On Continual Learning in Multiclass Queueing Problems

by

Lucas Poon

A thesis submitted in partial fulfillment of

the requirements for the degree of

Bachelor of Science

(Computer Science)

at the

University of Wisconsin – Madison

2024

Advisors

Yudong Chen

Josiah Hanna

Qiaomin Xie

Brahma S. Pavse

Abstract

An awesome study of important things is presented. I further describe my project here, but I will not exceed 350 words, for that is strictly forbidden for the abstract of this document. If you're submitting online, you can even put symbols in your abstract and title, but you'll have to find out the HTML character codes for the various symbols.

Acknowledgements

This is where any acknowledgements would go.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reinforcement Learning has made a lot of progress in recent year, where most of such progress lies in the common learning paradigm. In the typical learning paradigm, the objective is to develop an agent that solve a problem. However, many real-life problems often presents itself as a continual learning problem, in which a learning agent must continuously learn and adapt its decision making to a never-ending stream of data.

In this work, we investigate continual reinforcement learning with the context of multiclass queueing problems, where the environment is stochastic and the state space is unbounded. In these problems, the agent must learn to make optimal sequential decisions such that it minimizes the queue lengths. Furthermore, the agent must learn to do so efficiently, as queue lengths can grow indefinitely. Pavse et al. (2024) proposes promising strategies, particularly Lyapunov-based cost shaping and state-transformation, to aid the agent in learning. Lyapunov-based cost shaping is closely tied to potential-based shaping proposed in Ng et al. (1999), but it is applied in the context of average reward and unbounded state space settings.

Despite promising results, Pavse et al. (2024) finds that finding a good Lyapunov function for cost shaping is often difficult in practice. Hence our first key question is: Can we design a Lyapunov function for optimal credit assignment in average reward reinforcement learning?. Another observation in Pavse et al. (2024) is degradation in performance in the long run. Thus our second key question is: What are the underlying causes of policy collapse in continual learning, and what strategies can mitigate it or prevent it?

In this paper, we provide the following insights towards these questions in this work:

1. Ng et al. (1999) notes that the ideal potential function is the optimal value function. However, we empirically show that unlike the potential-based shaping, a Lyapunov function that is close to the optimal value function may not be the optimal credit assignment for average reward reinforcement learning.

2. We show that large gradient spikes is a major cause of policy collapse and we give the intuition that Adam's hyper-parameter β_2 has a smoothing effect that makes it a possible mitigation strategy towards policy collapse in continual learning problems. We also show the trade-offs for this smoothing, where the agent's ability to learn and adapt is decreased, which may not be ideal for continual learning problems.

We structure the rest of the paper as follows. Section 2 describes the current literature in potential based shaping and continual reinforcement learning. Section 3 introduces the prerequisite backgrounds to understand the rest of the paper. Section 4 details our observations on Lyapunov-based cost shaping as well as our findings on policy

collapse, followed by our intuitions on mitigation strategies. Finally, Section 5 outlines a summary on our results and discusses future work.

Chapter 2

Related Work

This section depicts the current literature relevant to the paper. Section 2.2 describes previous work on potential based shaping. Section 2.1 discusses works on defining reinforcement learning for continuing problems and Section 2.3 describes previous works on understanding and mitigating policy collapse and plasticity loss in continual reinforcement learning settings.

2.1 Continual Reinforcement Learning

Our work focuses on continual reinforcement learning, where we address problems where the agent environment interaction never terminates and performance is evaluated online (Sutton and Barto, 2018). This problem formulation is also often described as lifelong learning (Brunskill and Li, 2014), single-life reinforcement learning (Chen et al., 2022) and is closely related to transfer learning (Taylor and Stone, 2009).

2.2 Potential Based Shaping

Introduced by Ng et al. (1999), potential based shaping is a framework that guides the learning algorithm towards learning a good or even optimal policy faster. This is especially important in continual reinforcement learning problems since we lack samples and data. Similar to potential based shaping, Pavse et al. (2024) proposes Lyapunov-based cost shaping, applying in the context of continual learning as a means to encourage stability and optimality for the agent's learning.

Ng et al. (1999) presents theoretically that having the potential function to be equal to the optimal value function guarantees consistency with the optimal policy. Our work looks to see if this theoretical result still holds under the continual learning problem when we choose the Lyapunov function to be the optimal value function such that $\ell(s) = V^*(s)$.

2.3 Policy Collapse and Plasticity Loss

One of the major challenges in continual reinforcement learning is policy collapse (Dohare et al., 2023) and plasticity loss (Dohare et al., 2024; Lyle et al., 2023). Dohare et al. (2023) defines policy collapse to be the phenomenon where the learn policy degrades as the agent continues interacting in the environment. Plasticity loss is the ability of a neural network to learn and adapt to new information. These two issues both cause performance degradation in continual reinforcement learning problems, which makes finding the root causes to these issues difficult.

Our work finds cases where only policy collapse is observed and we further provide intuition to what causes the policy collapse phenomenon. We also discuss the

benefits and consequences of existing mitigation strategies such as tuning Adam's β_2 hyper-parameter, as well as L_2 regularization (Lyle et al., 2023; Dohare et al., 2023, 2024).

Chapter 3

Preliminaries

This section describes the relevant background necessary to understand the later sections. Specifically, Section 3.1 introduces reinforcement learning and the formulation for continual learning tasks, and Section 3.2 defines average reward as our reinforcement learning objective. Section 3.3 describes the problem settings considered in this work, Section 3.4 briefly discusses initial approaches to the problem in Pavse et al. (2024) and finally Section 3.5 describes the Adam optimizer algorithm.

3.1 Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning that emphasizes on learning from rewards gained from trial and error. The learner receive rewards for taking actions within an environment and over time learns a policy to take actions that maximize its rewards.

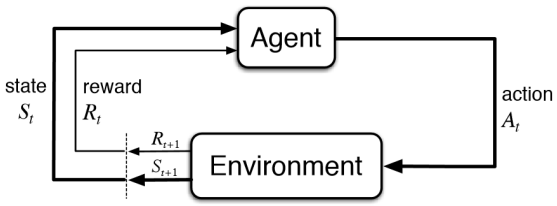


Figure 3.1: Diagram of the agent environment interaction loop

The agent at state S_t takes an action A_t and the environment returns a reward R_t and the agent transitions to the new state S_{t+1}. This cycle repeats, forming the agent environment interaction loop.

We model agents acting in continual learning environments as an infinite-horizon Markov decision process (MDP) (Puterman, 1994). An infinite-horizon MDP is denoted by the tuple $M := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, c, d_1 \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^d$ is the state space of the queueing system, $\mathcal{A}$ is the action space consisting of all the possible actions at each state, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics for each action taken by the agent at each state, $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the optimality cost function received by the agent for moving from a state to another given the action the agent has taken, and $d_1 \in \Delta(\mathcal{S})$ is the initial state distribution.

