Deeper Exploration with Distributional Reinforcement Learning

A Project Report

submitted by

YOGESH TRIPATHI

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

June 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Deeper Exploration with Distributional Reinforcement Learning**, submitted by **Yogesh Tripathi**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Balaraman Ravindran
Research Guide
Professor
Dept. of CSE
IIT Madras, 600036

Place: Chennai

Date: 17th June 2020

ACKNOWLEDGEMENTS

I am deeply indebted to my adviser, Prof. Balaraman Ravindran, who always encouraged me by providing constructive and valuable feedback from the beginning of this project. Prof. Ravindran gave me complete freedom in terms of surveying the literature, developing methodologies, and testing them, but also ensured that the project's goal remains achievable within the time and resource constraints by checking my progress from time to time. I want to thank my colleague, Vishnu Veerathu, for discussing thought-provoking ideas, clearing my doubts, and pointing out flaws and strengths in my work. I am incredibly grateful to my faculty advisor, Prof. C. Chandra Sekhar, who provided me crucial help and support in difficult times. I am also indebted to RISE Lab, IIT Madras, for providing me with the necessary computing resources without which this project would not have been successful. I am grateful to all the professors and staff of the Department of Computer Science and Engineering, IIT Madras, who made my journey fruitful and memorable.

I want to thank my friends, who were a constant support throughout my undergraduate. I am deeply thankful to my parents for their numerous sacrifices and immense love, which provided me a conducive environment to work. I am also grateful to my grandparents for their eternal blessings.

ABSTRACT

KEYWORDS: Explore-Exploit dilemma; Distributional Reinforcement Learning; Quantile Option Architecture; Quantile Regression Deep Q-Network.

Explore-Exploit dilemma is one of the major challenges faced by reinforcement learning agents. Commonly used strategies like ϵ-greedy exploration fail to guide the agent towards a more diverse set of exploratory steps. The recent developments in distributional reinforcement learning provide a rich representation of the return function, thus opening avenues for more systematic exploration strategies.

In this project, we address the explore-exploit dilemma based on recent advances in distributional reinforcement learning. We develop an exploration strategy which utilizes the return distribution in an option-based framework to generate deep exploration policies. We also leverage other improvements like Double Q-Learning, Prioritized Experience Replay and Multi-step learning in the options-framework to ameliorate exploration. Our algorithm provides significant improvement in performance in three environments which demand good exploration strategies for finding the optimal policy. We also examine the contribution of each improvement employed in our algorithm with a detailed ablation study.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ANN	Artificial Neural Network
C-51	Categorical-51
CNN	Convolutional Neural Network
Deep-QUOTA	Deep Quantile Option Architecture
DL	Deep Learning
DLTV	Decaying Left Truncated Variance
DQN	Deep Q-Network
KL-divergence	Kullback-Leibler Divergence
MDP	Markov Decision Process
QRDQN	Quantile Regression Deep Q-Network
QUOTA	Quantile Option Architecture
RL	Reinforcement Learning
TD	Temporal Difference

NOTATION

$\mathcal{M}$	Markov Decision Process (MDP)
$\mathcal{S}$	State space of MDP
$\mathcal{A}$	Action space of MDP
$\mathcal{P}$	Transition kernel of MDP
$\mathcal{R}$	Reward function of MDP
Ω	Option space
$[N]$	The set $\{1, 2, \ldots, N\}$
s_t	State at time step, t
a_t	Action taken at time step, t
r_t	Reward observed at time step, t
o_t	Option executed at time step, t
γ	Discounting factor
π	Policy of the agent
$Z(s, a)$	Return distribution for state, s and action, a
$\rho_\tau^\kappa(x)$	Huber quantile loss
$\pi^j(s)$	Intra option policy
$Q(s, a)$	$\mathbb{E}[Z(s, a)]$, the state-action value function
ϵ	Random action selection probability
ϵ_Ω	Random option selection probability
β	Option termination probability
N	Number of quantiles
M	Number of options
$\{q_i\}_{i=1,\ldots,N}$	Quantile estimation parameterized by θ and θ^-
Q_Ω	Option value parameterized by ψ and ψ^-
B	Batch-size
χ	Weight for quantile-loss priority
n	Horizon length in multi-step learning
F	Training frequency
T	Target and current network synchronization frequency
$\epsilon_1, \epsilon_2, \beta_c$	Priority experience replay parameters
$P_q(i)$	Quantile loss priority for sample i
$P_o(i)$	Option loss priority for sample i
$P(i)$	Probability of choosing sample i
Γ	Option to quantiles mapping function
α	Step-size
$\doteq$	By definition

CHAPTER 1

INTRODUCTION

Reinforcement Learning (RL) (Sutton and Barto [2018]) solves an online sequential decision making problem in which an agent interacts with an environment, by executing actions, and the environment responds by changing the state of the agent and giving a scalar reward. This interaction is modeled using a Markov Decision Process. The objective of the agent is to maximize the return, which is the expected value of cumulative reward. RL agents learn through experience, which naturally leads to a trade-off between exploration and exploitation. *Explore-Exploit dilemma* is the agent's dilemma at every step to either play the best action in its current state based on previous experience, or execute an action which leads it to unexplored parts of the state space, which might lead to better return.

Distributional perspective on RL, given by Bellemare *et al.* [2017], maintains the entire return distribution for every state-action pair in the MDP, unlike just the expected value of return in traditional Expected RL. This rich representation can be utilized the guide the exploration. In this work, we maintain the return distribution in the form of N quantiles of the distribution, as studied in Quantile-Regression Deep Q-Network by Dabney *et al.* [2017]. We propose to execute actions by sampling a particular quantile from this distribution and acting greedily based on it. We use options framework (Sutton *et al.* [1999]) by keeping a top-level option policy, corresponding to each quantile, and sample ϵ_Ω-greedily with respect to the option values, inspired from Quantile-Option Architecture by Zhang *et al.* [2018]. As each option makes a different assumption on the return value, this architecture leads to generating deep exploration policies based on the option selected. We also utilize other improvements like Double Q-Learning (van Hasselt *et al.* [2015]), Prioritized Experience Replay (Schaul *et al.* [2015]) and n-step learning (Sutton [1988]) in the distributional RL set-up to boost the exploration.

