a cost-shaping approach with similarities to ours, but focusing on deterministic dynamics; they consider discounted costs, and do not study the continuing setting with unbounded states.

The work of Shah et al. (2020) considered stochastic stability for RL in the unbounded state space setting. However, their work assumed a tabular setting, relied on access to the model of the environment, and ignored optimality to focus exclusively on stability. Our work makes stability practical for deep RL without having access to the environment transition dynamics and optimizes for both stability and optimality. The few other works that consider stability (Dai & Gluzman, 2022; Liu et al., 2022) assume that a stable policy is given and use it as a starting point for learning an optimal policy. We make no such assumption and try to learn a stable policy directly from a random policy.

The combination of unbounded state space and the continuing setting is scarce in the RL literature. Existing works acknowledge the challenges from unboundedness, but they either artificially bound the state space (Liu et al., 2022; Wei et al., 2023) or assume a finite-horizon training setting (Dai & Gluzman, 2022).

3. Preliminaries

Average-Reward. Consider an infinite-horizon Markov decision process (MDP) (Puterman, 2014), $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, c, d_1 \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^d$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}: \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition dynamics, $c: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ the cost function, and $d_1 \in \Delta(\mathcal{S})$ the initial state distribution. Here we follow the control-theoretic convention and consider costs (the negation of rewards). Without loss of generality, we assume the cost is non-negative, and often restrict to cost-functions that are only dependent on the next state, s_{t+1}. Typically, and in the context of our work, the optimality cost function c denotes a notion of distance from a target state such as the L_1 distance and the target state is the all-zero vector state (for more details, refer to Appendix C.3) (Leon-Garcia, 2008; Ault & Sharon, 2021).

In the online task formulation, an agent, acting according to policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, generates a *single* infinitely long trajectory: $s_1, c_1, a_1, s_2, ...$, where $s_1 \sim d_1$, $a_t \sim \pi(\cdot|s_t), c_t = c(s_t, a_t, s_{t+1})$, and $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. Unlike episodic RL, there are no resets in this formulation. Accordingly, we consider the long-run average-cost objective (Naik et al., 2019; 2021):

$$J^{\mathrm{O}}(\pi) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi \left[c_t\right]. \tag{1}$$

An optimal policy is one that minimizes $J^{\mathrm{O}}(\pi)$. In the average-cost setting, the analog of the standard RL value functions are the *differential value functions* which are defined as follows: the *differential* action-value function, $Q^\pi(s,a) := \lim_{T\to\infty} \mathbb{E}_\pi[\sum_{t=1}^{T}(c_t - J^{\mathrm{O}}(\pi)) \mid s_1 = s, a_1 = a]$, the differential state-value function, $V^\pi(s) := \lim_{T\to\infty} \mathbb{E}_\pi[\sum_{t=1}^{T}(c_t - J^{\mathrm{O}}(\pi)) \mid s_1 = s]$, and the differential advantage function, $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. The advantage function quantifies how good or bad an action is in a given state in yielding the long-term outcome. Note that since we are *minimizing costs* rather than maximizing rewards, good and bad actions should have negative and positive advantage respectively. We make the standard assumption that the MDP is communicating (Bertsekas, 2015), meaning that for any pair of states s and s', there exists a policy that can transition from s to s' in a finite number of steps with non-zero probability. This assumption guarantees that the optimal average cost value $\inf_\pi J^{\mathrm{O}}(\pi)$ is independent of the starting state.

Unbounded State Spaces and Stability. In an unbounded state space, there is no limit to values that the features of a state can take on. This type of state space is different from the bounded state space formulation used in popular RL testbeds such as in MuJoCo (Todorov et al., 2012) where all states are in a continuous *bounded* region (e.g., robotic arm joint angles).

In unbounded state space stochastic control problems, *stochastic stability* is a fundamental concept (Shah et al., 2020).

Definition 3.1 (Stochastic Stability)**.** A policy π is stable if and only if the the average long-term incurred cost is bounded, i.e. $J^{\mathrm{O}}(\pi) < \infty$.

Assuming nontrivial average cost $\inf_\pi J^{\mathrm{O}}(\pi) < \infty$ is possible, an optimal policy is stable. Since the cost c can be a distance measure from some target state, a stable policy is one that achieves bounded distance from the target state.

Lyapunov Functions. Lyapunov functions are standard tools in control theory to analyze the stability of a system (Meyn & Tweedie, 2012). Intuitively, they measure the "energy" of a state where the energy of a state is typically directly proportional to the cost of being in that state, and the energy of the target state is zero. A *control* Lyapunov function (CLF), $\ell : \mathcal{S} \to \mathbb{R}_{\geq 0}$, is one that satisfies: 1) $\liminf_{\|s\|\to\infty} \ell(s) = \infty$ and 2) for all but a finite number of states s: $\min_{a\in\mathcal{A}} \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[\ell(s') - \ell(s)] < 0$ i.e. there exists an action that in expectation decreases the energy ℓ in *one* step. It is known that finding a policy that outputs such actions will stabilize the system (Kellett & Teel, 2003). However, it is difficult to determine such a function without knowledge of transition dynamics especially in highly stochastic settings. In these cases, it is common to resort to an approximate CLF, where ℓ is decreased over multiple steps i.e. from a given state s, multiple actions are taken such that in the resulting state, s'', $\ell(s'') < \ell(s)$ (Taylor et al., 2019; Chang et al., 2019; Dai et al., 2021).

4. Why is Achieving Stochastic Stability Difficult for Deep RL?

In this section, we describe the core difficulties of achieving stochastic stability in highly stochastic MDPs with unbounded state spaces when minimizing the optimality cost objective. We do so using the queuing example in Figure 1.

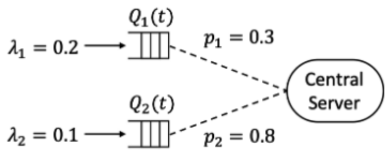


Figure 1. 2-queue network setting where a central server must select which of the two queues to serve. Q_i, λ_i, and p_i are the queue length, arrival rate, and service probability of queue i at each time-step. The image is taken from Liu et al. (2019).

The queue lengths at time t are given by $Q_i(t)$. The state is the vector of queue lengths i.e. $[Q_1(t), Q_2(t)]$, which is from an unbounded set since the queue lengths can grow arbitrarily large. The actions are discrete, denoting which queue the agent chose. In queuing theory, the optimality cost function is the total queue length of the system at the given time-step i.e. $c(s, a, s') = \|s'\|_1$ (see Appendix C.3 and

Leon-Garcia (2008)). An optimal policy is one that selects queues at each time-step such that the long-term average queue length is minimized. A stable policy ensures queue lengths are finite. In this specific example, the destabilizing action is one that serves an empty queue while the other queue is non-empty. If the agent too often chooses an empty queue, it runs the risk of the other queue blowing up in length. Thus, it is important that the agent quickly realizes that it should not serve the empty queue. Please refer to Appendix C for an in-depth description of the environment.

Challenge 1: Poor Credit Assignment. Credit assignment is one of the central challenges of RL. It is particularly, challenging in *information-sparse* MDPs (Arumugam et al., 2021; Pignatelli et al., 2023), where due to high stochasticity in the transition dynamics, $\mathcal{P}$, a single action lacks influence on the MDP. Intuitively, this lack of influence arises when an action has unintended consequences due to the stochasticity of the system. For example, if the server successfully decreases the length of say Q_1, but then a new job simultaneously arrives then from the agent's perspective, it appears that the action failed. Note that this challenge is separate from the typical sparse $0/1$ cost signal widely cited in the RL literature (Pignatelli et al., 2023; Harutyunyan et al., 2019); in our case, the cost function is dense, but the high stochasticity makes the MDP information-sparse.

In the queuing environment, a good action could have bad consequences (e.g., the server selects a non-empty queue, but due to stochasticity fails to serve it and a new job arrives) or a bad action could have neutral consequences (e.g., the server selects an empty queue and state remains the same). During learning, this noisiness can make distinguishing good actions from bad challenging since bad actions may appear good, and vice-versa. Ideally, serving an empty queue is adequately discouraged to avoid this confusion.

Challenge 2: Poor Extreme Generalization. The success of neural networks has primarily been due to their ability to *locally* generalize when given a *large* number of training samples per input (Chollet, 2017). However, when the destabilizing actions are inadequately discouraged in the unbounded state space setting, the queue lengths may be blowing up, which results in the agent to drift away into new states s. This drift means that: 1) the neural network must perform *extreme* generalization (e.g. generalize from queue lengths of 10 to 10^4) and 2) number of training samples for each previously-visited states s is *small*. In this situation, however, neural networks are known to output inaccurate predictions (Chollet, 2017; Xu et al., 2021; Haley & Soloway, 1992; Barnard & Wessels, 1992). Thus, the inaccuracy of the agent's neural network inhibits the agent from performing effectively in its environment.