Following the control-theoretic convention, we consider the cost to be the negation of rewards, which denotes the L_1 distance from the current state to the zero vector target state. Since our queue lengths cannot be negative, without loss of generality, we assume that the cost is non-negative and we restrict the cost function to only depend on the next state s_{t+1}.

3.2 Average Reward

In the continual learning task formulation, the agent acts accordingly to the policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, which generates a single and infinite stream of experiences: $(s_1, c_1, a_1, s_2, ..., s_t, c_t, a_t, ...)$, where $s_1 \sim d_1$, $s_{t+1} \sim P(\cdot|s_t, a_t)$, $c_t = c(s_t, a_t, s_{t+1})$ and $a \sim \pi(\cdot|s_t)$. To optimize for long-term behavior, we consider the long-run average-cost objective (Naik et al., 2019, 2021):

$$J^O(\pi) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi[c_t]. \tag{3.1}$$

where the optimal policy π is the policy that minimizes (3.1). We then define the following *differential value functions* for the average-cost setting, equivalent to the standard RL value functions:

$$V^\pi(s) := \lim_{T\to\infty} \mathbb{E}_\pi[\sum_{t=1}^{T} (c_t - J^O(\pi))|s_1 = s] \tag{3.2}$$

$$Q^\pi(s, a) := \lim_{T\to\infty} \mathbb{E}_\pi[\sum_{t=1}^{T} (c_t - J^O(\pi))|s_1 = s, a_1 = a] \tag{3.3}$$

$$A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s) \tag{3.4}$$

where (3.2) is the *differential* state-value function, (3.3) is the *differential* action-value function and (3.4) is the *differential* advantage function. The advantage function describes how much better or worse it is to action a in state s with respects to the long-term outcome compared to randomly sampling according to $\pi(\cdot|s)$. Furthermore, in this work, we assume that the MDP is communicating (Bertsekas, 2015), that is, for any two states $s, s' \in \mathcal{S}$, s' is accessible from s and s is also accessible from s' in a finite number of steps with a positive probability. From this assumption, the optimal average cost value is independent of the starting state d_1.

3.3 Control of Multiclass Queueing Networks

We formulate a multiclass queueing network to have Q different queue classes and S server stations. We denote $Q_i(t)$ to be the queue length of the i-th queue class at time t. Each Q_i corresponds to a server S_j, where S_j is the j-th server. For each Q_i, jobs arrive with arrival probability λ_i and have a service rate μ_i. At each timestep, each server can serve a single job from one of the queue classes that correspond to it. When a job is successfully served, it can leave the system or go to another server for further processing.

3.3.1 2-Queue Network

The 2-queue network depicted in Figure 3.2, which is taken and modified from Figure 4 in Liu et al. (2019), consists of two queue classes and one server station. At each timestep, each queue Q_i increases by 1 with probability λ_i and the server must select which of the two queues to serve.

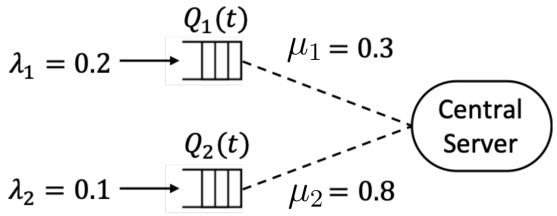


Figure 3.2: 2-Queue Network

In this setting, we have an extra parameter for each queue, c_i, which denotes the connection probability of the server to the queue. For a job to be serviced in this setting, the server must be able to: 1) connect to the queue with probability c_i, 2) serve it with probability μ_i. When a job is successfully served, the job leaves the

system and $Q_i(t)$ is reduced by 1. In this work, we consider the following 2-queue settings:

Load regime	λ_1	λ_2	μ_1	μ_2	c_1	c_2
Fully Connected (Medium Load)	0.2	0.2	0.3	0.8	1	1
Faulty Connections (High Load)	0.2	0.1	0.3	0.8	0.95	0.5
Faulty Connections (Very High Load)	0.2	0.1	0.3	0.8	0.7	0.5

Table 3.1: Load parameters for 2-queue network for Figure 3.2

3.3.2 Criss-Cross Network

The criss-cross network depicted in Figure 3.3, which is taken from Figure 1 in (Dai and Gluzman, 2022), consists of three queue class and two server stations. At each timestep, Q_1 and Q_3 increase by 1 with probability λ_1 and λ_2 respectively and Q_2 increases by 1 if the server serviced a job from Q_1. Furthermore, each server can choose to serve one of their corresponding queues, or idle.

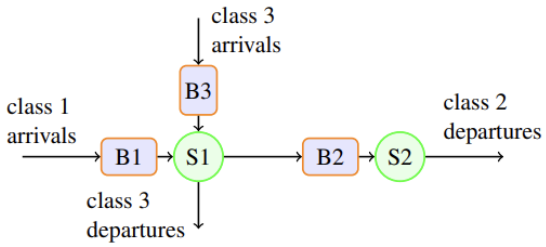


Figure 3.3: Criss-Cross Network

In this setting, jobs from the class 3 queue leaves the system after being serviced in server 1. Jobs from the class 1 queue becomes a class 2 job when serviced in server 1 and leaves the system only when server 2 serves that job. In this work, we consider the following criss-cross settings:

Load regime	λ_1	λ_3	μ_1	μ_2	μ_3
Imbalanced Low	0.3	0.3	2	1.5	2
Imbalanced Medium	0.6	0.6	2	1.5	2
Imbalanced High	0.9	0.9	2	1.5	2

Table 3.2: Load parameters for criss-cross network for Figure 3.3

3.4 STability and OPtimality (STOP)

Introduced by Pavse et al. (2024), STOP is an approach to create robust agent for continual learning problems with an emphasis on encouraging stability and optimality. This approach consists of 1) Lyapunov-based cost shaping and 2) state transformation. For the purpose of this work, we will only consider the sigmoid state transformation and build on the Lyapunov-based cost shaping and its performance on longer horizons in continual learning. Here we briefly define the new objective after Lyapunov-based cost shaping:

$$J^S(\pi) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi[\ell(s_{t+1}) - \ell(s_t) + c(s_t, a_t, s_{t+1})] \tag{3.5}$$

where ℓ is the Lyapunov function that intuitively describes the energy of a system and c is the true optimality cost. For further information, refer to Pavse et al. (2024) for the full definitions of cost shaping and state transformations.