1.1 Problem Statement

Exploration is a difficult task for RL agents, especially in large state and action spaces. By making the exploration decision more systematic, we can avoid a lot of wasteful exploratory steps which slow down the learning process. Most of the RL algorithms employ ϵ-greedy strategy, which selects a greedy action with respect to its current estimates of expected return with probability $1 - \epsilon$ and a random action with probability ϵ. Such a strategy does not inject any guidance into the exploratory steps. Certain set of exploratory actions are favorable in certain parts of environment, which is not taken into account in an ϵ-greedy setup.

1.1.1 Deep Exploration Policy

When employing an ϵ-greedy strategy, ϵ usually begins with a high value and eventually decays to very small values. This means that beyond a certain number of training steps, ϵ-greedy exploration would only deviate from the greedy-policy for very few steps and return back. This leads to very poor exploration of the state space as the trajectory taken mostly remains restricted to the one greedy with respect to agent's current expected return estimates. Deep exploration, suggested by Osband *et al.* [2016], generates full length exploratory policies using a Bootstrapped DQN. It maintains a set of action value estimates and acts greedily with respect to a sample from this set, in the entire trajectory. This idea is inspired by Thompson Sampling which samples from the posterior distribution and chooses actions based on the sampled value.

1.1.2 Using Distributional Reinforcement Learning

Distributional RL, introduced by Bellemare *et al.* [2017], maintains the entire return distribution for each state-action pair of the MDP. This provides richer representation of return than just maintaining the expected values of return for every state-action pair, as in case of Expected RL. The agent can benefit from this representation, by guiding its exploratory steps based on the return distribution. In this work, we try to generate deep exploration policies for our agent based on

the return distribution. We study this in the options framework, as introduced by Zhang *et al.* [2018], in their Quantile Option Architecture (QUOTA). We also investigate methods to introduce full hierarchy in QUOTA algorithm. We then examine the effects of introducing several improvements like Double Q-Learning, Prioritized Experience Replay and n-step learning, after adapting them to our setup.

1.2 Motivation

Most of the work based on Distributional Reinforcement Learning (Bellemare *et al.* [2017], Dabney *et al.* [2017], Gruslys *et al.* [2017] and Barth-Maron *et al.* [2018]) use ϵ-greedy exploration, which is a very unstructured and *shallow* decision making strategy, underutilizing the rich return distribution maintained by them. As introduced in QUOTA framework, the return distribution can be utilized by the agent for much efficient exploration by generating deep exploration policies. Although it utilizes an options framework, QUOTA is not a purely hierarchical setup. This is because in learning the return distribution, it selects the target for a sample greedily with respect to the mean of the action-value estimates of next state. However, a pure hierarchical setup would do this based on the option it employs in the next state. When an agent selects an option, it makes certain assumptions about the return that it can obtain from the state, which should be considered when creating the target estimates for that state. By disregarding the option and creating the target solely based on greedy policy, the agent's preferred assumptions will not be reflected in the quantile estimates.

The main motivation of this work is to answer the question of introducing a pure hierarchical setup in QUOTA framework to generate deep exploration policies and study the benefits of several extensions to Deep Q-Network like Double Q-learning, Prioritized Experience Replay and Multi-step learning, inspired by the Rainbow architecture by Hessel *et al.* [2017], after adapting them to the framework of QUOTA.

1.3 Contributions of this work

We introduce Deep-QUOTA, a purely hierarchical Quantile Option Architecture, in order to generate exploration strategies for the agent. Each quantile corresponds to an option and the agent selects a quantile and acts greedily with respect to it, throughout the entire trajectory. When the agent selects a lower quantile, it makes a *pessimistic* assumption on the return value from its current state and *optimistic* assumption when it selects an upper quantile. The agent adaptively changes its option (assumptions on return values), based on its current state and updates its exploration strategy.

The two main contributions of this project are as follows:

- Deep-QUOTA, a purely hierarchical Quantile Option Architecture, which not only uses quantile options to alter the agent's behaviour policy but also selects the target value for training the QRDQN component, by choosing the action recommended by the best option in the next state. This is different from the methodology employed by QUOTA, which uses greedy action with respect to expected return values for the next state for training QRDQN.
- We modify Double Q-learning, Prioritized Experience Replay and Multi-step learning to adapt to the Deep-QUOTA framework, and study their benefits in ameliorating the exploration process.

We propose three different MDPs to evaluate the performance of our algorithm. First two are chain MDPs - one, in which action selection based on higher quantiles would lead to faster exploration and discovery of optimal policy and in the other MDP, action selection based on a lower quantile would benefit exploration. The third MDP is a sparse reward gridworld in which discovery of highly rewarding state would require an efficient *deep* exploration strategy. We evaluated the performance of Deep-QUOTA agent against QUOTA, QRDQN and DQN and established superior performance of our algorithm in all three MDPs. We then analysed the significance of each component of our algorithm by an ablation study of all the improvements employed. We found that Prioritized Experience Replay is the most significant addition to our architecture, giving highest boost in performance. Thus, we will empirically show that Deep-QUOTA provides exploration benefits and superior performance than the chosen baselines.

1.4 Organization of the Thesis

The thesis is organized into five main chapters. The first chapter gives an overview of the Deep-QUOTA architecture, the motivation behind designing it and a gist of contributions of this work. The second chapter discusses the background needed to understand this work, and the closely related literature that this work draws inspiration from. The third chapter gives the Deep-QUOTA algorithm for discrete action space and explains the intuition behind it. The fourth chapter presents an empirical evaluation of our algorithm and an ablation study, along with the experimental setup, implementation and hyperparameter details, and environment used for evaluation. The final chapter gives our closing remarks and future directions in which this work can be extended.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we will first discuss the background needed to understand this work. We will formally study the Markov Decision Process, expected RL, distributional RL and options framework. Then we will discuss the closely related literature that this work draws inspiration from.