In Section 6, we show that online deep RL agents that do not handle these challenges perform poorly. In the next section, we present a cost shaping approach that explicitly encourages stability, which leads to discouragement of unstable actions, and a state transformation approach that makes deep RL algorithms more amenable to better generalization.

5. STOP: STability and OPtimality

We now introduce our method, STability and OPtimality (STOP), an approach based on two ideas: 1) Lyapunov-based cost shaping that explicitly optimizes for stability in addition to optimality and 2) state transformations for better generalization across unbounded state samples. In Appendix B, we include the pseudo-code.

5.1. Lyapunov-based Cost Shaping

Recall from Section 3, that an action that solves $\min_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}[\ell(s') - \ell(s)] < 0$ leads to stabilizing the system. Our approach is to learn a policy that directly tries to do so. Thus, the agent minimizes the following stability and optimality objective:

$$J^{\mathrm{S}}(\pi) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi [\underbrace{\ell(s_{t+1}) - \ell(s_t)}_{\text{stability}} + \underbrace{c(s_t, a_t, s_{t+1})}_{\text{optimality}}] \tag{2}$$

The Lyapunov function ℓ can take on many forms. In this section, we explain the intuition behind selecting ℓ using the 1) linear function: $\ell_1 := \|s\|_1$ and 2) quadratic form: $\ell_2 := \|s\|_2^2$ where $\|\cdot\|_{\{1,2\}}$ is the $\{1,2\}$-norm, and the true optimality cost is c. These choices are inspired by the literature on stochastic networks and control (Srikant & Ying, 2014; Neely, 2010). As noted in Section 3, ℓ can be an approximate Lyapunov function—the above choices of ℓ are not exact in general. As shown in our experiments, an approximate Lyapunov function, could still offer significant benefits. In Section 5.1.3, we draw a connection between J^{S} and potential-based shaping (Ng et al., 1999).

5.1.1. Intuition for Encouraging Stability

To understand why the Lyapunov-based shaping leads to stability, we analyze the advantage functions of destabilizing actions of a PPO agent in the queuing problem given in Figure 1. While in general it is unknown which actions destabilize the system, in the particular case of Figure 1, the destabilizing actions are those that serve empty queues when a non-empty queue is available. These actions should have positive and high advantage function estimates (recall we are minimizing costs).

We analyze the normalized advantages of an average-reward PPO agent (Zhang & Ross, 2021; Schulman et al., 2017) at the *start* of learning before any policy updates. Initially, the agent acts uniformly at random until it fills up the rollout buffer. The advantages are then computed using this data when the agent optimizes: 1) $c(s, a, s')$, 2) $\ell_1(s') - \ell_1(s) + c(s, a, s')$ (linear), and 3) $\ell_2(s') - \ell_2(s) + c(s, a, s')$ (quadratic). We can then compare the estimated normalized advantages for destabilizing actions. In Table 1, we can see that with cost-shaping destabilizing actions are discouraged much more than without cost-shaping, which accordingly makes subsequent performance of the agent better. Without cost-shaping, destabilizing actions are assigned a relatively lower advantage contributing to less discouragement of destabilizing actions.

Cost Function	**Normalized Advantages**
$c(s, a, s')$	0.09 ± 0.07
$c(s, a, s') + \ell_1(s') - \ell_1(s)$	0.25 ± 0.08
$c(s, a, s') + \ell_2(s') - \ell_2(s)$	0.25 ± 0.05

Table 1. Interquartile mean (IQM) statistics (Agarwal et al., 2021) of the normalized advantage estimates for unstable actions across 20 trials in the 2-queue example from Figure 1. Higher is better. To account for the variations in magnitudes, we divide the computed advantages by their standard deviation across the rollout buffer of size 128. Note that since we are *minimizing costs* rather than maximizing rewards, bad actions should have positive advantage.

5.1.2. Choosing a Lyapunov Function

In this section, we provide intuition on the choice of the fixed Lyapunov function by drawing from results on queuing theory and stochastic control. We first note that from Equation 2, we have it that the Lyapunov function is essentially used for potential-based reward shaping. Ng et al. (1999) note that the ideal potential function for reward shaping is the optimal value function i.e. $\ell(s) = V^*(s)$. However, the need for $V^*(s)$ creates a chicken-and-egg problem where to learn $V^*(s)$, we need an optimal policy, but we want to find the optimal policy. Thus, ideally we can resort to a Lyapunov function such that $\ell(s) \approx V^*(s)$. Moreover, intuitively, we would expect that since the stability cost $\ell(s') - \ell(s)$ models a hill-climbing strategy, we would expect *any* CLF ℓ to yield practical benefits. However, we empirically find that this expectation often fails.

Our selection for a Lyapunov function is inspired by the analysis of the popular MaxWeight (MW) algorithm in queuing (Stolyar, 2004). There are variants of MW under a Lyapunov function of the form $\sum_i^n Q_i^{(\beta+1)}$ for $\beta > 0$ (Stolyar, 2004), where $\beta = 0$ corresponds to the linear Lyapunov function, and $\beta = 1$ corresponds to the quadratic function and classical MW algorithm. This algorithm has been proven to achieve a queue length that is at most within a factor of two from the optimal in certain heavy traffic regime of some queuing systems (Maguluri & Srikant, 2016). For the MW variant with general β, the algorithm is stable but their delay performance is unclear compared to the classical choice of $\beta = 1$. Our selection is also based on the fact that using the linear Lyapunov function ($\beta = 0$) is difficult to ensure stability (Krishnasamy et al., 2019) and is typically used in very simple systems (Fayolle et al., 1993). Motivated by these prior analyses, we explore various values of β.

In Section 6, we explore a range of β values, and find that as β gets larger, the performance of the RL agent improves. However, once β gets too large, performance deteriorates. Intuitively, this observation is expected: as we increase β, $\ell(s)$ starts to approximate $V^*(s)$ better based on the MW analysis; however, once β becomes too large, the high variance of the large unbounded costs causes performance to degrade (Gupta et al., 2023). Therefore, when we use domain-knowledge to select ℓ in unbounded state space problems, we want ℓ such that it approximates $V^*(s)$ and its variance is not significantly high.

5.1.3. Theoretical Properties

We investigate the theoretical properties of Lyapunov-based cost shaping. All proofs are provided in Appendix A. The following proposition characterizes the relation between the reshaped objective J^{S} and original optimality objective J^{O}.

Proposition 5.1. *For any policy π,*

$$J^{\mathrm{S}}(\pi) = J^{\mathrm{O}}(\pi) + \lim_{T\to\infty} \frac{\mathbb{E}_\pi[\ell(s_{T+1})]}{T}.$$

Therefore $J^{\mathrm{S}}(\pi) \geq J^{\mathrm{O}}(\pi)$, and $J^{\mathrm{S}}(\pi) = J^{\mathrm{O}}(\pi)$ if and only if $\mathbb{E}_\pi[\ell(s_{T+1})] = o(T)$.

The next proposition proves that under mild conditions, our approach is valid and will recover the optimal policy with respect to the original objective J^{O}.

Proposition 5.2. *Suppose that for any optimal policy $\pi^\star$, $\limsup_{T\to\infty} \mathbb{E}_{\pi^\star}[\ell(s_T)] < \infty$. Then $\arg\min_\pi J^{\mathrm{S}}(\pi) = \arg\min_\pi J^{\mathrm{O}}(\pi)$.*

The conditions for Proposition 5.2 are easily satisfied as long as the user-specified Lyapunov function ℓ is not growing too rapidly. For instance, if $\ell = \|\cdot\|_1$, then the above conditions hold when any optimal policy induces a stationary distribution and is stable (in the sense of Definition 3.1). If ℓ grows polynomially (e.g. $\ell = \|\cdot\|_2^2$), then it suffices for optimal policies to induce subexponential stationary distributions, which is true of a wide class of queuing problems and is ensured by standard technical assumptions (Hajek, 1982; Shah et al., 2020). Proposition 5.2 is analogous to the main result of Ng et al. (1999), which considers the discounted return criterion. They also make more stringent assumptions than ours, as in the unbounded state setting they require a uniformly bounded shaping term, which prohibits the use of Lyapunov functions.

5.2. State Transformations

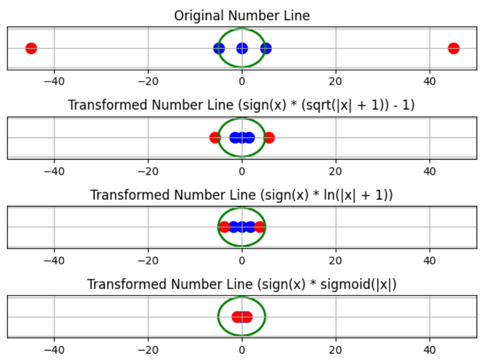


Figure 2. State transformations applied to 1D original untransformed points. Extreme points (red) appear closer to samples that agent was trained on (blue), thereby mitigating extreme generalization burdens. The green circle's radius is 5 units and provides a sense of proximity of the points.