3.5 Adam Algorithm

Adam (Kingma and Ba, 2014) is an optimization algorithm that uses only the first-order gradients to update the model parameters θ_t for iteration t. The update rule

of Adam given the gradient g_t at iteration t is given as following:

$$m_t = \frac{\beta_1}{1-\beta_1^t} \cdot m_{t-1} + \frac{1-\beta_1}{1-\beta_1^t} \cdot g_t \tag{3.6}$$

$$v_t = \frac{\beta_2}{1-\beta_2^t} \cdot v_{t-1} + \frac{1-\beta_2}{1-\beta_2^t} \cdot g_t^2 \tag{3.7}$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{3.8}$$

where α is the learning rate, ϵ is a stability hyper-parameter and β_1, β_2 are hyper-parameters to control the exponential decay rates of the moving averages. More intuitively, we compute the bias-corrected first moment estimate and the bias-corrected second raw estimate in 3.6 and 3.7 respectively. Then, we update the parameters in 3.8. In practice, the default betas are $\beta_1 = 0.9, \beta_2 = 0.999$ are usually used for both supervised learning and reinforcement learning (Paszke et al., 2019).

Chapter 4

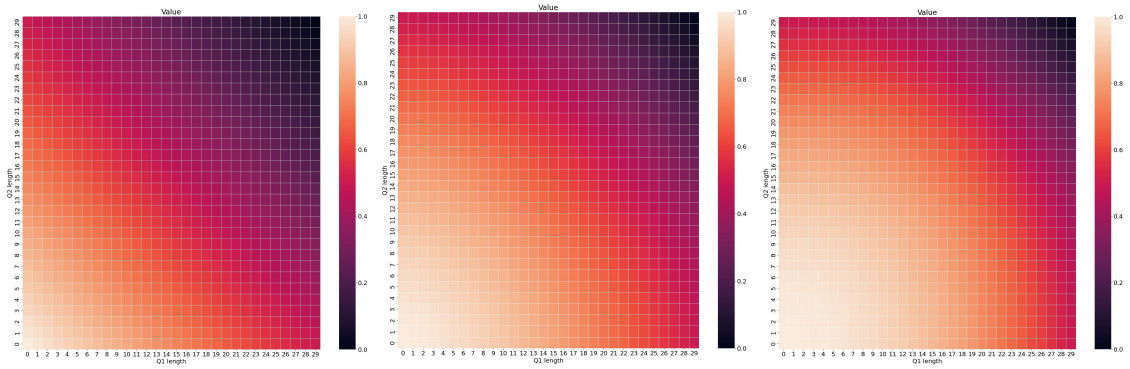
Results

In this section, we present out results in two parts. Section 4.1 responds to our first key question: Can we design a Lyapunov function for optimal credit assignment in average reward reinforcement learning? and Section 4.2 we explore our second key question: What are the underlying causes of policy collapse in continual learning, and what strategies can mitigate it or prevent it? In addition, we also provide some intuition on the consequences of existing mitigation strategies.

4.1 Exploration on Lyapunov Function

The first part of our results, we build on the initial insights from Pavse et al. (2024) and focus on choosing a suitable Lyapunov function $\ell(s)$. As discussed in Ng et al. (1999), the optimal value function serves as the ideal Lyapunov function where $\ell(s) = V^*(s)$. However, Pavse et al., notes that we often do not know what the optimal value function is and a more appropriate approach is to find a Lyapunov function that approximates the optimal value function such that $\ell(s) \approx V^*(s)$.

In this section, we explore various Lyapunov function choices of the form $\sum_i^n Q_i^{(\beta+1)}$ for $\beta > 0$, inspired by variants of MaxWeight (Stolyar, 2004). For notational convenience we denote $p = \beta + 1$ to be the Lyapunov power. To gain a better insight on which Lyapunov function is best, we generated heatmaps of the optimal value function and compared it with different Lyapunov function choices. In the following results, we consider $p = 1.0, 1.5, 2.0, 2.5, 3.0$ and plot their normalized heatmaps:



(a) $p = 1.0$ (b) $p = 1.5$ (c) $p = 2.0$

(d) $p = 2.5$ (e) $p = 3.0$

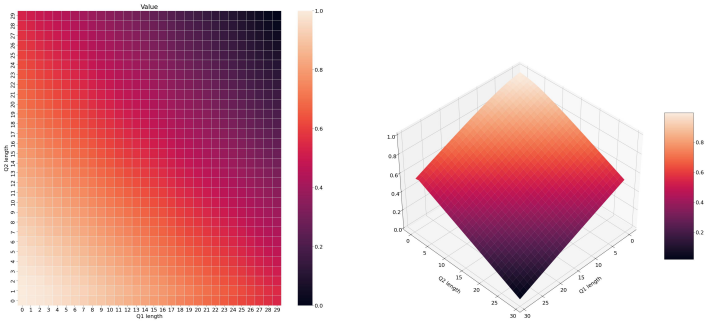
Figure 4.1: Normalized Lyapunov value functions for $p = 1.0, 1.5, 2.0, 2.5, 3.0$

4.1.1 Comparison of Lyapunov and Optimal Value Function

Ideally, the best Lyapunov function is the function with the same shape as the optimal value function. Hence, we compare the Lyapunov function heatmaps with the

optimal value function for the different 2-queue network settings. However, due to the unbounded nature of the queue lengths, our state space is unbounded, hence we cannot honestly compute the optimal value function. As an alternative, we compute an approximation of the optimal value function with dynamic programming on a bounded state space. Specifically, we restrict the state space such that the maximum queue length is 120, that is, when there is an arrival at queue i at time t and $Q_i(t-1) = 120$, our queue length stays at 120, i.e., $Q_i(t) = 120$.

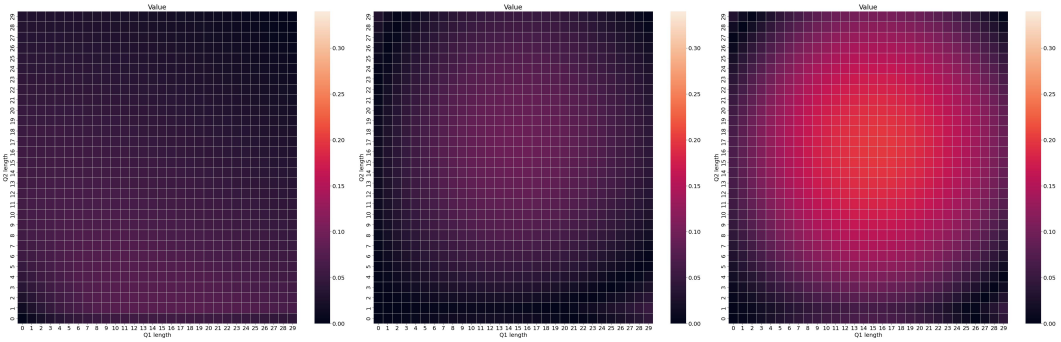
Fully Connected (Medium Load)