2.1 Background

We model the sequential decision making (RL) problem using a Markov Decision Process(MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. $\mathcal{S}$ represents the state space, $\mathcal{A}$ represents the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the transition probability kernel and $\gamma \in [0, 1]$ represents the discounting ratio. We use $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ to denote the stochastic policy used by the agent.

2.1.1 Solving the RL problem

Let $Z^\pi(s, a)$ denote the random variable corresponding to the discounted sum of rewards obtained from state s on executing action a, defined in Equation (2.1).

$$Z^\pi(s,a) \doteq \sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \text{ , where } S_0 = s, A_0 = a, S_{t+1} \sim p(\cdot \mid S_t, A_t), A_t \sim \pi(\cdot \mid S_t) \tag{2.1}$$

We also note that the state-action value function $Q^\pi(s, a) = \mathbb{E}_{\pi,\mathcal{P},\mathcal{R}}[Z^\pi(s, a)]$. The recursive Bellman equation is defined by Equation (2.2).

$$Q^\pi(s,a) = \mathbb{E}[R(s,a)] + \gamma \mathbb{E}_{S',A'}[Q^\pi(S', A')] \tag{2.2}$$

Solving an RL problem is essentially determining an optimal policy π^*, such that $Q^{\pi^*}(s, a) \geq Q^\pi(s, a),\ \forall(\pi, s, a)$. All possible optimal policy share the same state-

action value function Q^*, which is the fixed point of the Bellman optimality operator $\mathcal{T}$ (Bellman [2003]). The Bellman optimality equation is given by Equation (2.3).

$$\mathcal{T}Q(s,a) \doteq \mathbb{E}[R(s,a)] + \gamma \mathbb{E}[\max_{a'} Q(S', a')] \tag{2.3}$$

Q-Learning, given by Watkins and Dayan [1992], is an off-policy online algorithm for estimating state-action values for the optimal policy, given by Equation (2.4).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \tag{2.4}$$

where $(s_t, a_t, r_{t+1}, s_{t+1})$ is a transition observed by the agent and α denotes the step size. Mnih *et al.* [2015] extended this to very high dimensional state spaces using a Deep Q-Network (DQN), which uses an artificial neural network (ANN), θ in the form of a deep convolutional neural network (CNN) to parameterize Q. DQN uses stochastic gradient descent to minimize the loss function in Equation (2.5).

$$\frac{1}{2}(r_{t+1} + \gamma \max_a Q_{\theta^-}(s_{t+1}, a) - Q_\theta(s_t, a_t))^2 \tag{2.5}$$

where $(s_t, a_t, r_{t+1}, s_{t+1})$ is sampled from a replay buffer and θ^- are the parameters of a target network, which is periodically synchronized with θ.

2.1.2 Distributional RL

Bellemare *et al.* [2017] studies a distributional perspective on solving the RL problem, by maintaining the entire return distribution for every state-action pair (s, a), which is the random variable $Z^\pi(s, a)$ defined in Equation (2.1). Similar to the Bellman equation shown in Equation (2.3), they give a distributional version given as:

$$\forall (s,a),\ Z^\pi(s,a) \stackrel{D}{=} R(s,a) + \gamma Z(S', A'),\ \text{where } S' \sim P(\cdot \mid s, a), A' \sim \pi(\cdot|S') \tag{2.6}$$

where $X \stackrel{D}{=} Y$ denotes that X and Y are two random variables distributed according to the same law. Bellemare *et al.* [2017] also give the distributional version of

Bellman optimality operator for control, $\mathcal{T}$ defined by:

$$\forall(s,a),\ \mathcal{T}Z(s,a) \overset{D}{=} R(s,a) + \gamma Z(S^{'}, \arg\max_{a^{'}} Q(S^{'}, a^{'})),\ \text{where } S^{'} \sim P(\cdot \mid s,a) \tag{2.7}$$

Dabney *et al.* [2017] use quantile approximation of $Z(s,a)$, given by $Z_\theta(s,a)$, represented by a uniform mix of N supporting quantiles:

$$Z_\theta(s,a) \doteq \frac{1}{N}\sum_{i=1}^{N} \delta_{q_i(s,a;\theta)} \tag{2.8}$$

where δ_x denotes the Dirac Delta function at $x \in \mathbb{R}$ and $q_i(s,a;\theta)$ denoted the ith quantile of $Z(s,a)$. The ANN, θ used to parameterize the quantile values is trained by minimizing the Huber quantile regression loss, defined as:

$$\mathcal{L}(s_t, a_t, r_{t+1}, s_{t+1}) \doteq \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \rho^{\kappa}_{\hat{\tau}_i}(r_{t+1} + \gamma q_j(s_{t+1}, a^*; \theta) - q_i(s_t, a_t; \theta)) \tag{2.9}$$

where $(s_t, a_t, r_{t+1}, s_{t+1})$ represents a sample from the experience replay buffer, $a^* = \arg\max_a \frac{1}{N}\sum_{k=1}^{N} q_k(s_{t+1}, a)$, $\hat{\tau}_i = \frac{\tau_{i-1}+\tau_i}{2}$ and $\tau_i = \frac{i}{N}$. $\rho^{\kappa}_{\tau}(u) = |\tau - \mathbb{I}_{u<0}| L_\kappa(u)$ where $\mathbb{I}$ is the indicator function and $L_\kappa(u)$ represents the Huber loss, defined as:

$$L_\kappa(u) \doteq \begin{cases} \frac{1}{2}u^2 & ;\ x \le \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & ;\ \text{otherwise} \end{cases}$$

2.1.3 Options Framework

Sutton *et al.* [1999] developed a temporal abstraction of action, which they called option. An option o from the option set Ω is defined by the triplet $(\mathcal{I}_o, \pi_o, \beta_o)$. $\mathcal{I}_o \subseteq \mathcal{S}$ denotes the subset of states in which option o can be initiated, $\pi_o : \mathcal{S} \times \mathcal{A} \to [0,1]$ denotes the intra-option policy employed by the agent when executing o and $\beta_o : \mathcal{S} \to [0,1]$ is the termination function which denotes the probability of option o terminating in a particular state.