In order to improve generalization in unbounded state spaces during training for more accurate credit assignment, we propose to use state transformations to compress the unbounded state space. These transformations are applied coordinate-wise to the state features, which are unbounded, before inputting them into the policy and value networks; the cost function is computed based on the original state.

Neural networks are known to perform *extreme generalization* poorly (Chollet, 2017; Xu et al., 2021; Haley & Soloway, 1992; Barnard & Wessels, 1992), which is problematic in the unbounded state space setting where states can be arbitrarily far from each other. Through Figure 2, we provide some intuition on how state transformations can mitigate the burden of extreme generalization. Figure 2 shows how a set of points are transformed by three different transformation functions. The blue dots are points that the neural network has already trained on and the red extreme points are the points the neural network must generalize to. We can see that in the original number line, the red points are "far" from the blue points, which can make generalization challenging. However, when we apply different state transformations, we can shrink the distance between points while preserving ordering, thus mitigating the burden of extreme generalization.

We consider the following transformation functions: 1) symmetric square root: $\text{symsqrt}(x) := \text{sign}(x)(\sqrt{|x|+1} - 1)$ (Kapturowski et al., 2019), 2) symmetric natural log: $\text{symloge}(x) := \text{sign}(x)\ln(|x|+1)$ (Hafner et al., 2023), and 3) $\text{symsigmoid}(x) := \text{sign}(x)(1/(1+e^{-|x|}))$. All these functions 1) are bi-jections and 2) preserve the ordering of points. Of course, there are trade-offs between transformations. For example, while symsigmoid reduces the generalization burden, it may effectively collapse the large states, making it challenging to differentiate between states and features. On the other hand, with symloge, the agent is still burdened by extrapolation but representation collapse is far less severe.

6. Empirical Study

We present an empirical study of STOP on various challenging stochastic control tasks with unbounded state spaces. We seek to answer the questions:

1. Does STOP enable learning of stable policies in settings where deep RL algorithms fail to?
2. Are both the stability cost and state transformation components essential for learning stable policies?

6.1. Setup

We first describe the environments, the algorithms we evaluate, and how we evaluate performance. We refer the reader to Appendix C for more details.

Environments. We conduct our experiments on the following environments. In each domain, the optimality cost function is the one typically used in the literature (Leon-Garcia, 2008; Ault & Sharon, 2021).

Single-server allocation queuing: We consider three variants of the 2-queue setup: 1) medium load with no faulty connections (Figure 1), 2) high load with faulty connections, and 3) very high load with faulty connections. The optimality cost function is the average queue length and the target state is all-zero queue lengths.

N-model network: We consider three setups of the N-network model (Harrison, 1998) : 1) high load, 2) very high load #1, and 3) very high load #2. The optimality cost function is the average holding cost and the target state has 0 holding cost.

Traffic control: We also evaluate our approach on the SUMO (Behrisch et al., 2011) traffic simulator, but due to space constraints defer to Appendix C. In this domain, SUMO models a real-life traffic situation by bounding the number of cars per lane. However, the success of STOP and failure of existing deep RL algorithms suggests our ideas are even applicable to the *bounded* state space setting. The optimality cost function is the total waiting time and the target state has 0 waiting time.

Algorithms. Our baseline is average-reward PPO (Zhang & Ross, 2021) (denoted by O since it optimizes the optimality criterion only) since it is designed: 1) for the infinite horizon setting without discounting (Naik et al., 2021) and 2) to be robust to hyperparameter tuning. For the server allocation and N-model networks environments, we also evaluate MAXWEIGHT (Tassiulas & Ephremides, 1990), a strong baseline algorithm with knowledge of the transition dynamics. In particular, MAXWEIGHT is known to be asymptotic optimal in heavy-traffic regimes (Maguluri & Srikant, 2016). When PPO is equipped with our cost-shaping approach and/or state transformations, we refer to it as STOP. MW starts the online interaction process with perfect estimation of some part of the transition dynamics, while STOP must learn a stable and optimal policy without this knowledge. Note that: 1) the optimal policy is unknown in our environments (Ganti et al., 2007; Dai & Gluzman, 2022), 2) it is generally unknown how far MAXWEIGHT is from optimal (Tassiulas & Ephremides, 1990), and 3) computing the optimal policy with full knowledge of the transition dynamics is not straightforward since there are infinite states in the unbounded state setting (Shah et al., 2020).

Online Evaluation. The agent starts from a random start state with a randomly-initialized policy and is never reset. We plot the true cost incurred by the agent vs. interaction time-steps. An increasing curve indicates instability as $c(s, a, s') \to \infty$, so the agent is drifting away from the target state; a flat curve indicates stability as $c(s, a, s') < \infty$; and a decreasing curve indicates improvement towards optimality as $c(s, a, s') \to 0$.

6.2. Stability Encouragement

Our first set of experiments aims to determine whether STOP stabilizes the agent. We answer two questions: 1) does the number of destabilizing actions taken reduce over training? and 2) is the STOP agent within a low bounded L_1 distance from the all-zero queue lengths with high probability?

In answering both these questions, we use the quadratic Lyapunov function, $\ell_2(s) = \|s\|_2^2$ for our shaping cost with no state transformations on the queuing network shown in Figure 1. We answer the first question with Figure 3(a) where we plot the fraction of destabilizing actions taken by the PPO agent when it is optimizing with (STOP) and without the shaped cost. We calculate destabilizing actions as the fraction of (s, a) samples in the rollout buffer which serve an empty queue. We can see that the STOP quickly stops taking destabilizing actions, while $\approx 40\%$ of the actions taken by PPO without cost-shaping are destabilizing.

We answer the second question with Figure 3(b) and (c). In Figure 3(b) and Figure 3(c), we plot the state-visitation distribution of the agent after 100K interaction steps. We see that the STOP agent visits states that have an L_1 distance of at most 20 with probability 0.95, while the vanilla PPO agent does not even appear in this bounded region.

6.3. Main Results

We now compare vanilla PPO algorithm to PPO equipped with STOP in Figure 4 on a variety of highly stochastic MDPs. We set the rollout buffer length to 200 and keep all other hyperparameters for STOP and the baseline the same (Huang et al., 2022), and show results for agents that achieved the lowest true optimality cost at the end of interaction time. We evaluate the following stability variations: $\ell(s) = \|s\|_p^p$, where $p = \{1, 1.5, 2, 2.5, 3, 4, 5\}$. All STOP agents use the symloge transform.

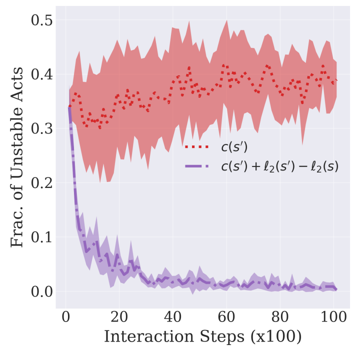
In all experiments, the naïve PPO agents, which optimized the optimality criterion directly and used no state transformations, were unstable. On the other hand, STOP agents are able to achieve stability. In $4/6$ cases, we found STOP out-performed or was extremely competitive with MW. To the best of our knowledge, no online RL algorithm in unbounded state spaces has outperformed MW. In very high load environments, such as Figure 4(f) we find that the high load causes all algorithms to be unstable.

In the highly stochastic queuing environments only, we found that using the linear stability cost and optimality cost is insufficient to achieve stability: the linear stability cost (bounded between $[-1, 1]$) and optimality cost have different magnitude scales, which can cause the latter to dilute the former, effectively eliminating any benefit of the stability cost. In these cases, we replaced the optimality cost c with $-1/(c + 1)$. On the other hand, we were able to keep the original optimality cost for all the other variations. While the linear agent ($p = 1$) can stabilize the system, the use of the reciprocal may contribute to the relatively poor performance compared to the agents ($p \neq 1$).

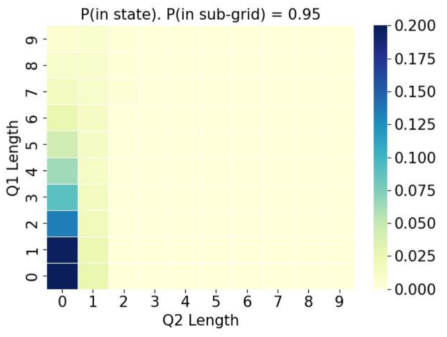
In general, we found that moderate values of p achieved the lowest average queue length or holding cost in high load settings. For example, in $p = 3$ (cubic; gray line) performed reasonably well in all the environments. While $p = 1$ (linear; green line) and $p = 5$ (light blue line) performed well in only two or three environments (see: (a), (b), (d), (e)). The ability of $p = 1$ to achieve low average queue length in settings (a) and (b) corresponds to the fact that the policy that serves the queue with the largest service rate ($c\mu$-rule) performs well in these settings (Buyukkoc et al., 1985). These results align with our expectation: smaller values of p may inadequately discourage de-stabilizing actions and are potentially poorer approximations of the optimal value function, while larger p values result in high variance of the unbounded cost, which can degrade performance.