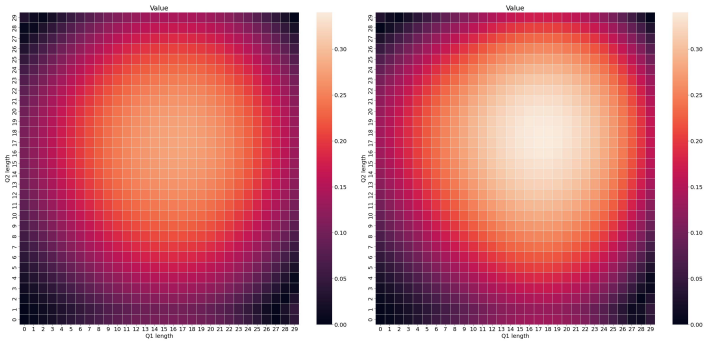
(a) Normalized optimal value function heatmap in 2D

(b) Normalized optimal value function heatmap in 3D

Figure 4.2: Normalized optimal value function heatmap for states $[0, 30]$ for fully connected (medium load) setting in the 2-queue network



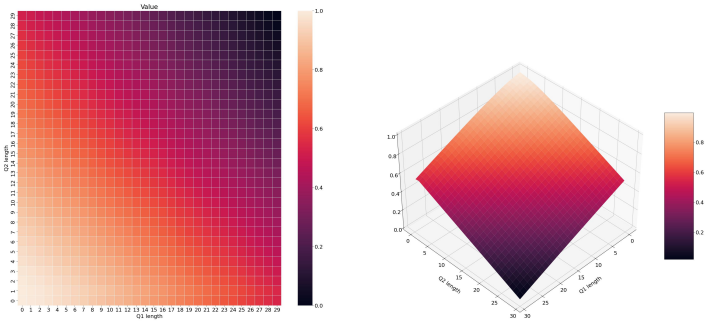
(a) $p = 1.0$ (b) $p = 1.5$ (c) $p = 2.0$



(d) $p = 2.5$ (e) $p = 3.0$

Figure 4.3: Normalized absolute difference between Lyapunov and optimal value functions for $p = 1.0, 1.5, 2.0, 2.5, 3.0$

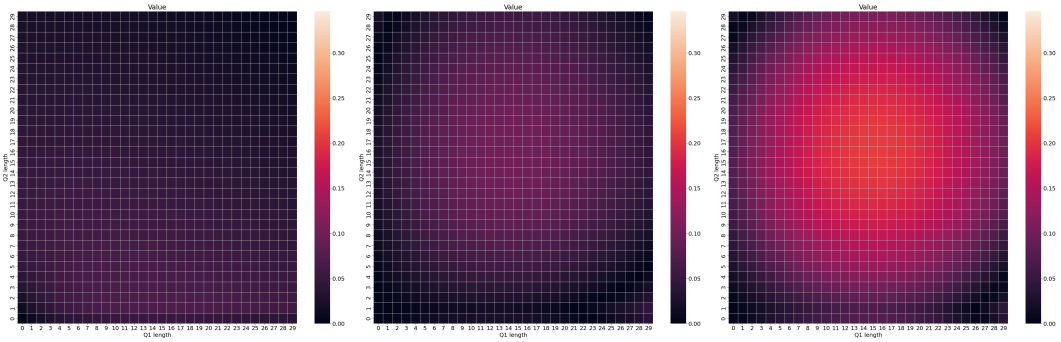
Faulty Connections (High Load)



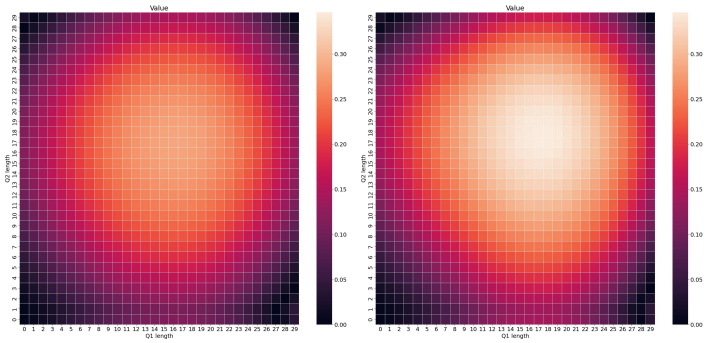
(a) Normalized optimal value function heatmap in 2D

(b) Normalized optimal value function heatmap in 3D

Figure 4.4: Normalized optimal value function heatmap for states $[0, 30]$ for faulty connections connected (very high load) setting in the 2-queue network



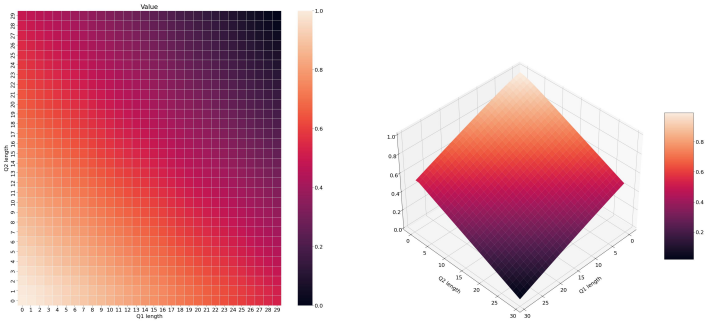
(a) $p = 1.0$ (b) $p = 1.5$ (c) $p = 2.0$



(d) $p = 2.5$ (e) $p = 3.0$

Figure 4.5: Normalized absolute difference between Lyapunov and optimal value functions for $p = 1.0, 1.5, 2.0, 2.5, 3.0$

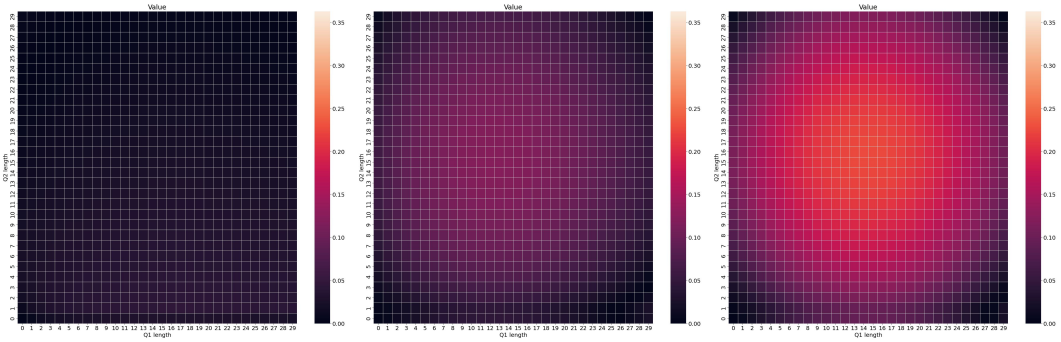
Faulty Connections (Very High Load)