Sutton *et al.* [1999] defined option value function $Q_\Omega(s,o)$ for every state-option pair (s,o), which can be learned using Intra-Option Q-Learning algorithm given

by the equation below.

$$Q_\Omega(s_t, o_t) \leftarrow Q_\Omega(s_t, o_t) + \alpha(r_{t+1} + \gamma(\beta_{o_t}(s_{t+1}) \max_{o'} Q_\Omega(s_{t+1}, o') + (1 - \beta_{o_t}(s_{t+1}))Q_\Omega(s_{t+1}, o_t)) - Q_\Omega(s_t, o_t)) \tag{2.10}$$

Here, $(s_t, o_t, r_{t+1}, s_{t+1})$ denotes a sample in the trajectory where the agent executes option o_t in state s_t.

With that, we have summarized all the machinery needed to understand our algorithm.

2.2 Related Work

Using Deep Exploration policies was first introduced by Osband *et al.* [2016] by using a Bootstrapped DQN. They maintain K separate state-action value function estimates in the form of K heads in the DQN, sharing the state-representation network. For every episode, the agent selects one head and acts ϵ-greedily with respect to the Q value estimates maintained by that head. This idea is inspired by Thompson Sampling which essentially samples from the posterior distribution of the parameters and acts greedily based on the sample. They also show that due to different initializations of each head, each of them generate a different policy which aids the exploration process. However, the diversity induced by the initialization is neither structured nor explainable. The amount of diversity in the policies generated also varies significantly based on the initialized parameter values.

Distributional RL was introduced by Bellemare *et al.* [2017], in which they empirically showed the benefits of maintaining the return distribution. The C-51 algorithm given in this work minimizes the KL-divergence between target return distribution and maintained distribution over the canonical atoms. However, in the theoretical results, they show convergence guarantees for minimizing the Wasserstein metric. Quantile Regression DQN by Dabney *et al.* [2017] removes this gap by using a quantile approximation of return distribution, which is trained by minimizing the Huber quantile loss, which they show is equivalent to mini-

mizing the Wasserstein metric. Yang *et al.* [2019] develop a fully parameterized quantile function which removes the hyperparameters of return distribution like number of atoms in C-51 and number of quantiles in QRDQN. In this work, we develop our methodology using QRDQN algorithm due to its simplicity and its elegant fitting with the options framework.

There has been a lot of work recently on using distributional RL for efficient exploration. Mavrin *et al.* [2019] introduced the Decaying Left Truncated Variance (DLTV) algorithm which uses a decaying exploration bonus, $c_t\sqrt{\sigma_+^2}$, where c_t is the decay schedule and σ_+^2 is the variance of the return distribution computed using only the quantiles higher than the median. They argue that the decaying schedule is to account for the exponentially reducing parametric noise and the left truncated variance gives higher weightage to the right tail of the distribution, which is more relevant for instantiating optimism in face of uncertainty.

Tang and Agrawal [2018] take an approximate Bayesian inference approach towards this problem. They assume that the parameters of the return distribution are sampled from a distribution parameterized by ϕ. Their objective is to minimize the KL-divergence between the distribution over the parameters and the posterior probability of N samples from the return distribution, over the parameter ϕ. However, they show the advantage of this approach only on simple environments. Nikolov *et al.* [2018] use the return distribution for evaluating the information gain of a state-action pair. They use information-gain to evaluate a regret-information ratio and the agent selects action based on this value. The entire return distribution is only used to evaluate the variance, which is a heavy underutilization of the information contained in the distribution.

Zhang *et al.* [2018] maintain the return distribution using QRDQN and use it in an options framework, which they call Quantile Option Architecture (QUOTA). QUOTA maintains an option corresponding to each quantile in the return distribution. At every step, the agent selects an option, ϵ_Ω-greedily with respect to the option values and executes action greedily based on the quantile corresponding to the selected option. However, QUOTA does not obey the hierarchical setup in training the QRDQN component as it does not involve setting the target values based on the option executed by the agent in the subsequent state. Instead, it

uses a greedy action with respect to the Q-values of the subsequent state to set its target. This breaks the assumptions made the agent about the return value of its current state in the form of options. We modify the QUOTA framework to introduce a purely hierarchical setup, which gives greater meaning to the agent's assumptions when training the QRDQN component.

This work also draws inspiration from Rainbow architecture, proposed by Hessel *et al.* [2017], which studies the benefits of several improvements to the DQN algorithm. We modify Double Q-learning, Prioritized Experience Replay and Multi-step learning in the distributional RL framework and perform an ablation study of each of these improvements.

CHAPTER 3

DEEP-QUOTA: DEEP QUANTILE OPTION ARCHITECTURE

In this chapter, we will discuss the intuition behind the Deep-QUOTA framework and the full Deep-QUOTA algorithm for discrete action space, along with all the details of improvements used for efficient exploration.

3.1 Need for Deep-QUOTA

Deep-QUOTA makes two significant modifications in QUOTA framework. First, it utilizes the hierarchical framework in training the QRDQN component. Second, it adapts certain improvements in Deep RL to its framework and benefits from them. We will now discuss the motivation behind each of them.

