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

Abstract i

Acknowledgements ii

List of Figures **v**

List of Tables **vi**

1 Introduction **1**

2 Related Work **2**

2.1 Continual Reinforcement Learning 2

2.2 Policy Collapse and Plasticity Loss 2

3 Preliminaries **3**

3.1 Control of Multiclass Queueing Networks 3

3.1.1 2-Queue Network 3

3.1.2 Criss-Cross Network 4

3.2 Reinforcement Learning 5

3.3 Infinite Horizon Markov Decision Process 5

3.4 Average Reward 6

3.5 STability and OPtimality (STOP) 7

3.6 Adam Algorithm 8

4 Results **9**

4.1 Exploration on Lyapunov Function 9

4.1.1 Heatmaps 9

4.2 Adam and Policy Collapse 11

4.2.1 Adam Beta Exploration 11

4.2.2 Adam Beta Tuning is not enough 14

4.2.3 Intuition on Adam Betas 16

5 Discussion and Further Work **17**

Bibliography **18**

List of Figures

3.1 2-Queue Network 4

3.2 Criss-Cross Network 5

4.1 Optimal value function heatmap for states $[0, 30]$ of fully connected (medium load) in the 2-queue network 10

4.2 Fully connected with medium load 12

4.3 Faulty connections with high load 13

4.4 Faulty connections with very high load 13

4.5 Detailed graphs for STOP-2.5 with Adam $\beta_1 = 0.9, \beta_2 = 0.8$ 15

4.6 Detailed graphs for STOP-2.5 with Adam $\beta_1 = 0.9, \beta_2 = 0.8$ 16

List of Tables

3.1 Load parameters for 2-queue network for Figure 3.1 4

3.2 Load parameters for criss-cross network for Figure 3.2 5

Chapter 1

Introduction

Chapter 2

Related Work

2.1 Continual Reinforcement Learning

Our work focuses on continual reinforcement learning

2.2 Policy Collapse and Plasticity Loss

Chapter 3

Preliminaries

3.1 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.

3.1.1 2-Queue Network

The 2-queue network depicted in Figure 3.1, 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.

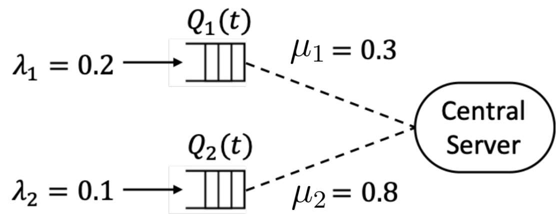


Figure 3.1: 2-Queue Network

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_1
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 3.1

3.1.2 Criss-Cross Network

The criss-cross network depicted in Figure 3.2, 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.

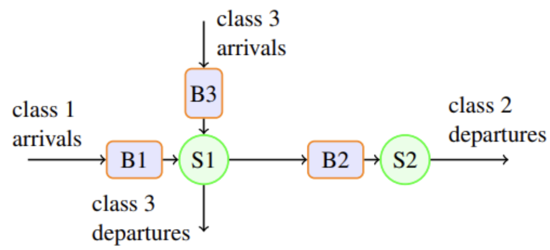


Figure 3.2: Criss-Cross Network

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 3.2

3.2 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.

3.3 Infinite Horizon Markov Decision Process

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}.

3.4 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.

3.5 STability and OPtimality (STOP)

Introduced in Pavse et al., STOP is an approach to create robust agent for continual learning problems with an emphasis on encouraging stability and optimality (2024). 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.

3.6 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:

$$m_t = \frac{\beta_1}{1-\beta_1^t} \cdot m_{t-1} + \frac{1-\beta_1}{1-\beta_1^t} \cdot g_t \tag{3.5}$$

$$v_t = \frac{\beta_2}{1-\beta_2^t} \cdot v_{t-1} + \frac{1-\beta_2}{1-\beta_2^t} \cdot g_t^2 \tag{3.6}$$

$$\theta_{t+1} = \theta_1 - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{3.7}$$

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 3.5 and 3.6 respectively. Then, we update the parameters in 3.7. 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