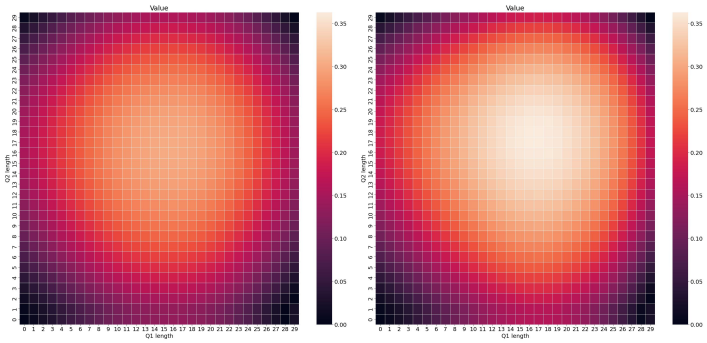
(a) Normalized optimal value function heatmap in 2D

(b) Normalized optimal value function heatmap in 3D

Figure 4.6: Normalized optimal value function heatmap for states $[0, 30]$ for faulty connections connected (very high load) setting in the 2-queue network



(a) $p = 1.0$ (b) $p = 1.5$ (c) $p = 2.0$



(d) $p = 2.5$ (e) $p = 3.0$

Figure 4.7: Normalized absolute difference between Lyapunov and optimal value functions for $p = 2.0, 1.5, 2.0, 2.5, 3.0$

4.1.2 Summary of Heatmap Observations

The heatmaps presented in Figures 4.2, 4.4 and 4.6 illustrate the behavior of the optimal value function across the queue lengths. Note that even though we computed dynamic programming with maximum queue length of 120, the plots focuses on the state space where $Q_1, Q_2 \in [0, 30]$. The color intensity represents the value of each state, with lower value states being darker.

We observe a near linear decrease in value as the average queue length increases. As we approach states where either Q_1 or Q_2 tends to 0, the value remains high at $[0, 0]$ (the optimal state), but the improvement in value becomes less significant as the queue lengths decrease further.

To further validate the near-linear shape of the optimal value functions, we applied to model the cross-term between Q_1 and Q_2 across different state bounds. The results for each are summarized below:

State bound	Q_1^2	Q_2^2	Q_1	Q_2	Coef. Ratio $\left\lvert\frac{Q_1^2}{Q_1}\right\rvert$	Coef. Ratio $\left\lvert\frac{Q_2^2}{Q_2}\right\rvert$
[0, 30]	-0.09324	-0.09295	-46.06988	-43.32195	0.00154	0.00301
[0, 45]	-0.05226	-0.05226	-47.04658	-43.32195	0.00094	0.00212
[0, 60]	0.00133	0.00321	-48.82095	-47.11132	0.00002	0.00007

Table 4.1: Table of cross-term coefficients on different state bounds on fully connected (Medium Load) setting rounded to 5 d.p.

State bound	Q_1^2	Q_2^2	Q_1	Q_2	Coef. Ratio $\left\lvert\frac{Q_1^2}{Q_1}\right\rvert$	Coef. Ratio $\left\lvert\frac{Q_2^2}{Q_2}\right\rvert$
[0, 30]	0.05226	-0.01171	-51.53214	-46.32623	0.00101	0.00025
[0, 45]	0.01442	-0.03698	-50.69300	-45.85704	0.00028	0.00081
[0, 60]	-0.00047	0.00183	-49.73413	-47.90776	0.00001	0.00004

Table 4.2: Table of cross-term coefficients on different state bounds on faulty connections (High Load) setting rounded to 5 d.p.

State bound	Q_1^2	Q_2^2	Q_1	Q_2	Coef. Ratio $\left\lvert\frac{Q_1^2}{Q_1}\right\rvert$	Coef. Ratio $\left\lvert\frac{Q_2^2}{Q_2}\right\rvert$
[0, 30]	0.72893	0.61966	-75.78481	-68.25969	0.00962	0.00908
[0, 45]	0.31801	0.23977	-66.94255	-60.33981	0.00475	0.00397
[0, 60]	-0.00906	-0.00565	-54.05267	-51.97891	0.00017	0.00011

Table 4.3: Table of cross-term coefficients on different state bounds on faulty connections (Very High Load) setting rounded to 5 d.p.

Tables 4.1, 4.2 and 4.3 depict the cross-term coefficients on different state bounds for different settings in the 2-queue network. Additionally, we also present the absolute coefficient ratios between Q_1^2 and Q_1, as well as Q_2^2 and Q_2. These ratios highlight the relative contribution of the quadratic terms compared to their respective linear terms. We observe that the coefficient ratios generally decreases as the state bounds increase, suggesting that the optimal value function becomes increasingly linear as the average queue length grows.

Figures 4.3, 4.5 and 4.7 presents the absolute difference between each Lyapunov function and the optimal value function for each setting. A darker state indicates a smaller difference. Note that the ideal Lyapunov function would have an absolute difference of 0, resulting in a uniform black heatmap. We observe that for all three settings, the linear Lyapunov ($p = 1.0$) has the least difference compared to other

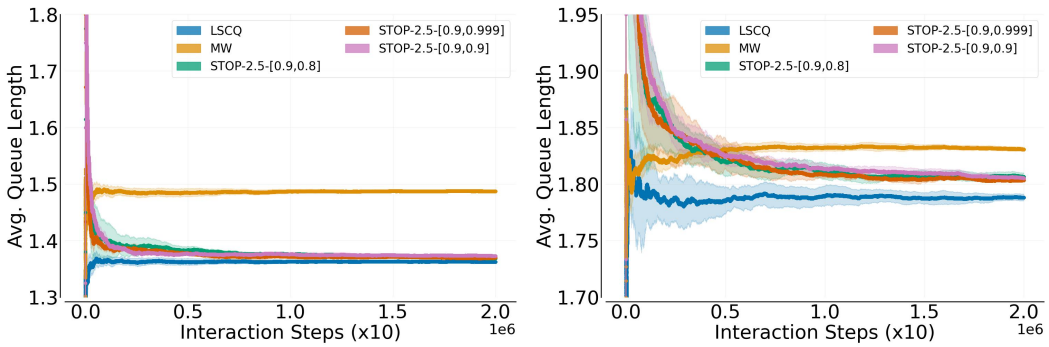
Lyapunov function choices. Moreover, the magnitude of the difference increases as we increase our Lyapunov power p. However, this contradicts with the results presented in Figure 4 in Pavse et al. (2024), where the linear Lyapunov diverges in the faulty connections (Very High Load) setting. Moreover, STOP with $p = 3.5$ performs better than STOP with $p = 2.0$ in all cases even though the heatmaps show that $p = 2.5$ has a bigger absolute difference to the optimal value function than $p = 2.0$.