6.4. Ablation Studies

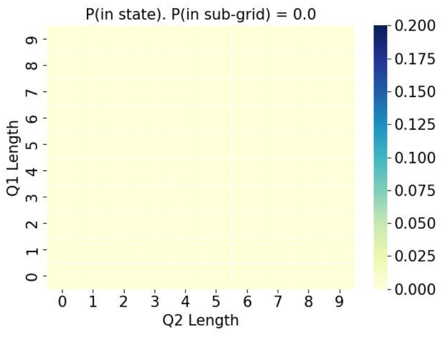
We have shown that STOP enables learning in highly stochastic environments with unbounded state spaces. We now analyze the importance of different components of STOP.



(a) Fraction of unstable actions taken

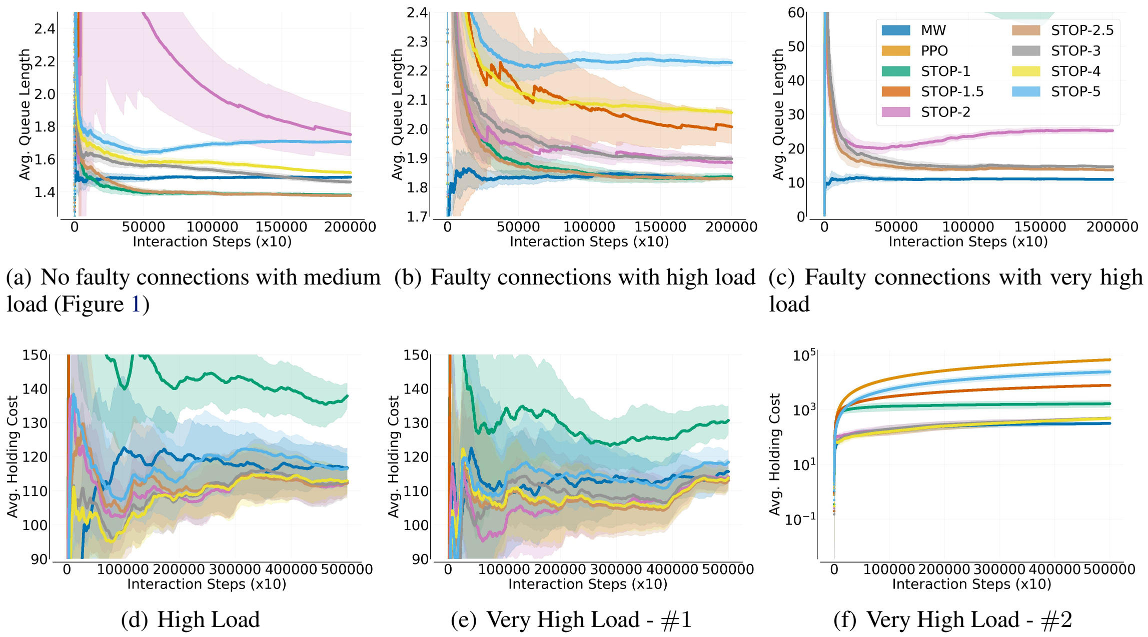


(b) STOP state visitation distribution



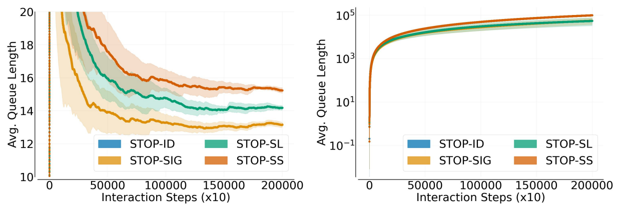
(c) PPO state visitation distribution

Figure 3. Stability verification. (a) fraction of unstable actions taken by the agent over the course of training of STOP $(c(s')+\ell_2(s')-\ell_2(s))$ and PPO $(c(s'))$ agents; lower is better. (b) and (c) state-visitation distribution of STOP and PPO agents respectively; higher $\Pr(\text{in sub-grid})$ is better. Note that the empty region of (c) shows the failure of the PPO agent to visit the specified bounded region near the target state. All quantities were computed over 10 trials on the 2-queue setting from Figure 1.



(a) No faulty connections with medium load (Figure 1)
(b) Faulty connections with high load
(c) Faulty connections with very high load
(d) High Load
(e) Very High Load - #1
(f) Very High Load - #2

Figure 4. True optimality criterion vs. interaction time-steps on three server-allocation queue networks (top row) and three N-network model environments (bottom row). Lower is better. Algorithms are PPO vs. STOP-p, where p denotes the power of the Lyapunov function. All STOP variants use the symloge state transformation. We also report the performance of MAXWEIGHT (MW). Recall that unlike MW, STOP does not know the transition dynamics. The IQM (Agarwal et al., 2021) is computed of the performance metrics over 20 trials with 95% confidence intervals. The vertical axis of (f) is log-scaled. To enhance visibility we zoom into the plot, which hides performance of O in some cases since it was unstable. We refer the reader to Appendix C for the zoomed-out plots.



(a) STOP with varying state transformations (b) Only state transformations, no stability cost

Figure 5. Ablations. True optimality criterion vs. interaction timesteps on 2 queue setup with faulty connections. Lower is better. The IQM (Agarwal et al., 2021) is computed of the performance metrics over 20 trials with 95% confidence intervals. The vertical axis of (b) is log-scaled. ID: identity; SIG: symsigmoid, SS: symsqrt; SL: symloge. Refer to Appendix C for the zoomed-out plots.

6.4.1. STOP with Varying State Transformations

In this experiment, we seek to understand the importance of the state transformation component of STOP. We evaluate the performance of a STOP PPO agent with $\ell(s) = \|s\|_3^3$ (cubic) and different state transformations from Section 5.2. From Figure 5(a), applying state transformations result in significantly better performance compared to no application of such a transformation. The significant improvement over no state transformations (ID) suggests that state transformations are indeed critical and that in very difficult environments the stability cost is insufficient. For different transformations, the agent faces different levels of extreme generalization burden. When the agent uses sigmoid, all its extreme generalization burden is mitigated, while when it uses the squareroot it suffers more of the burden compared to log and sigmoid.

6.4.2. State Transformations with No Stability Cost

In Figure 5(b), we remove the stability cost. We observe that applying only state transformations is insufficient for the agent to stablize and the stability cost component of STOP is critical. This result also confirms that simply bounding the state space (such as applying sigmoid) is insufficient to stabilize the system.

7. Conclusion

Our work showed that online deep RL agents that directly try to be optimal in highly stochastic environments with unbounded state spaces actually end up unstable. To address this instability, we introduced STOP that explicitly encourages stability and optimality. We provided insight on how to leverage domain knowledge from queuing theory to appropriately select Lyapunov functions that: 1) approximate the optimal value function, 2) adequately discourage de-stabilizing actions, and 3) have low variance. We showed that STOP trained highly performant deep RL policies in these difficult environments, and even out-performed strong baselines from the queuing literature. STOP improves the reliability of online deep RL policies on challenging stochastic control problems by ensuring that the policies are within bounded distance from the desired state.

A key takeaway is that deep RL algorithms that are successful on MuJoCo and Atari may fail to generalize well to environments with high stochasticity and unbounded state spaces. We shed light on the fact that there are real-world-inspired problems that appear simple, but are surprisingly challenging, and that as a community we must innovate new techniques to solve these challenging problems.

In this work, we showed that dense optimality cost functions used in real-world-inspired domains could be poor learning signals. An interesting future direction would be to formally investigate why they are so beyond their inability to do proper credit assignment. Another interesting future direction would be to explore how we can learn an appropriate Lyapunov function from data.

Impact Statement

Our work is largely focused on studying fundamental RL research questions, and thus we do not see any immediate negative societal impacts. The aim of our work is to improve the reliability of deep RL algorithms on real-world-inspired domains. By leveraging STOP, a user takes a step towards improving the reliability of their decision-making agents.

Acknowledgements

The authors thank Abhinav Narayan Harish, Adam Labiosa, Andrew Wang, Lucas Poon, and the anonymous reviewers at NeurIPS, ICLR, and ICML for their feedback on earlier drafts of this work. Y. Chen and M. Zurek were supported in part by National Science Foundation Awards CCF-2233152 and DMS-2023239. Q. Xie was supported in part by National Science Foundation Awards CNS-1955997 and EPCN-2339794. J. Hanna and B. Pavse were supported in part by American Family Insurance through a research partnership with the University of Wisconsin—Madison's Data Science Institute.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

Alegre, L. N. SUMO-RL. https://github.com/LucasAlegre/sumo-rl, 2019.

Arumugam, D., Henderson, P., and Bacon, P. An information-theoretic perspective on credit assignment in reinforcement learning. *CoRR*, abs/2103.06224, 2021. URL https://arxiv.org/abs/2103.06224.

Ault, J. and Sharon, G. Reinforcement learning benchmarks for traffic signal control. In *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track*, December 2021.

Barnard, E. and Wessels, L. Extrapolation and interpolation in neural network classifiers. *IEEE Control Systems Magazine*, 12(5):50–53, 1992. doi: 10.1109/37.158898.