3.1.1 Using the hierarchy

The Quantile Option Architecture (QUOTA) proposed by Zhang *et al.* [2018], although based on an options framework, does not employ a purely hierarchical setup. For a sample $(s_t, a_t, r_{t+1}, s_{t+1}, o_t)$ in the replay buffer, the QRDQN component of QUOTA is trained by minimizing the following loss function:

$$\mathcal{L}(s_t, a_t, r_{t+1}, s_{t+1}, o_t) \doteq \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \rho^{\kappa}_{\hat{\tau}_i}(r_{t+1} + \gamma q_j(s_{t+1}, a^*; \theta) - q_i(s_t, a_t; \theta)) \quad (3.1)$$

Here, a^* is chosen as $\arg\max_a \frac{1}{N} \sum_{k=1}^{N} q_k(s_{t+1}, a)$, which clearly shows the QRDQN training completely disregards the entire options framework. When a certain option executed in a state leads the agent to high rewarding states, then it is desirable to have that option executed more frequently in that state. If the quantile adjustments are oblivious to the option executed, then the option's benefits will only be visible in the state-option values.

We propose to choose the target for quantile estimation using Equation (3.1), except that a^* is chosen using the action recommended by best option in the subsequent state, as shown in Equation (3.2).

$$a^* \doteq \arg\max_a q_{\Gamma(o)}(s_{t+1}, a; \theta), \;\; o \doteq \arg\max_{o'} Q_\Omega(s_{t+1}, o') \tag{3.2}$$

where $\Gamma : \Omega \to 2^{[N]}$ maps every option to set of quantiles that option acts greedily with respect to, and N is the number of quantiles. If the QRDQN component is trained based on this target, the inferences that the agent draws about the state in the form of option values, is adhered to even in quantile adjustments. This adherence is crucial, which can be explained through the following example.

Suppose an agent runs an option corresponding to lower quantiles in a trajectory and obtains a good return. This means that a pessimistic assumption on return value leads to better trajectory in those states, which will be reflected in the option values. Now, when training the QRDQN component, QUOTA will select subsequent action for target without exploiting the additional knowledge about the benefit of pessimistic assumption, hence leading to higher chance of the action being sub-optimal. Since training QRDQN is a more data-intensive task than training the option-value network, utilizing the relatively accurate estimates given by the option-value network is desirable. Deep-QUOTA avails this information stored in the hierarchy, by choosing the action recommended by the best option in the subsequent state for constructing the QRDQN training target. Since option-value network would reflect the advantage of executing actions based on pessimistic assumption, there is a higher chance that true optimal actions are selected in subsequent states while training.

3.1.2 Other improvements

Rainbow architecture by Hessel *et al.* [2017] inspired us to adapt the improvements developed for DQN to the Deep-QUOTA setting and study if it aids exploration and speeds up learning. We maintain two copies of the Deep-QUOTA network, current and target, parameterized by θ and θ^-. We select the target for QRDQN and the option-value network based on target network, with the best

option/action selected using the current network. van Hasselt *et al.* [2015] show that such architecture helps in eliminating maximization bias in the estimates. Schaul *et al.* [2015] show the benefits of having a Prioritized Experience Replay based on TD Error. We prioritize sampling from the experience buffer based on certain proportion of Huber quantile loss and Intra-option Q-Learning loss. We also use Multi-step Q-learning, given by Sutton [1988], for both Option-network and QRDQN training, leading to construction of more stable targets. All of these additions provide a performance boost in our chosen environment, among which, the modified Prioritized Experience Replay provides the greatest benefit.

3.2 Deep-QUOTA: the details

Deep-QUOTA algorithm can be studied in two parts, just like in the last section. We will describe each part in detail below and give the complete algorithm in the end.

We maintain N quantile estimations $\{q_i\}_{i=1,\dots,N}$ for quantile levels $\{\hat{\tau}_i\}_{i=1,\dots,N}$ with a Quantile Regression DQN, parameterized by θ. We use M options sharing same initiation set $\mathcal{S}$, same termination function β, which is a constant function. $\Gamma : \Omega \to 2^{[N]}$ denotes the set of quantiles corresponding to which each option acts. In our setup, we use $\Gamma(o_i) \doteq \{(i-1)K + j \mid j \in [K]\}$, each option thus averaging over K quantiles, to increase stability (where $K = N/M$, we assume N is divisible by M). Each option o_j recommends action ϵ-greedily with respect to mean of the quantiles obtained from $\Gamma(o_j)$. π^j, the Intra-option policy for option o_j is given by Equation (3.3).

$$\pi^j(s) \doteq \begin{cases} \arg\max_a \frac{1}{K} \sum_{k \in \Gamma(o_j)} q_k(s,a) & ; \text{with probability } 1-\epsilon \\ \text{random action} & ; \text{with probability } \epsilon \end{cases} \tag{3.3}$$

The option-value network parameter, ϕ is learned using Intra-Option Q-learning, exactly like QUOTA. At time step t, a new option is selected with probability β and the previous option is continued with probability $1-\beta$. When selecting a new option, the greedy option with respect to option value estimates Q_Ω is selected with probability $1 - \epsilon_\Omega$, and a random option with probability ϵ_Ω. The network

architecture for Deep-QUOTA is shown in Figure 3.1.