4.2 Adam and Policy Collapse

In the second part of our results, we build on the work of Pavse et al. (2024). In the paper, STOP is only trained for 2 million time-steps to provide an initial insight to the convergence of the algorithm in a continual learning setting. In our work, we further train STOP on a much longer horizon to observe the performance of the algorithm in longer horizons.

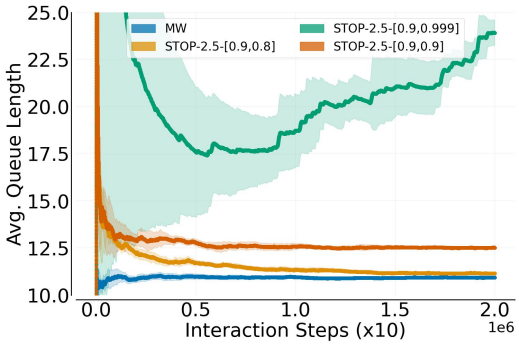
4.2.1 Adam Beta Exploration

Previous works have shown that Adam can become problematic in continual learning settings, leading to issues such as policy collapse (Dohare et al., 2023) and plasticity loss (Lyle et al., 2023; Dohare et al., 2024). Lyle et al. and Dohare et al. suggests using a lower Adam β_2 value as a method to mitigate policy collapse and plasticity loss. Hence, we train STOP in the 2-queue network on 20 million interaction time-steps with $\ell(s) = \|s\|_2^{2.5}$ and Adam $\beta_1 = 0.9$, $\beta_2 = 0.999, 0.9$ or 0.8. Note that the standard beta values for Adam are $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (Paszke et al., 2019).



(a) Fully connected with medium load

(b) Faulty connections with high load



(c) Faulty connections with very high load

Figure 4.8: STOP trained on 20 million steps with varying Adam β_2 values on 2-queue network

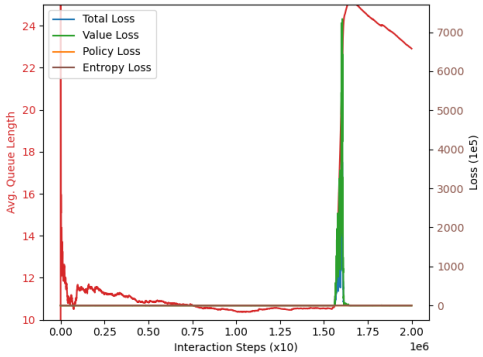
The algorithms in Figure 4.8 are denoted by STOP-p-[β_1, β_2], where p is the power of the Lyapunov function, β_1, β_2 are the beta values for the Adam optimizer. We also show the performance of MaxWeight (MW). It is important to note that unlike MW, STOP does not know the transition dynamics of the MDP.

Our results show that STOP performs consistently well in two settings: fully connected (medium load) and faulty connections (high load), as shown in Figure 4.8a and Figure 4.8b, respectively. This is true regardless of the Adam beta hyper-parameters. However, in the faulty connections (very high load) setting (Figure 4.8c), we observe that with Adam betas set to $\beta_1 = 0.9, \beta_2 = 0.999$, the policy fails to learn effectively

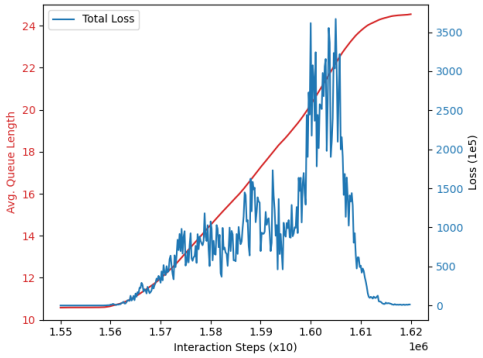
and even collapses similar to observations in Dohare et al. (2023). In contrast, when Adam betas are set to $\beta_1 = 0.9, \beta_2 = 0.8$, the algorithm performs exceptionally well, converging to the performance of MW and exceeds performance results previously reported in Pavse et al. (2024).

4.2.2 Adam Beta Tuning is not enough

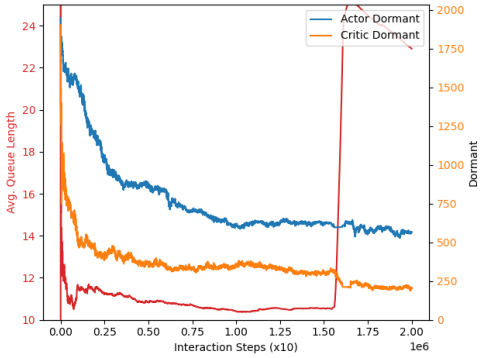
Initial results in Figure 4.8 show promising results, however on deeper inspection of the 20 independent trials, we observe policy collapse still occurs when the Adam $\beta_2 = 0.8$. In the following results, we show the two such cases we try to find the cause.



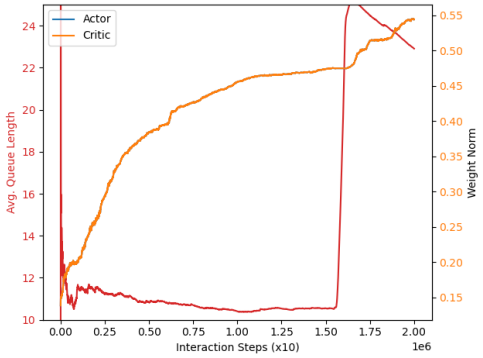
(a) Avg. Queue Length vs. Loss



(b) Avg. Queue Length vs. Loss (Zoomed In)

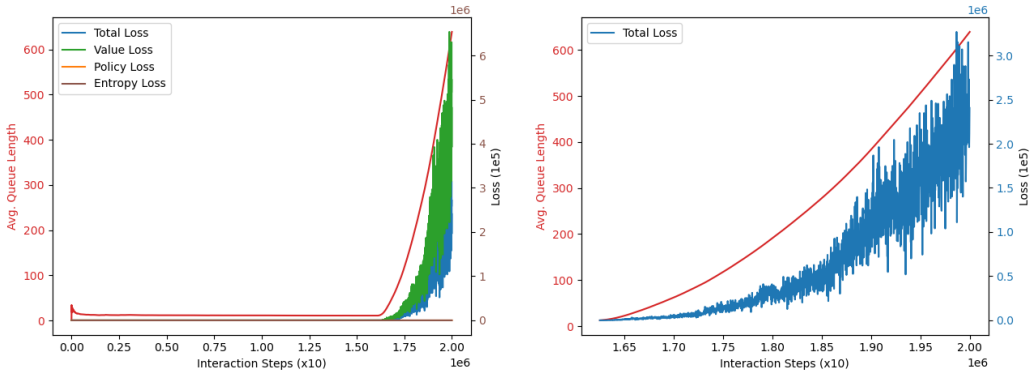


(c) Avg. Queue Length vs. No. of Dormant Units



(d) Avg. Queue Length vs. Weight Magnitude

Figure 4.9: Detailed graphs for STOP-2.5 with Adam $\beta_1 = 0.9, \beta_2 = 0.8$ for seed 175587