Behrisch, M., Bieker-Walz, L., Erdmann, J., and Krajzewicz, D. Sumo – simulation of urban mobility: An overview. volume 2011, 10 2011. ISBN 978-1-61208-169-4.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pp. 908–918, 2017.

Bertsekas, D. P. Dynamic programming and optimal control, 4th edition, volume II. *Athena Scientific*, 2015.

Buyukkoc, C., Varaiya, P. P., and Walrand, J. C. The cμ rule revisited. *Advances in Applied Probability*, 17:237 – 238, 1985. URL https://api.semanticscholar.org/CorpusID:125747190.

Chang, Y.-C., Roohi, N., and Gao, S. Neural lyapunov control. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2647c1dba23bc0e0f9cdf75339e120d2-Paper.pdf.

Chen, A. S., Sharma, A., Levine, S., and Finn, C. You Only Live Once: Single-Life Reinforcement Learning, October 2022. URL http://arxiv.org/abs/2210.08863. arXiv:2210.08863 [cs].

Chollet, F. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017. ISBN 1617294438.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 8092–8101, 2018.

Dai, H., Landry, B., Yang, L., Pavone, M., and Tedrake, R. Lyapunov-stable neural-network control, 2021.

Dai, J. G. and Gluzman, M. Queueing network controls via deep reinforcement learning. *Stochastic Systems*, 12(1): 30–67, 2022. doi: 10.1287/stsy.2021.0081. URL https://doi.org/10.1287/stsy.2021.0081.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Dean, S., Tu, S., Matni, N., and Recht, B. Safely learning to control the constrained linear quadratic regulator. In *2019 American Control Conference (ACC)*, pp. 5582–5588. IEEE, 2019.

Dohare, S., Lan, Q., and Mahmood, A. R. Overcoming policy collapse in deep reinforcement learning. In *Sixteenth European Workshop on Reinforcement Learning*, 2023. URL https://openreview.net/forum?id=m9Jfdz4ymO.

Eysenbach, B., Gu, S., Ibarz, J., and Levine, S. Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning. February 2018. URL https://openreview.net/forum?id=S1vuO-bCW.

Fayolle, G., Malyshev, V. A., Menshikov, M. V., and Sidorenko, A. F. Lyapunov functions for jackson networks. *Math. Oper. Res.*, 18(4):916–927, nov 1993. ISSN 0364-765X.

Ganti, A., Modiano, E., and Tsitsiklis, J. N. Optimal transmission scheduling in symmetric communication models with intermittent connectivity. *IEEE Transactions on Information Theory*, 53(3):998–1008, March 2007. ISSN 1557-9654. doi: 10.1109/TIT.2006.890695.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Gupta, A., Yu, J., Zhao, T. Z., Kumar, V., Rovinsky, A., Xu, K., Devlin, T., and Levine, S. Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention, April 2021. URL http://arxiv.org/abs/2104.11203. arXiv:2104.11203 [cs].

Gupta, D., Chandak, Y., Jordan, S. M., Thomas, P. S., and da Silva, B. C. Behavior alignment via reward function optimization, 2023.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models, 2023.

Hajek, B. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied probability*, 14(3):502–525, 1982.

Haley, P. and Soloway, D. Extrapolation limitations of multilayer feedforward neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pp. 25–30 vol.4, 1992. doi: 10.1109/IJCNN.1992.227294.

Han, W., Levine, S., and Abbeel, P. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6435–6442, September 2015. doi: 10.1109/IROS.2015.7354297.

Hans, A., Schneegaß, D., Schäfer, A. M., and Udluft, S. Safe exploration for reinforcement learning. In *ESANN*, pp. 143–148, 2008.

Harrison, J. M. Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies. *The Annals of Applied Probability*, 8(3):822 – 848, 1998. doi: 10.1214/aoap/1028903452. URL https://doi.org/10.1214/aoap/1028903452.

Harutyunyan, A., Dabney, W., Mesnard, T., Azar, M., Piot, B., Heess, N., van Hasselt, H., Wayne, G., Singh, S., Precup, D., and Munos, R. Hindsight credit assignment, 2019.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Kapturowski, S., Ostrovski, G., Dabney, W., Quan, J., and Munos, R. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

Kellett, C. and Teel, A. Results on discrete-time control-lyapunov functions. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 6, pp. 5961–5966 Vol.6, 2003. doi: 10.1109/CDC.2003.1271964.

Koller, T., Berkenkamp, F., Turchetta, M., and Krause, A. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066. IEEE, 2018.

Krishnasamy, S., Sen, R., Johari, R., and Shakkottai, S. Learning unknown service rates in queues: A multi-armed bandit approach, 2019.

Leon-Garcia, A. *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson/Prentice Hall, Upper Saddle River, NJ, third edition, 2008. ISBN 9780131471221 0131471228.

Liu, B., Xie, Q., and Modiano, E. Reinforcement learning for optimal control of queueing systems. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 663–670, 2019. doi: 10.1109/ALLERTON.2019.8919665.

Liu, B., Xie, Q., and Modiano, E. Rl-qn: A reinforcement learning framework for optimal control of queueing systems. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 7(1), aug 2022. ISSN 2376-3639. doi: 10.1145/3529375. URL https://doi.org/10.1145/3529375.

Maguluri, S. T. and Srikant, R. Heavy traffic queue length behavior in a switch under the maxweight algorithm. *Stochastic Systems*, 6(1):211–250, 2016.

Mahadevan, S. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1):159–195, March 1996. ISSN 1573-0565. doi: 10.1007/BF00114727. URL https://doi.org/10.1007/BF00114727.

Mao, H., Schwarzkopf, M., He, H., and Alizadeh, M. Towards safe online reinforcement learning in computer systems. 2019. URL https://api.semanticscholar.org/CorpusID:204978262.

Meyn, S. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022.

Meyn, S. P. and Tweedie, R. L. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

Naik, A., Shariff, R., Yasui, N., and Sutton, R. S. Discounted reinforcement learning is not an optimization problem. *CoRR*, abs/1910.02140, 2019. URL http://arxiv.org/abs/1910.02140.

Naik, A., Abbas, Z., White, A., and Sutton, R. S. Towards reinforcement learning in the continuing setting. 2021. URL https://drive.google.com/file/d/1xh7WjGP2VI_QdpjVWygRC1BuH6WB_gqi/view.

Neely, M. J. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010. ISBN 160845455X.

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

Pignatelli, E., Ferret, J., Geist, M., Mesnard, T., van Hasselt, H., and Toni, L. A survey of temporal credit assignment in deep reinforcement learning, 2023.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Schwartz, A. A Reinforcement Learning Method for Maximizing Undiscounted Rewards. pp. 298–305, December 1993. ISBN 978-1-55860-307-3. doi: 10.1016/B978-1-55860-307-3.50045-9.

Shah, D., Xie, Q., and Xu, Z. Stable reinforcement learning with unbounded state space. In Bayen, A. M., Jadbabaie, A., Pappas, G., Parrilo, P. A., Recht, B., Tomlin, C., and Zeilinger, M. (eds.), *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 581–581. PMLR, 10–11 Jun 2020. URL https://proceedings.mlr.press/v120/shah20a.html.

Sharma, A., Xu, K., Sardana, N., Gupta, A., Hausman, K., Levine, S., and Finn, C. Autonomous reinforcement learning: Formalism and benchmarking. *ArXiv*, abs/2112.09605, 2021.

Sharma, A., Ahmad, R., and Finn, C. A state-distribution matching approach to non-episodic reinforcement learning, 2022.

Srikant, R. and Ying, L. *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge University Press, USA, 2014. ISBN 1107036054.

Stolyar, A. L. Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. *The Annals of Applied Probability*, 14(1):1–53, 2004. ISSN 10505164. URL http://www.jstor.org/stable/4140489.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Tassiulas, L. and Ephremides, A. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*, pp. 2130–2132 vol.4, 1990. doi: 10.1109/CDC.1990.204000.

Taylor, A. J., Dorobantu, V. D., Le, H. M., Yue, Y., and Ames, A. D. Episodic learning with control lyapunov functions for uncertain robotic systems. *CoRR*, abs/1903.01577, 2019. URL http://arxiv.org/abs/1903.01577.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Vinogradska, J., Bischoff, B., Nguyen-Tuong, D., Romer, A., Schmidt, H., and Peters, J. Stability of controllers for gaussian process forward models. In *International Conference on Machine Learning*, pp. 545–554, 2016.

Wan, Y., Naik, A., and Sutton, R. S. Learning and Planning in Average-Reward Markov Decision Processes. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10653–10662. PMLR, July 2021. URL https://proceedings.mlr.press/v139/wan21a.html. ISSN: 2640-3498.

Wei, C.-Y., Jafarnia-Jahromi, M., Luo, H., Sharma, H., and Jain, R. Model-free Reinforcement Learning in Infinite-horizon Average-reward Markov Decision Processes, February 2020. URL http://arxiv.org/abs/1910.07072. arXiv:1910.07072 [cs, stat].