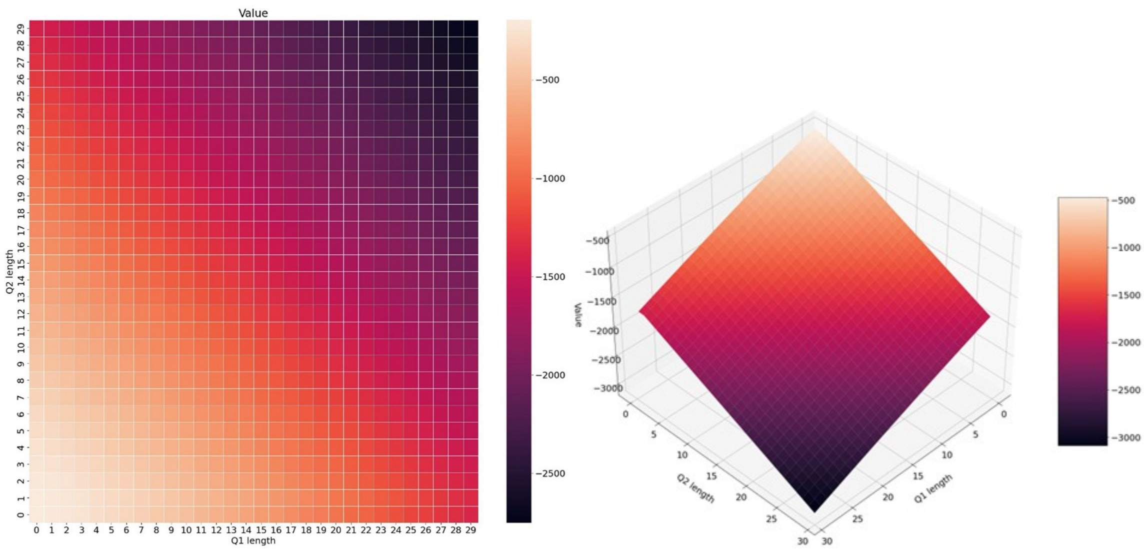
4.1 Exploration on Lyapunov Function

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 $\ell(s) \approx V^*(s)$.

4.1.1 Heatmaps

To gain a better insight on choosing the Lyapunov function, we generated heatmaps of the optimal value function and compared it with different Lyapunov function choices. Due to the unbounded nature of the queue lengths, we cannot honestly compute the optimal value function since the state space is unbounded. In the following results, the optimal value function depicted is an approximation from 3000 iterations of value iteration with the maximum queue length bounded by 120.

Fully Connected (Medium Load)



(a) Optimal value function heatmap in 2D (b) Optimal value function heatmap in 3D

Figure 4.1: Optimal value function heatmap for states $[0, 30]$ of fully connected (medium load) in the 2-queue network

The heatmaps from 4.1, illustrate the behavior of the optimal value function across the queue lengths where $Q_1, Q_2 \in [0, 30]$. The color intensity represents the value of each state, with lower value states being darker. The heatmap shows a nearly linear decrease in value as the average queue length increases. As we approach states where either Q_1 or Q_2 tends to 0, we see a slight non-linear shape, indicating that improvements are diminishing as we approach state $[0, 0]$.

To further verify the near-linear shape of the optimal value function, we used regression to model the cross-term between Q_1 and Q_2 in the optimal value function on different state bounds:

when $Q_1, Q_2 \in [0, 30]$:

$$V^*(s) \approx -0.09324 \cdot Q_1^2 - 0.09295 \cdot Q_2^2 - 46.06988 \cdot Q_1 - 43.32195 \cdot Q_2 \tag{4.1}$$

when $Q_1, Q_2 \in [0, 45]$:

$$V^*(s) \approx -0.05226 \cdot Q_1^2 - 0.05226 \cdot Q_2^2 - 47.04658 \cdot Q_1 - 43.51059 \cdot Q_2 \tag{4.2}$$

when $Q_1, Q_2 \in [0, 60]$:

$$V^*(s) \approx 0.00133 \cdot Q_1^2 + 0.00321 \cdot Q_2^2 - 48.82095 \cdot Q_1 - 47.11132 \cdot Q_2 \tag{4.3}$$

In the cross-term models 4.1, 4.2 and 4.3, we see that the magnitude of the coefficients for the linear terms Q_1 and Q_2 increases as we increase the bounds for the queue lengths. Similarly, the magnitude of the coefficients of the non-linear terms decreases as we increase the bounds for the queue lengths.