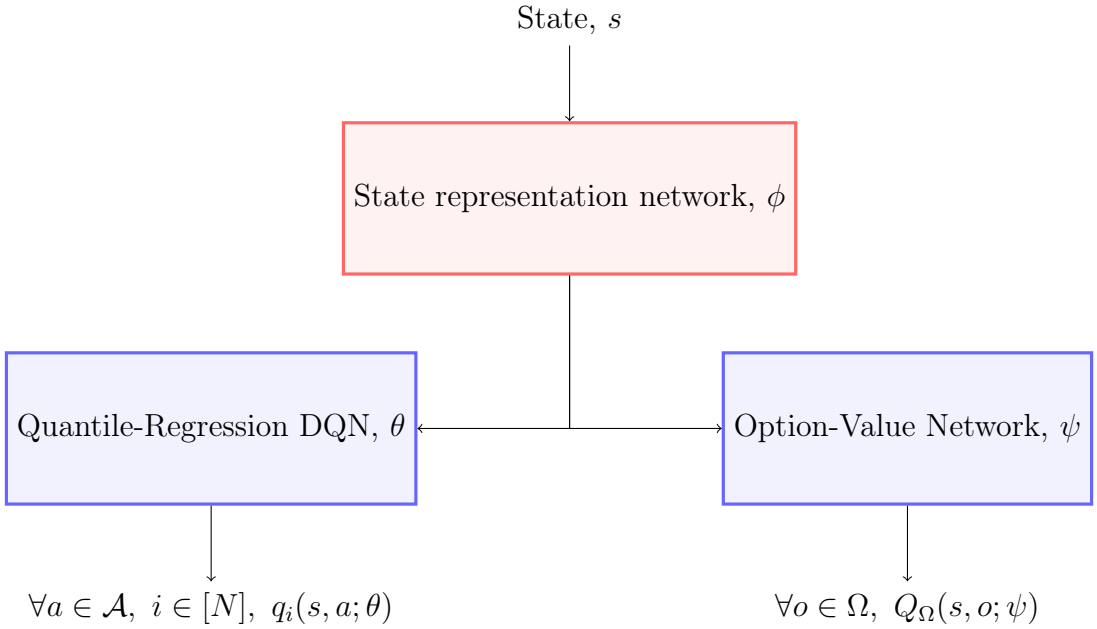

Figure 3.1: Deep-QUOTA network architecture: The are two components - Quantile regression DQN, trained by minimizing Huber quantile loss, and Option Value Network, trained by minimizing Intra-Option Q-learning loss. Both components share the state representation network.

3.2.1 Hierarchical QRDQN

The QRDQN component of Deep-QUOTA is trained by minimizing the Huber quantile loss function using Stochastic Gradient Descent. We maintain two QRDQN networks, current and target, parameterized by θ and θ^- respectively. For a sample $(s_t, a_t, r_{t+1}, s_{t+1}, o_t)$, the loss is given by:

$$\mathcal{L}(s_t, a_t, r_{t+1}, s_{t+1}, o_t) \doteq \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \rho^{\kappa}_{\hat{\tau}_i}(r_{t+1} + \gamma q_j(s_{t+1}, a^*; \theta^-) - q_i(s_t, a_t; \theta)) \quad (3.4)$$

where a^* is chosen to be the action recommended by the best option in the subsequent state, as given below:

$$a^* \doteq \arg\max_a \frac{1}{K} \sum_{k \in \Gamma(o)} q_k(s_{t+1}, a; \theta),\ \ o \doteq \arg\max_{o'} Q_\Omega(s_{t+1}, o'; \psi) \quad (3.5)$$

Note the choice of θ instead of θ^- in Equation (3.5), which signifies the Double-Q learning alteration we use in this work for creating the target values.

3.2.2 Other improvements

The three improvements that we have employed in the Deep-QUOTA algorithm are detailed below.

Double Q-Learning

For estimating the Option-values too, we maintain two option-value networks, current and target, parameterized by ψ and ψ^- respectively, which are synchronized periodically. The option-value loss for a sample $(s_t, a_t, r_{t+1}, s_{t+1}, o_t)$, given by the Equation (3.6), is minimized using Stochastic Gradient Descent.

$$\begin{aligned}\mathcal{L}_\Omega(s_t, a_t, r_{t+1}, s_{t+1}, o_t) \doteq \frac{1}{2}(r_{t+1} + \gamma(\beta Q_\Omega(s_{t+1}, \arg\max_o Q_\Omega(s_{t+1}, o; \psi); \psi^-) \\ +(1-\beta)Q_\Omega(s_{t+1}, o_t; \psi^-)) - Q_\Omega(s_t, o_t; \psi))^2\end{aligned} \tag{3.6}$$

The variation of Double Q-learning used in training QRDQN is already given in Section 3.2.1.

Prioritized Experience Replay

For every sample present in the experience buffer, we associate two priority values quantile-loss priority and option-loss priority, denoted as P_q and P_o. P_q and P_o for i^{th} sample in the buffer is calculated as:

$$\begin{aligned}P_q(i) &\doteq |\mathcal{L}(i)|^\omega + \epsilon_1 \\ P_o(i) &\doteq |\mathcal{L}_\Omega(i)|^\omega + \epsilon_2\end{aligned}$$

where $L(i)$ and $L_\Omega(i)$ are the Huber quantile loss and option-value loss for i^{th} sample. For brand new samples, where $\mathcal{L}$ and $\mathcal{L}_\Omega$ values are not known, we set P_q and P_o for that sample equal to the maximum P_q and P_o values in the buffer respectively. The overall priority of the i^{th} sample is calculated as:

$$P(i) \doteq \chi \frac{P_q(i)}{\sum_k P_q(k)} + (1-\chi)\frac{P_o(i)}{\sum_k P_o(k)}, \ \chi \in [0,1] \tag{3.7}$$

Based on $P(i)$, we pick B (batch-size) samples and train the QRDQN and option-value network. After training, we update the P_q and P_o values only for the samples we draw from the buffer.

Such a sampling strategy induces a bias, hence we need a bias-correction term in our update. Just like Prioritized Experience Replay for DQN, we use $w(i)$, given in Equation (3.8) for bias correction of i^{th} sample,

$$w(i) \doteq \left(\frac{1}{N}\frac{1}{P(i)}\right)^{\beta_c} \tag{3.8}$$

where N denotes the number of samples in the buffer and β_c is a hyperparameter used to scale the correction.

Multi-step learning

In training QRDQN, we use the target value given by Equation (3.9) and for training option-value network, we use the target value given by Equation (3.10), for a sample $(s_t, a_t, [r_{t+1}, \ldots, r_{t+n}], s_{t+n}, o_t)$ from the replay buffer. In previous sections, we showed all equations with $n = 1$ for simplicity, but they can be easily extended as shown here.

$$\text{Huber-quantile loss target} \doteq \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n q_j(s_{t+n}, a^*; \theta^-) \tag{3.9}$$

$$\begin{aligned}\text{Option loss target} \doteq \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n (\beta Q_\Omega(s_{t+n}, \arg\max_o Q_\Omega(s_{t+n}, o; \psi); \psi^-) \\ +(1-\beta) Q_\Omega(s_{t+n}, o_t; \psi^-))\end{aligned} \tag{3.10}$$

3.2.3 Full algorithm

Deep-QUOTA algorithm for discrete action space is given in Algorithm 1 below.