(a) Avg. Queue Length vs. Loss

(b) Avg. Queue Length vs. Loss (Zoomed In)

Figure 4.10: Detailed graphs for STOP-2.5 with Adam $\beta_1 = 0.9, \beta_2 = 0.8$ for seed 496018

From Figure 4.9c we observe that the cause of this policy collapse is not related to the increase in dormant units or the decrease in weight magnitude as observed in (Dohare et al., 2024). However, we did observe loss spikes as shown in 4.9 a and 4.9 b, where there was a sudden decrease in performance whenever there was a spike in losses. This is similar to how Dohare et al. (2023) had observed a sudden nonzero gradient which led to a spike in updates. In the case of 4.9, we see that after the sudden spike in performance loss, the agent is able to recover itself as the average queue length decreases. From Figure 4.10, we find similar results as in 4.9, however, this time the agent is unable to recover.

4.2.3 Intuition on Adam Betas

Dohare et al. (2023) provides an intuition that in an idealized scenario where the gradients for all updates are zero except at a large time t, Adam with default parameters results in a large total change at a later timestep, such as $t + 40$. We apply a similar reasoning to understand and extend this concept, particularly to gain insight

into what the β_2 hyper-parameter does.

We fix $\beta_1 = 0.9$, $\epsilon = 1e-8$ and some learning rate α. Now suppose we have a long sequence of gradients $(g_0, ..., g_t, g_{t+1}, ...)$, where $g_i = 0$ if $i \neq t$ and $g_t > 0$ for $i \geq 0$ and some large t. Then at time t, we compute our Adam update:

$$m_t = \frac{0.9}{1 - 0.9^t} \cdot m_{t-1} + \frac{0.1}{1 - 0.9^t} \cdot g_t \approx 0.1 g_t \tag{4.1}$$

$$v_t = \frac{\beta_2}{1 - \beta_2^t} \cdot v_{t-1} + \frac{1 - \beta_2}{1 - \beta_2^t} \cdot g_t^2 \approx (1 - \beta_2) g_t^2 \tag{4.2}$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{0.1 g_t}{\sqrt{(1 - \beta_2) g_t^2} + \epsilon} \tag{4.3}$$

For Equations 4.1 and 4.2, we use the fact that $g_0, ..., g_{t-1}$ are all zeros and for a large t, $\beta_1^t = 0$ and $\beta_2^t = 0$ since $\beta_1, \beta_2 \in [0, 1)$. Then our update in Equation 4.3 is approximately

$$\alpha \cdot \frac{0.1}{\sqrt{1 - \beta_2}}$$

Subsequently, using the fact that $g_{t+1}, g_{t+2}, ...$ are all zeros, we can also compute the updates at times $t + 1, t + 2, ..., t + k$ for some $k \geq 0$:

$$\begin{aligned}
&\alpha \cdot \frac{0.9 \times 0.1}{\sqrt{\beta_2 (1 - \beta_2)}} && \text{at time } t + 1 \\
&\alpha \cdot \frac{0.9^2 \times 0.1}{\sqrt{\beta_2^2 (1 - \beta_2)}} && \text{at time } t + 2 \\
&\vdots \\
&\alpha \cdot \frac{0.9^k \times 0.1}{\sqrt{\beta_2^k (1 - \beta_2)}} && \text{at time } t + k
\end{aligned}$$

Then the total change at time $t + k$ is given by:

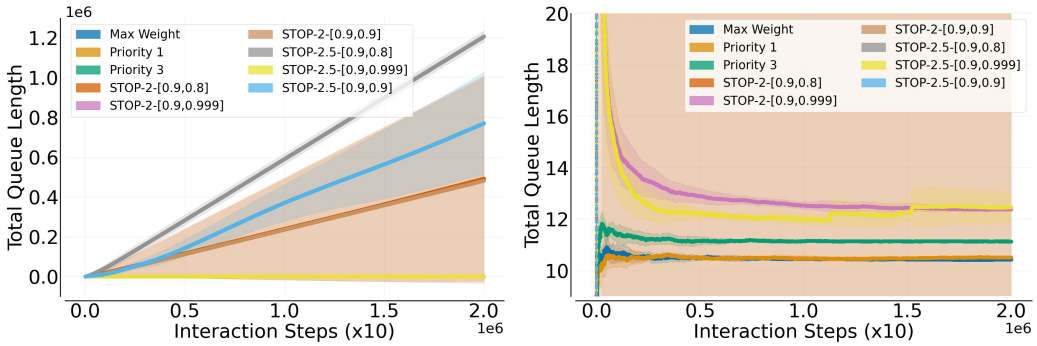
$$\sum_{n=0}^{k} \alpha \cdot \frac{0.9^n \cdot 0.1}{\sqrt{\beta_2^n \cdot (1 - \beta_2)}}$$

In the following results, we compute the total change at t+40 given $\beta_2 = 0, 8, 0.9, 0.999$.

β_2	t	$t+1$	$t+2$	$t+3$	...	Total change at $t+40$
0.8	$0.224 \cdot \alpha$	$0.225 \cdot \alpha$	$0.226 \cdot \alpha$	$0.228 \cdot \alpha$	...	$10.122 \cdot \alpha$
0.9	$0.316 \cdot \alpha$	$0.3 \cdot \alpha$	$0.285 \cdot \alpha$	$0.27 \cdot \alpha$	...	$5.413 \cdot \alpha$
0.999	$3.162 \cdot \alpha$	$2.847 \cdot \alpha$	$2.564 \cdot \alpha$	$2.309 \cdot \alpha$	...	$31.286 \cdot \alpha$

Table 4.4: Total change for varying β_2 values at $t+40$

Intuitively, the results of Table 4.4 show that as we set β_2 closer to 0.9, our agent becomes more resilient to gradient spikes as the smoothing effect is much stronger. However, over-smoothing leads to overgeneralization, as seen in our results in Figure 4.8c, where we were able to learn better with $\beta_2 = 0.8$. Another example of this can be seen when applying this to the imbalanced high setting for the criss-cross network in Figure 4.11, where we observe that the agent fails to learn and diverges unless we use the default hyper-parameter values for β_2.



(a) Total Queue Length (b) Total Queue Length (Zoomed In)

Figure 4.11: STOP trained on 20 million steps with varying Adam β_2 values on criss-cross network imbalanced high setting

4.2.4 Non-Stationary Transition Dynamics

In this section, we want to gain an initial intuition to how the hyper-parameter β_2 affects the agent's ability to adapt. We return to our 2-queue network, where we consider the environment to initially begin in the faulty connection (high load)

regime and transition to the faulty connection (Very High Load) regime after 10 million timesteps. Note that these two load regimes are very similar, where the only difference is that the very high load regime has a slightly lower connection probability c_1 for queue 1.