Wei, H., Liu, X., Wang, W., and Ying, L. Sample efficient reinforcement learning in mixed systems through augmented samples and its applications to queueing networks, 2023.

Westenbroek, T., Castaneda, F., Agrawal, A., Sastry, S., and Sreenath, K. Lyapunov design for robust and efficient robotic reinforcement learning. In *6th Annual Conference on Robot Learning*, 2022.

Xu, K., Zhang, M., Li, J., Du, S. S., Kawarabayashi, K.-I., and Jegelka, S. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=UH-cmocLJC.

Yu, M., Yang, Z., Kolar, M., and Wang, Z. Convergent policy optimization for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3121–3133, 2019.

Zhang, S., Wan, Y., Sutton, R. S., and Whiteson, S. Average-Reward Off-Policy Policy Evaluation with Function Approximation. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 12578–12588. PMLR, July 2021. URL https://proceedings.mlr.press/v139/zhang21u.html. ISSN: 2640-3498.

Zhang, Y. and Ross, K. W. On-policy deep reinforcement learning for the average-reward criterion. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12535–12545. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zhang21q.html.

Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., and Levine, S. The Ingredients of Real-World Robotic Reinforcement Learning, April 2020. URL http://arxiv.org/abs/2004.12570. arXiv:2004.12570 [cs, stat].

A. Theoretical Results

In this section we provide the proofs for our theoretical results, which are restated below for readers' convenience.

Proposition 5.1. *For any policy π,*

$$J^{\mathrm{S}}(\pi) = J^{\mathrm{O}}(\pi) + \lim_{T\to\infty} \frac{\mathbb{E}_\pi[\ell(s_{T+1})]}{T}.$$

Therefore $J^{\mathrm{S}}(\pi) \geq J^{\mathrm{O}}(\pi)$, and $J^{\mathrm{S}}(\pi) = J^{\mathrm{O}}(\pi)$ if and only if $\mathbb{E}_\pi[\ell(s_{T+1})] = o(T)$.

Proof. Fix a policy π. Then we can calculate that

$$\begin{aligned}
J^{\mathrm{S}}(\pi) &= \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_\pi\left[\sum_{t=1}^{T} \left(c(s_t, a_t, s_{t+1}) + \ell(s_{t+1}) - \ell(s_t)\right)\right] \\
&= \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_\pi\left[\ell(s_{T+1}) - \ell(s_1) + \sum_{t=1}^{T} c(s_t, a_t, s_{t+1})\right] \\
&= \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_\pi\left[\ell(s_{T+1})\right] + \frac{1}{T}\mathbb{E}_\pi\left[\sum_{t=1}^{T} c(s_t, a_t, s_{t+1})\right] \\
&= J^{\mathrm{O}}(\pi) + \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_\pi\left[\ell(s_{T+1})\right]
\end{aligned}$$

where we used the fact that $\lim_{T\to\infty} \frac{\ell(s_1)}{T} = 0$.

The fact that $J^{\mathrm{S}}(\pi) \geq J^{\mathrm{O}}(\pi)$ then follows from the fact that ℓ is non-negative. □

Proposition 5.2. *Suppose that for any optimal policy $\pi^\star$, $\limsup_{T\to\infty} \mathbb{E}_{\pi^\star}[\ell(s_T)] < \infty$. Then $\arg\min_\pi J^{\mathrm{S}}(\pi) = \arg\min_\pi J^{\mathrm{O}}(\pi)$.*

Proof. Fix a policy $\pi^\star$ which is optimal for J^{O}. From Proposition 5.1 we have that $J^{\mathrm{S}}(\pi^\star) \geq J^{\mathrm{O}}(\pi^\star)$. To show the reverse inequality, we can use Proposition 5.1 to calculate that

$$\begin{aligned}
J^{\mathrm{S}}(\pi^\star) &= J^{\mathrm{O}}(\pi^\star) + \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\pi^\star}\left[\ell(s_{T+1})\right] \\
&\leq J^{\mathrm{O}}(\pi^\star) + \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}_{\pi^\star}[\ell(s_T)] \\
&= J^{\mathrm{O}}(\pi^\star)
\end{aligned}$$

using the assumption that $\limsup_{T\to\infty} \mathbb{E}_{\pi^\star}[\ell(s_T)] < \infty$. Therefore $J^{\mathrm{S}}(\pi^\star) = J^{\mathrm{O}}(\pi^\star)$. Now for any other policy π, since $\pi^\star$ is optimal for J^{O}, we have that $J^{\mathrm{O}}(\pi^\star) \leq J^{\mathrm{O}}(\pi)$. Therefore

$$J^{\mathrm{S}}(\pi^\star) = J^{\mathrm{O}}(\pi^\star) \leq J^{\mathrm{O}}(\pi) \leq J^{\mathrm{S}}(\pi)$$

where we used the fact from Proposition 5.1 that $J^{\mathrm{S}}(\pi) \geq J^{\mathrm{O}}(\pi)$. Therefore $\pi^\star \in \arg\min_\pi J^{\mathrm{S}}(\pi)$, so $\arg\min_\pi J^{\mathrm{S}}(\pi) \supseteq \arg\min_\pi J^{\mathrm{O}}(\pi)$. Furthermore, if $\pi \notin \arg\min_\pi J^{\mathrm{O}}(\pi)$, then we must have $J^{\mathrm{O}}(\pi) > J^{\mathrm{O}}(\pi^\star)$, in which case by again using Proposition 5.1 we have that

$$J^{\mathrm{S}}(\pi) \geq J^{\mathrm{O}}(\pi) > J^{\mathrm{O}}(\pi^\star) = J^{\mathrm{S}}(\pi^\star)$$

so $\pi \notin \arg\min_\pi J^{\mathrm{S}}(\pi)$. Thus $\arg\min_\pi J^{\mathrm{S}}(\pi) \subseteq \arg\min_\pi J^{\mathrm{O}}(\pi)$ and we can conclude that $\arg\min_\pi J^{\mathrm{S}}(\pi) = \arg\min_\pi J^{\mathrm{O}}(\pi)$ as desired. □

B. STOP Pseudo-code

Algorithm 1 STOP+PPO

```
1: Input: policy parameters $\theta_0$, critic net parameters $\phi_0$, state transformation function $\sigma$, rollout buffer $\mathcal{D}$ of length $N$.
2: for t = 1, 2, ... do
3:     Collect sub-trajectory in rollout buffer $\{\sigma(s_k), a_k, \sigma(s_{k+1}), l_k\}_{k=1}^{N}$ from environment using $\pi_{\theta_{\lfloor t/N \rfloor}}$ {Note that rollout buffer contains the transformed states and the cost $l_k := \underbrace{\ell(s_{k+1}) - \ell(s_k)}_{\text{stability cost}} + \underbrace{c(s_k, a_k, s_{k+1})}_{\text{optimality cost}}$ is a function of the non-transformed states.}
4:     if $t\%N == 0$ then
5:         {Periodically update policy and critic parameters}
6:         Using rollout buffer $\mathcal{D}$ update $\theta$ and $\phi$ with average-reward PPO (Zhang & Ross, 2021).
7:         Empty $\mathcal{D}$
8:     end if
9:     Record performance of agent according to true optimality cost at time-step $t$, $c(s_t, a_t, s_{t+1})$, as a function of non-transformed states $\{s_t, s_{t+1}\}$.
10: end for
```

C. Supporting Content and Empirical Results

In this section, we include additional details and experiments that complement the main results. We also include the code in the supplementary zip file.

C.1. Visualizations of State Transformations

To provide better intuition of the different state transformations we considered in Section 5.2, we visualize them in Figure 6.

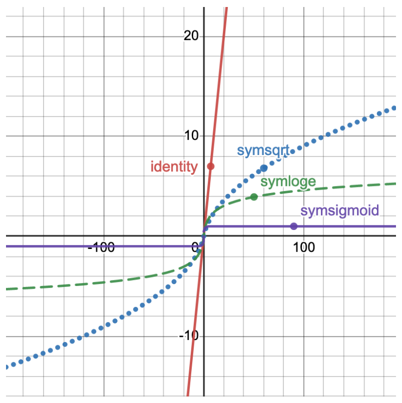


Figure 6. Visualizations of transformation functions.