Algorithm 1 Deep-QUOTA algorithm for discrete action space

Input:

ϵ: Random action selection probability
ϵ_Ω: Random option selection probability
β: option termination probability
$\{q_i\}_{i=1,\dots,N}$: quantile estimation parameterized by θ and θ^-
Q_Ω: option value parameterized by ψ and ψ^-
B: Batch-size
χ: weight for quantile-loss priority
n: horizon length in multi-step learning
F: Training frequency
T: Target and current network synchronization frequency
$\epsilon_1, \epsilon_2, \beta_c, \omega$: Priority experience replay parameters
Γ: Option to quantiles mapping function
α: Step-size

Output:

θ, ψ

1: **for** each time step t **do**
2: Observe the state s_t
3: Select an option o_t as:

$$o_t \leftarrow \begin{cases} o_{t-1} & ; \text{ with probability } 1-\beta \\ \text{random option} & ; \text{ with probability } \beta\epsilon_\Omega \\ \arg\max_{o\in\Omega} Q_\Omega(s_t, o; \psi) & ; \text{ with probability } \beta(1-\epsilon_\Omega) \end{cases}$$

4: Select action a_t as:

$$a_t \leftarrow \begin{cases} \text{random action} & ; \text{ with probability } \epsilon \\ \arg\max_{a\in\mathcal{A}} \frac{1}{K} \sum_{k\in\Gamma(o_t)} q_k(s_t, a; \theta) & ; \text{ with probability } 1-\epsilon \end{cases}$$

5: Execute a_t, get reward r_{t+1} and transit to next state, s_{t+1}
6: After n such steps, insert sample $i = (s_t, a_t, [r_{t+1}, \dots, r_{t+n}], s_{t+n}, o_t)$ in the experience buffer with $P_q(i) = \max_{i'\in\text{Buffer}} P_q(i')$ and $P_o(i) = \max_{i'\in\text{Buffer}} P_o(i')$

if $t \bmod F = 0$ **then**

Compute $P(i)$ for each sample in the buffer as:

$$P(i) = \chi \frac{P_q(i)}{\sum_k P_q(k)} + (1-\chi)\frac{P_o(i)}{\sum_k P_o(k)}$$

Sample a batch of size B based on distribution P

$\mathcal{L} \leftarrow 0$

$\mathcal{L}_\Omega \leftarrow 0$

for each sample $i = (s_t, a_t, [r_{t+1}, \ldots, r_{t+n}], s_{t+n}, o_t) \in$ Batch **do**

$o^* = \arg\max_{o'} Q_\Omega(s_{t+n}, o'; \psi)$

$a^* = \arg\max_a \frac{1}{K} \sum_{k \in \Gamma(o^*)} q_k(s_{t+n}, a; \theta)$

$y_{t,j} = \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n q_j(s_{t+n}, a^*; \theta^-),\ \forall j \in [N]$

$w(i) = \left(\frac{1}{N}\frac{1}{P(i)}\right)^{\beta_c}$

$\mathcal{L}(i) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \rho^{\kappa}_{\hat{\tau}_i}(y_{t,j} - q_i(s_t, a_t; \theta))$

$\mathcal{L} \leftarrow \mathcal{L} + w(i)\mathcal{L}(i)$

$P_q(i) \leftarrow |L(i)|^\omega + \epsilon_1$

$y^o_t = \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n (\beta Q_\Omega(s_{t+n}, o^*; \psi^-) + (1-\beta) Q_\Omega(s_{t+n}, o_t; \psi^-))$

$\mathcal{L}_\Omega(i) = \frac{1}{2}(y^o_t - Q_\Omega(s_t, o_t; \psi))^2$

$\mathcal{L}_\Omega = \mathcal{L}_\Omega + w(i)\mathcal{L}_\Omega(i)$

$P_o(i) \leftarrow |L_\Omega(i)|^\omega + \epsilon_2$

end for

$\theta \leftarrow \theta - \alpha \nabla_\theta \frac{\mathcal{L}}{B}$

$\psi \leftarrow \psi - \alpha \nabla_\psi \frac{\mathcal{L}_\Omega}{B}$

end if

if $t \bmod T = 0$ **then**

$\theta^- \leftarrow \theta$

$\psi^- \leftarrow \psi$

end if

end for

CHAPTER 4

PERFORMANCE STUDY

In this chapter, we will empirically study the benefit of Deep-QUOTA framework in three different MDPs which require deep exploration strategies for determining the optimal policy. We will also perform an ablation study of the improvements employed in Deep-QUOTA. Implementation of all the experiments are publicly available[1].

4.1 Optimistic Chain World

First, we evaluate our algorithm on a chain MDP (adapted from Zhang *et al.* [2018]) against QUOTA and QRDQN. The chain has N non-terminating states and two terminating states. At every non-terminating state, the agent has two possible actions, `UP` and `LEFT`. On choosing `UP`, the agent receives a reward of 0 and the trajectory terminates. On choosing left, the agent receives a reward sampled from $\mathcal{N}(0, 1)$ and transits to next non-terminating state to the left. In the N^{th} non-terminating state, `LEFT` gives a reward of $+10$ and the agent transits into a terminating state. The MDP is depicted in Figure 4.1 below. Zhang *et al.* [2018] use very small values of N (maximum value of 6), which we increase upto 30 in our experiments.