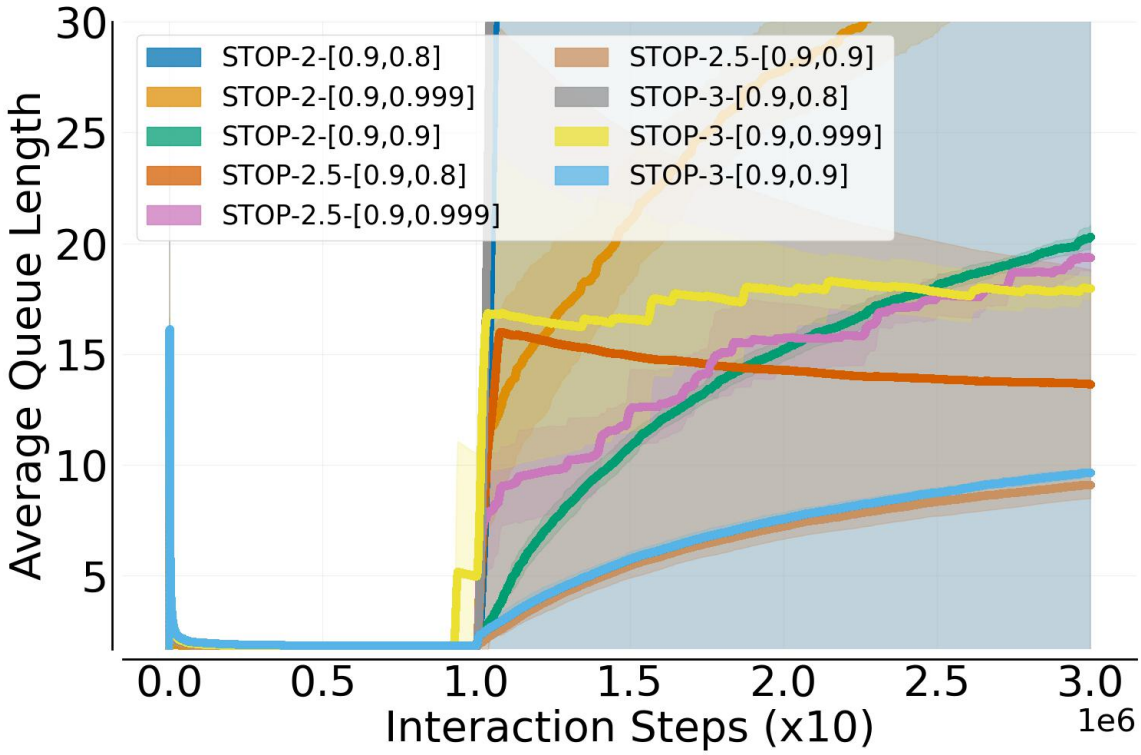


Figure 4.12: STOP trained on 30 million steps with varying Adam β_2 values on 2-queue network with non stationary transition dynamics

From Figure 4.12, we observe that when Adam's $\beta_2 = 0.9$, our agent adapts to the new environment very smoothly. With Lyapunov power $p = 2.5$ and Adam $\beta_2 = 0.8$, the agent also began to converge. However, for the rest, the behavior remained unstable even after 20 million timesteps after the change in environment. This result aligns well with our intuitions in the previous section, where we noted that setting Adam's β_2 close to 0.9 results in high smoothing, while values farther from 0.9 cause the agent to be sensitive to gradient spikes.

Chapter 5

Discussion and Further Work

In this section, we discuss our results from 4 to ultimately answer our initial questions and provide further directions. Section 5.1 discusses our first question: can we design a Lyapunov function for optimal credit assignment in average reward reinforcement learning?. Section 5.2 discusses what underlying causes of policy collapse in continual learning our results show, and also discusses the benefits and drawbacks of strategies that attempt to mitigate policy collapse? In this work, we gain intuition on two fundamental aspects of continual reinforcement learning: 1) selecting an appropriate Lyapunov function that guides the agent towards stability and optimality, and 2) understanding the role of the Adam optimizer β_2 hyper-parameter in policy collapse and plasticity loss.

5.1 Lyapunov Function

Our results showed that a Lyapunov function that is close to the optimal value function does not necessarily guarantee better optimal convergence.

5.2 Policy Collapse and Plasticity Loss

Our results show that there is no one-size-fits-all value for Adam's β_2 hyper-parameter that is best suited for continual learning. We observe that the β_2 value determines how much smoothing we apply to our updates. This intuitively creates a smoothing effect similar to applying L_2 regularization (Goodfellow et al., 2016), where we apply a penalty for large weights as suggested in Dohare et al. (2023; 2024). However, while these methods show better resilience to policy collapse from loss spikes, they introduce significant trade-offs. By dampening the large updates, we limit the agent's ability to learn optimally or in the worst case, diverge as shown in our results. Moreover, our results show that smoothing only reduces policy collapse, which may not be robust enough for continual learning problems where we need to learn forever.

Bibliography

Bertsekas, D. P. (2015). *Dynamic programming and optimal control*. Number volume 2. Athena Scientific, fourth edition, volume ii edition.

Brunskill, E. and Li, L. (2014). Pac-inspired option discovery in lifelong reinforcement learning. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 316–324, Bejing, China. PMLR.

Chen, A. S., Sharma, A., Levine, S., and Finn, C. (2022). You only live once: Single-life reinforcement learning.

Dai, J. G. and Gluzman, M. (2022). Queueing network controls via deep reinforcement learning. *Stochastic Systems*, 12(1):30–67.

Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., and Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774.

Dohare, S., Lan, Q., and Mahmood, A. R. (2023). Overcoming policy collapse in deep reinforcement learning. In *Sixteenth European Workshop on Reinforcement Learning*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

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

Lyle, C., Zheng, Z., Nikishin, E., Avila Pires, B., Pascanu, R., and Dabney, W. (2023). Understanding plasticity in neural networks. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 23190–23211. PMLR.

Naik, A., Shariff, R., Yasui, N., Yao, H., and Sutton, R. S. (2019). Discounted reinforcement learning is not an optimization problem.

Naik, A., Shariff, R., Yasui, N., Yao, H., and Sutton, R. S. (2021). Towards reinforcement learning in the continuing setting.

Ng, A., Harada, D., and Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.

Pavse, B. S., Zurek, M., Chen, Y., Xie, Q., and Hanna, J. P. (2024). Learning to stabilize online reinforcement learning in unbounded state spaces.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition.

Stolyar, A. L. (2004). MaxWeight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. *The Annals of Applied Probability*, 14(1):1 – 53.

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

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(56):1633–1685.