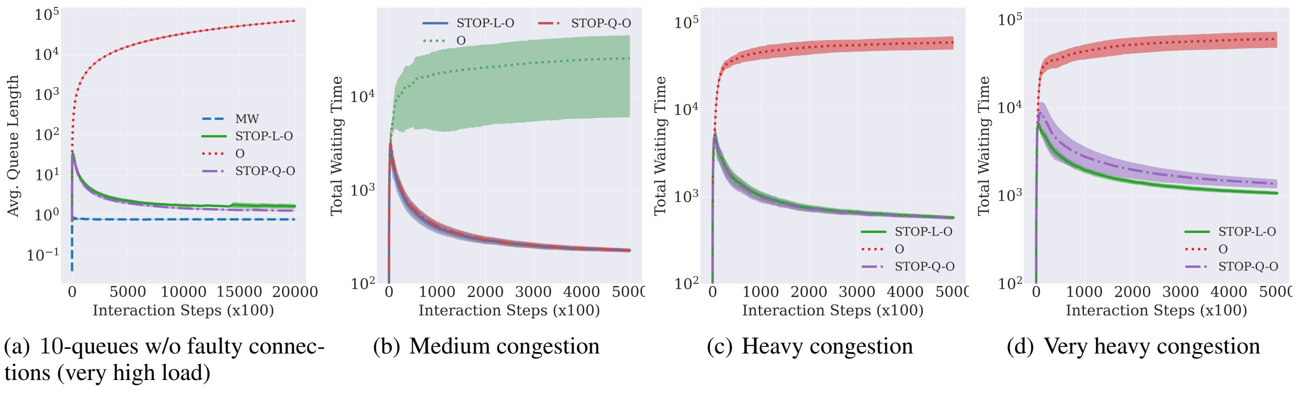
C.2. Additional Main Results

In this section, we include the remaining main results on a 10-queue server allocation problem and on the traffic control simulator. We also include the zoomed-out results from Figure 4 and Figure 5 in Figure 8.

C.3. Environments

In this section, we provide additional details about the environments.

Single-server allocation queuing In this environment, there is a single central server that must select among a set of queues to serve. In general, there can be up to N queues (Figure 9 show a sample 2 queue setup). At each time-step, new



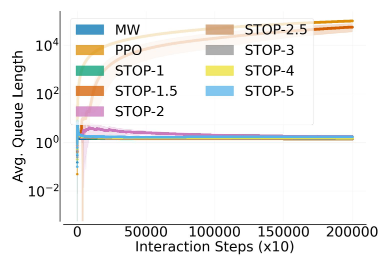
(a) 10-queues w/o faulty connections (very high load)
(b) Medium congestion
(c) Heavy congestion
(d) Very heavy congestion

Figure 7. True optimality criterion vs. interaction time-steps on 10-queue server allocation (first image) and traffic control environment (remaining three). Lower is better. Algorithms are PPO (O) vs. STOP + PPO which use all STOP components, where we evaluate the linear (STOP-L-O) and quadratic (STOP-Q-O) stability cost function. For the queuing environment, we also report the performance of MAXWEIGHT (MW). Recall that unlike MW, STOP does not know the transition dynamics. The IQM (Agarwal et al., 2021) is computed of the performance metrics over 15 trials with 95% confidence intervals.

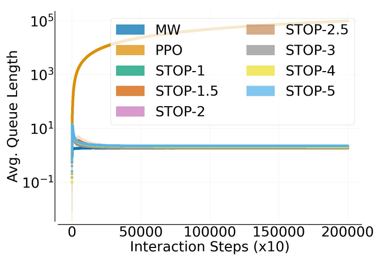
jobs arrive in each queue following a Bernoulli process with probability λ_i for queue i. Note that at each time-step, at most one job enters each queue, which means more than one job may enter the whole system in total. At each time-step, the server must select among the N queues to serve. A successfully served queue will mean a job will exit that queue, so at most one job can exit the system at a given time-step. When a server selects a queue i, the server succeeds in making a job exit only if: it can connect to queue i (which is dependent on the connectivity probability, c_i) and the job is successfully served (which is dependent on the service probability, p_i). The state of the server is queue length of each queue and $0/1$ flag indicating whether the server can connect to a specific queue, resulting in $2N$-dimensional state. The action space is index of the queue, resulting in N dimensions. The goal is to minimize the average queue length. In the non-faulty connection setting, all the connectivity flags are 1. The optimal policy in the faulty connection setting is an open problem (Ganti et al., 2007). We consider settings where the system is stabilizable i.e. $\sum_i^N \frac{\lambda_i}{p_i} < 1 - \prod_i^N (1 - c_i)$ and $\frac{\lambda_i}{p_i} < c_i$.

The Bernoulli probability parameters of the tested environments are:

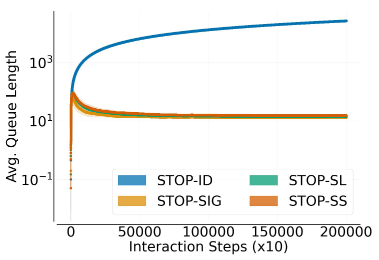
1. 2-queue *without* faulty connections (see Figure 1) (medium load)
   - Arrival rates: $\lambda_1 = 0.2$, $\lambda_2 = 0.1$
   - Service rates: $p_1 = 0.3, p_2 = 0.8$
   - Connection probabilities: $c_1 = 1, c_2 = 1$
2. 2-queue with faulty connections (high load)
   - Arrival rates: $\lambda_1 = 0.2$, $\lambda_2 = 0.1$
   - Service rates: $p_1 = 0.3, p_2 = 0.8$
   - Connection probabilities: $c_1 = 0.95, c_2 = 0.5$
3. 2-queue with faulty connections (very high load)
   - Arrival rates: $\lambda_1 = 0.2$, $\lambda_2 = 0.1$
   - Service rates: $p_1 = 0.3, p_2 = 0.8$
   - Connection probabilities: $c_1 = 0.7, c_2 = 0.5$
4. 10-queue with non-faulty connections (very high load)
   - Arrival rates: $\lambda_1 = 0.05, \lambda_2 = 0.01, \lambda_3 = 0.2, \lambda_4 = 0.4, \lambda_5 = 0.05, \lambda_6 = 0.01, \lambda_7 = 0.02, \lambda_8 = 0.01, \lambda_9 = 0.015, \lambda_{10} = 0.01$
   - Service rates: $p_1 = 0.9, p_2 = 0.85, p_3 = 0.95, p_4 = 0.75, p_5 = 0.9, p_6 = 0.9, p_7 = 0.85, p_8 = 0.9, p_9 = 0.9, p_{10} = 0.85$



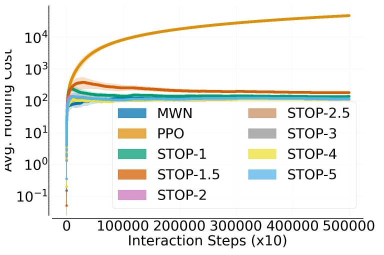
(a) Faulty connections with medium load (Figure 1)



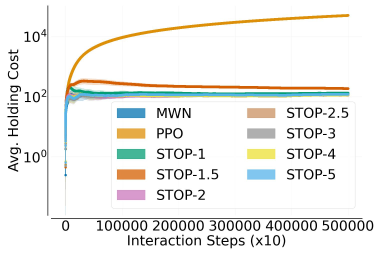
(b) Faulty connections with high load



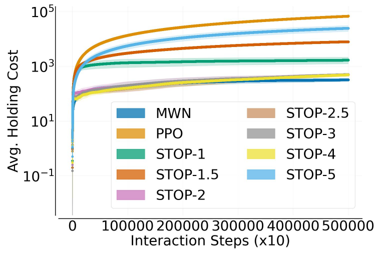
(c) Varying state transformation



(d) High Load



(e) Very High Load - #1



(f) Very High Load - #2

Figure 8. Zoomed out version of Figure 4 and Figure 5. True optimality criterion vs. interaction time-steps on three server-allocation queue networks (top row) and three N-network model environments (bottom row). Lower is better. Algorithms are PPO vs. STOP-p, where p denotes the power of the Lyapunov function. All STOP variants use the symloge state transformation. We also report the performance of MAXWEIGHT (MW). Recall that unlike MW, STOP does not know the transition dynamics. The IQM (Agarwal et al., 2021) is computed of the performance metrics over 20 trials with 95% confidence intervals. All vertical axes are log-scaled.

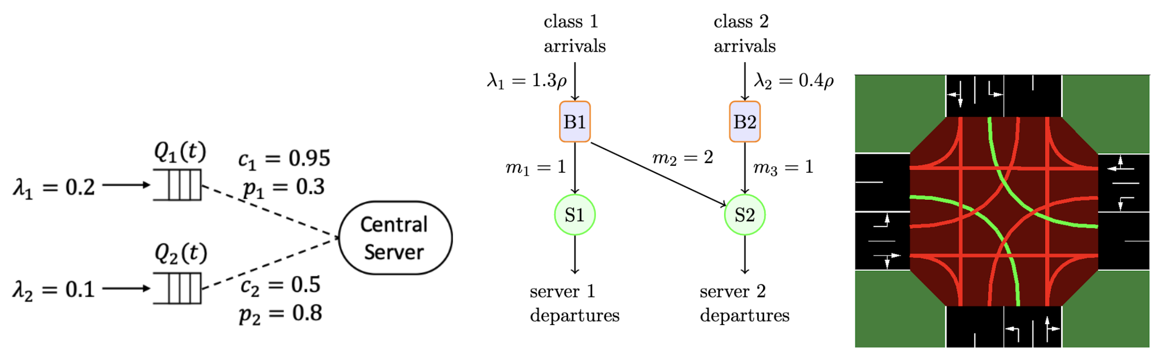
- Connection probabilities: $c_i = 1$ for all $1 \le i \le 10$

When evaluating the RL algorithms, we compare their performance to MAXWEIGHT (Tassiulas & Ephremides, 1990; Stolyar, 2004), a well-known algorithm that achieves stability for a certain class of queuing scenarios, but which relies on the knowledge of the system model (i.e., some parts of the transition dynamics) and it is generally unknown how far MAXWEIGHT is from optimality. It is a very strong non-RL baseline from decades of research from the stochastic networking community.