Similar to Pavse et al. (2024), we consider the Lyapunov function to be of the form $\sum_i^n Q_i^{(\beta+1}$ for $\beta > 0$, inspired by variants of MaxWeight (Stolyar, 2004).

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).

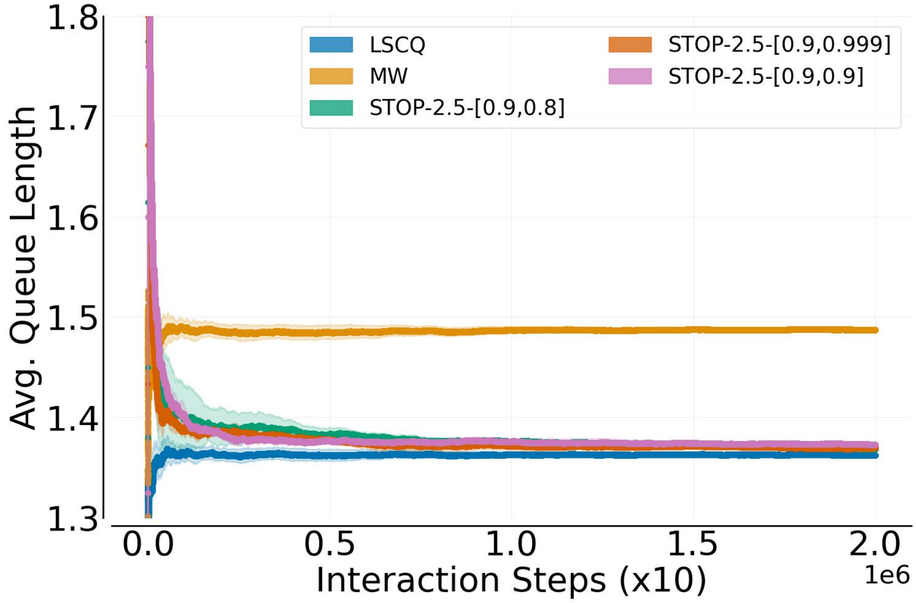


Figure 4.2: Fully connected with medium load

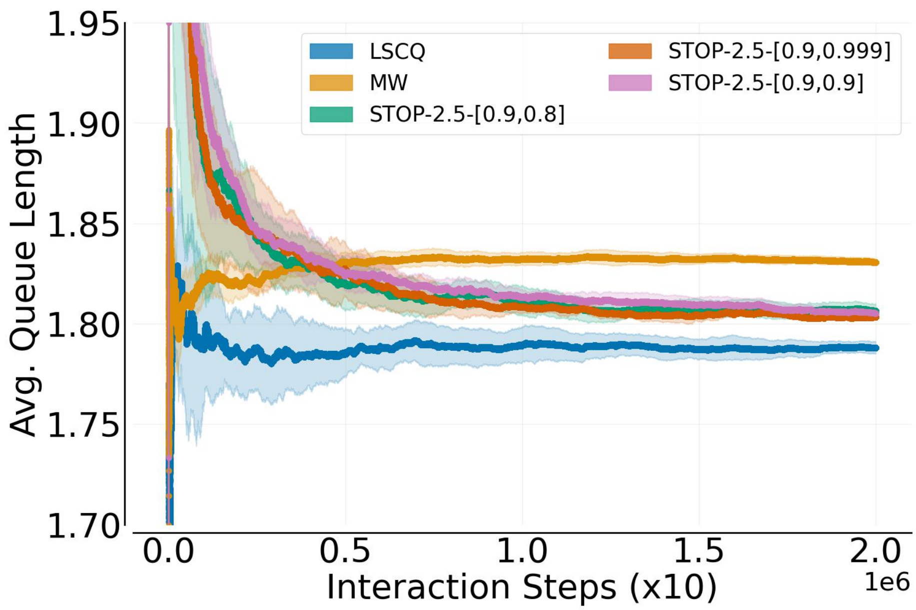


Figure 4.3: Faulty connections with high load

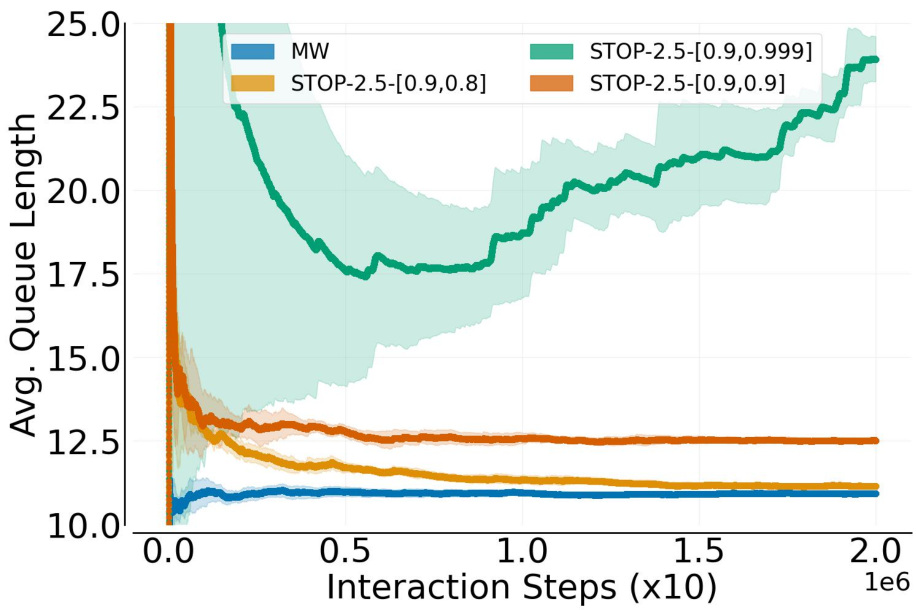


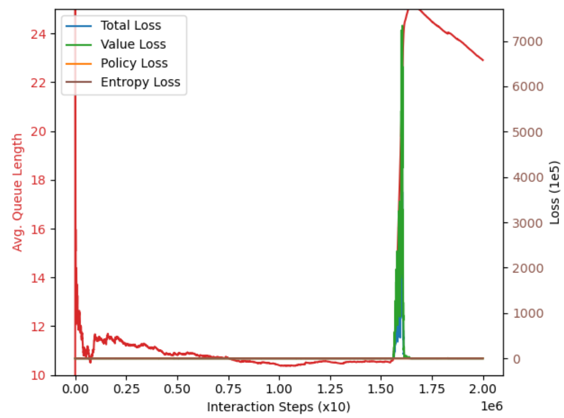
Figure 4.4: Faulty connections with very high load

The algorithms 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.

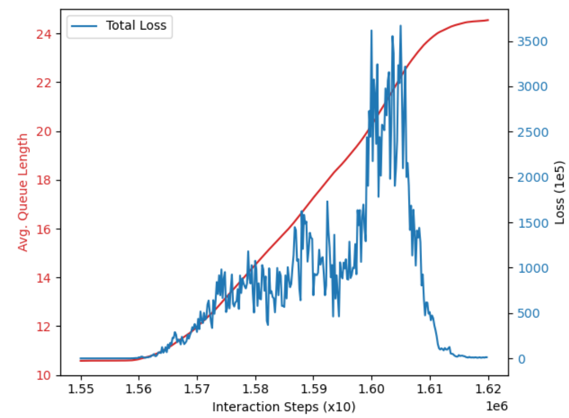
Our results demonstrates that STOP consistently performs well in two settings: the fully connected (medium load) and faulty connections (high load) settings, as shown in Figure 4.2 and Figure 4.3 respectively. This is true regardless of the Adam beta hyper-parameters. However, in the faulty connections (very high load) setting (Figure 4.4), 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