2-model Network In this domain, there are two queues of jobs for class 1 and class 2 jobs, B_1 and B_2. These jobs come into the queues following a Poisson process with arrival rates λ_1 and λ_2, which is a function of ρ which determines the load. Class 1 jobs can be processed by server S1 as well as server S2, while class 2 jobs can only be processed by server S2. The success rate of S2 serving class 1 jobs is given by the service rate $\mu_2 = 1/m_2$ and of serving class 2 jobs is $\mu_3 = 1/m_3$. Similarly, success of S1 serving class 1 jobs is given by $\mu_1 = 1/m_1$. All these success rates are exponentially distributed with mean m_i. The holding cost of keeping a job waiting in B_1 is 3 and in B_2 is 1. The agent in this case is S2, which must decide whether to serve class 1 jobs or class 2 jobs. Its state is the queue lengths of B_1 and B_2. Its action is the discrete action denoting the index of the selected queue. The optimality criterion is the average holding cost per time-step i.e. $3x_1 + x_2$, where x_i is the number of waiting jobs in queue i. For more details, refer to Dai & Gluzman (2022). We refer to their code for the environment: `https://github.com/mark-gluzman/NmodelPPO/blob/master/NmodelDynamics.py`. We consider settings where i.e. $\frac{\lambda_1 - \mu_1}{\mu_2} \le 1 - \frac{\lambda_2}{\mu_3}$.

The parameters of the environments we evaluated on are:

1. High load
   - $\rho = 0.99$ and using parameters exactly as in Figure 9.
2. Very high load #1
   - $\rho = 0.99$ and using parameters exactly as in Figure 9.



(a) Single-server allocation (b) 2-Model Network (c) Traffic control

Figure 9. Left: Server-allocation. Image taken from Liu et al. (2019). Center: 2 model network. Image taken from Dai & Gluzman (2022). Right: An example intersection of the traffic control environment. Image taken from Alegre (2019).

3. Very high load #2
   - $\rho = 0.95$
   - Arrival rates: $\lambda_1 = 0.9$, $\lambda_2 = 0.8$
   - Service rates: $\mu_1 = 1, \mu_2 = 0.9, \mu_3 = 0.8$

Total Queue Length as Optimality Objective . While the lessons in our paper are generally applicable to RL, our work is grounded in queuing theory. As such, we are specifically interested in learning control policies that minimize and bound the *system latency*. Therefore, according to Little's Law (Leon-Garcia, 2008), we seek to minimize the true optimality cost function, the total queue length, $c(s, a, s') = \|s'\|_1$, where states s and s' are vectors consisting of the current and next queue lengths of each queue in the system and a is the action taken.

Traffic control In this environment, a traffic controller must select from a set of phases (shown in green in Figure 9), a set of non-conflicting lanes, to allow cars to move. At each time-step, new cars arrive in each lane at different rates, which determines the traffic congestion level. In our experiments, we considered medium to very high levels of traffic congestion. The state is the number of cars waiting in each lane along with indicator flags for which lanes have a green and yellow light. The action space is the number of phases. The state space is 21 dimensions and the action space is 4. The goal is to minimize the total waiting time of all the cars. To model a real-life traffic situation, the SUMO simulator places a cap of ≈ 100 on each lane. We use the SUMO simulator implementation (Behrisch et al., 2011; Alegre, 2019).

For exact traffic demands used in the experiments, see the `sumo/nets/big-intersection/generator.py` file in the attached code.

C.4. Additional Empirical Setup Details

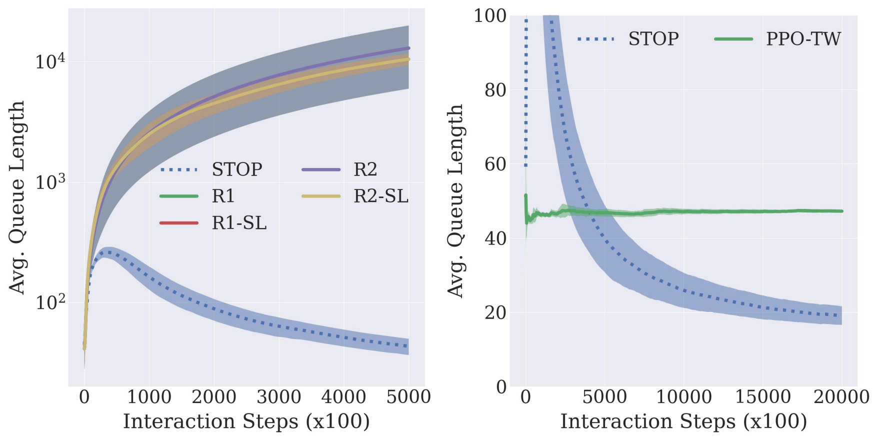
PPO Training We train average-reward PPO (Zhang & Ross, 2021; Dai & Gluzman, 2022) using the default hyperparameters (network architecture, learning rate, mini batches, epochs over the dataset etc.) in the cleanRL code base (Huang et al., 2022). For all algorithms and variations, we set the interval between policy updates during the interaction (rollout buffer length) to be 200 . The agent starts in a randomly initialized state, takes a number of actions until it fills up its rollout buffer and then makes policy updates using this buffer, and the process repeats. We normalize the advantages in the rollout buffer by dividing each by the standard deviation computed over the buffer. As suggested by Dohare et al. (2023), we set Adam's beta values to be $\beta_1 = \beta_2 = 0.9$.

C.5. Hardware For Experiments

For all experiments, we used the following compute infrastructure:

- Distributed cluster on HTCondor framework

- Intel(R) Xeon(R) CPU E5-2470 0 @ 2.30GHz
- RAM: 5GB
- Disk space: 5GB



(a) Other bounded formulations of true cost (b) Training wheels comparison

Figure 10. Average queue length on the 2-queue problem w/ faulty connections over time by (a) different RL agents optimizing different cost formulations vs. our STOP agent and (b) PPO-TW. Performance metrics are computed over 5 trials with 95% confidence intervals. Lower is better.

D. Transforming Optimality Cost Function Leads to Instability

In Figure 10(a) we also include the performance of RL agents optimizing other transformations of the true optimality cost function $c(s_{t+1})$. We consider the transformed cost: 1) $c'(s_t, a_t, s_{t+1}) := -\exp(-||s_t + a_t||_2^2)$ and 2) $c'(s_t, a_t, s_{t+1}) := -\exp(-||s_{t+1}||_2^2)$. The former is R_1 and latter is R_2 in the plot. We show performance when these agents do not use any state transformation and when they use the symloge transformation $(-SL)$. We find that the agents still unstable, thus providing further evidence that simply transforming the true cost function in this class of problems is insufficient to yield good performance.

E. PPO With Training Wheels

In this section, we provide preliminary evidence that equipping PPO with training wheels i.e. a stable policy may perform worse than STOP.

In this experiment, we evaluate PPO with training wheels (PPO-TW). The PPO-TW setup closely models that of (Mao et al., 2019) where we equip an on-policy policy gradient algorithm (PPO) with a stable policy. In our case, the stable policy is MAXWEIGHT (Stolyar, 2004; Tassiulas & Ephremides, 1990). MAXWEIGHT is deployed if the maximum queue length of the system exceeds 100, at which point MAXWEIGHT is used until it drives the system's maximum queue lengths to less than 50. Once it has done that, the PPO policy is deployed. Note that 1) MAXWEIGHT relies on knowing information of the transition dynamics, which can be limiting. STOP, on the other hand, does not assume access to such knowledge and 2) PPO-TW optimizes the true optimality cost (average queue length).

From Figure 10(b), we find that while STOP performs poorly during initial phases of learning, it is able to significantly outperform PPO-TW later on. STOP is able to learn the stabilizing and optimal actions from a destabilizing, random policy. In the case of PPO-TW, however, since the initial RL policy is poor (random) it causes the agent to diverge, which violates the safety condition often, which results in frequent deployment of the stable policy. However, this off-policy data cannot be used to update the PPO policy. Thus, the PPO policy continues to remain poor since it has inadequate data to train on, which causes the agent to diverge until it violates the safety condition, at which point the stable policy is deployed again. As noted by (Mao et al., 2019), the off-policy data generated by the stable policy cannot be used to train the PPO policy. As we have noted in our future work as well, an interesting further direction will be to apply our ideas to off-policy algorithms.