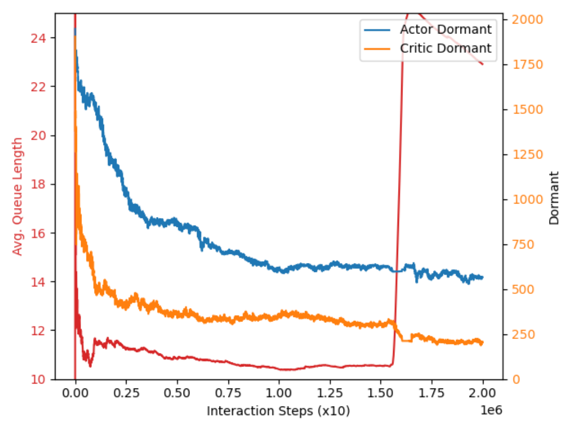
Initial results in Figure 4.4 show promising results, however on deeper inspection of the 20 independent trials, we observe policy collapse still occurs when the Adam $\beta_2 = 0.8$. In the following results, we show the two such cases we try to find the cause.



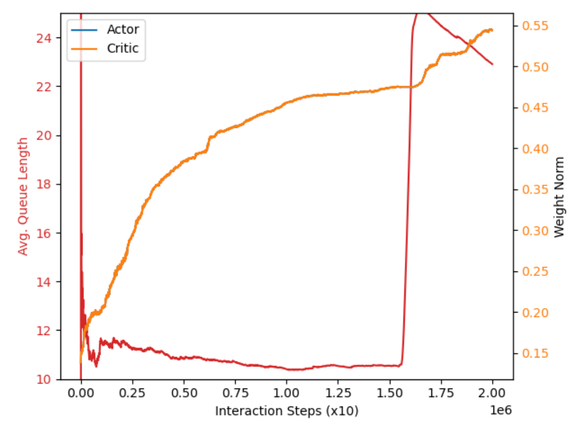
(a) Avg. Queue Length vs. Loss



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



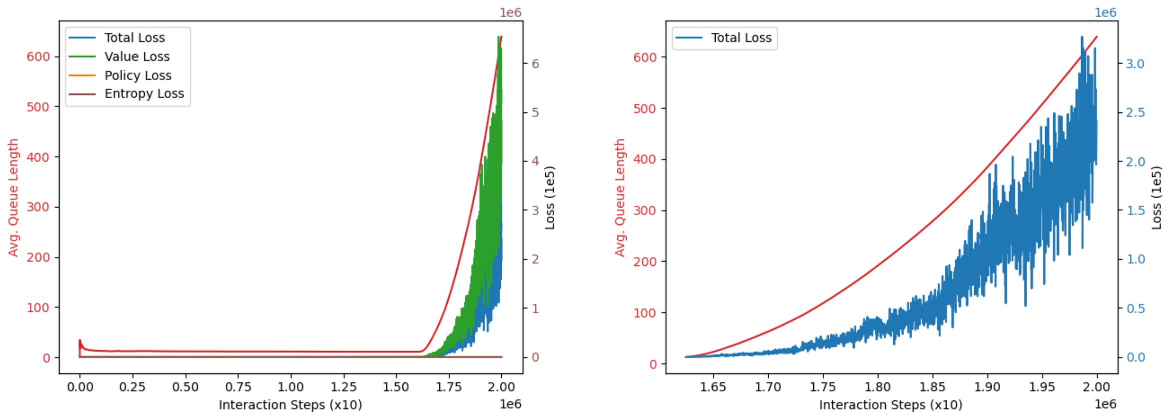
(c) Avg. Queue Length vs. No. of Dormant Units



(d) Avg. Queue Length vs. Weight Magnitude

Figure 4.5: Detailed graphs for STOP-2.5 with Adam $\beta_1 = 0.9, \beta_2 = 0.8$ for seed 175587

From Figure 4.5.c 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 depicted in 4.5.a and 4.5.b, where there was a sudden decrease in performance whenever there was a spike in the loss. This is similar to how Dohare et al. (2023) had observed sudden non-zero gradient which led to a spike in the updates. In the case of 4.5, we see that after the sudden spike in performance loss, the agent is able to recover itself as the average queue length decrease. Next we show results of another seed Figure 4.6, where we find similar results as 4.5, however this time the agent is not able to recover.



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

Figure 4.6: Detailed graphs for STOP-2.5 with Adam $\beta_1 = 0.9, \beta_2 = 0.8$ for seed 496018

4.2.3 Intuition on Adam Betas

Chapter 5

Discussion and Further Work

Bibliography

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

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*.

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.