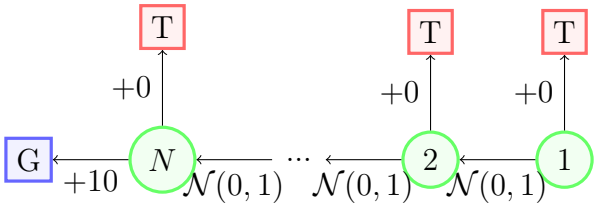

Figure 4.1: Chain MDP in which acting on higher quantiles leads to efficient exploration

[1] https://github.com/yogesh1q2w/Deep-QUOTA.git

4.1.1 Results

We evaluated the performance of Deep-QUOTA against QUOTA and QRDQN for chain sizes 5, 10, 15, 20, 25 and 30. We set number of quantiles, N to 3, with three options, one corresponding to each quantile for all experiments. We use two-fully connected layers of size 64 for state-representation network and add linear layers of appropriate sizes for QRDQN and Option-value network. We linearly decayed ϵ and ϵ_Ω from 1.0 to 0.0 over 10^5 training steps. β was chosen to be 0.0, leading to fixed choice of option over an entire episode, which was inspired from Bootstrapped DQN. The batch size was set to 64, χ was set to 0.5, ϵ_1 and ϵ_2 to 10^{-3}, and β_c was linearly increased from 0.6 to 1.0 over 10^5 training steps. The network was trained every 2 steps and synchronization frequency of network was set to 100. The entire network was trained using RMSProp optimizer with an initial learning rate of 10^{-4}.

The learning curves for chain length 5, 10, 20 and 30 are shown in Figure 4.2. We have provided the learning curves for remaining chain sizes in Appendix A. Figure 4.3 shows the fraction of trajectories taken by the agent during training that are optimal. The option values of start state of the Optimistic Chain MDP for various chain lengths are shown in Figure 4.4. Each plot represents an average over three independent runs of the experiment where standard errors are shown as shadow/bars.

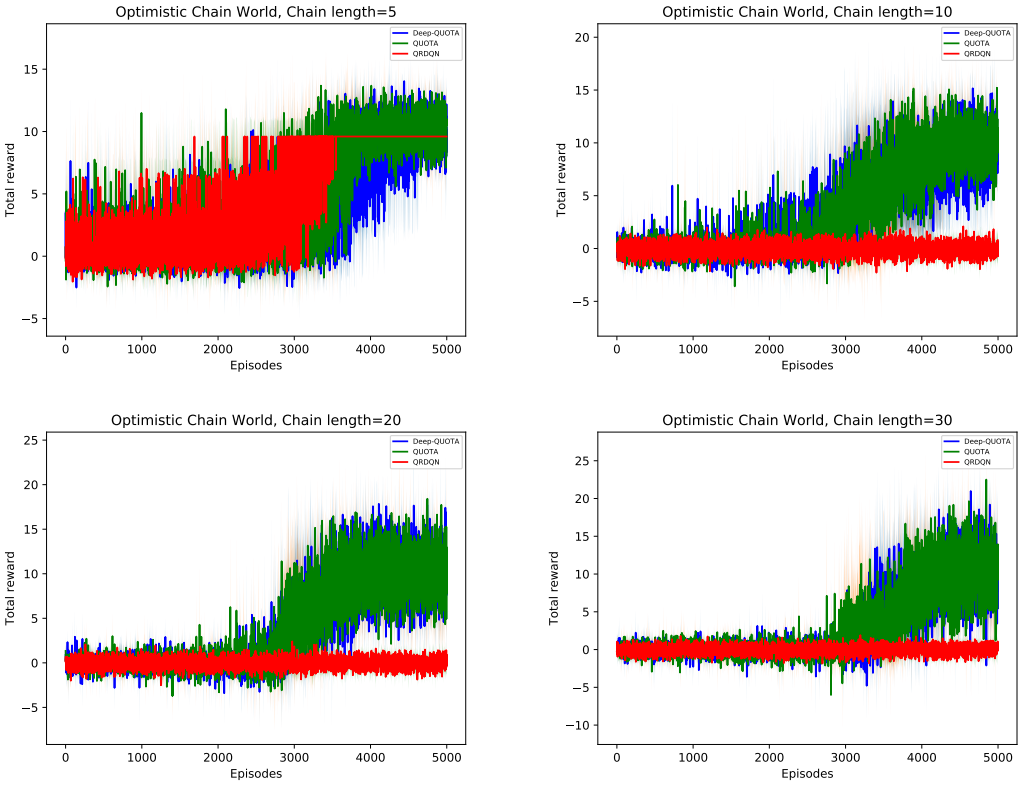

Figure 4.2: Learning curves for DeepQUOTA, QUOTA and QRDQN for Optimistic Chain World with chain sizes = 5, 10, 20 and 30. Each curve is an average over three independent runs.

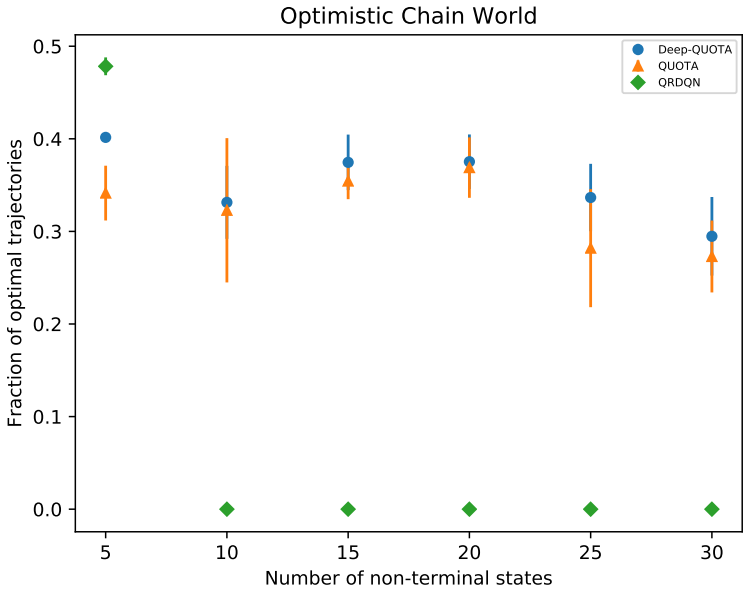

Figure 4.3: Fraction of optimal trajectories taken by each agent during training against number of non-terminal states in the Optimistic Chain MDP.

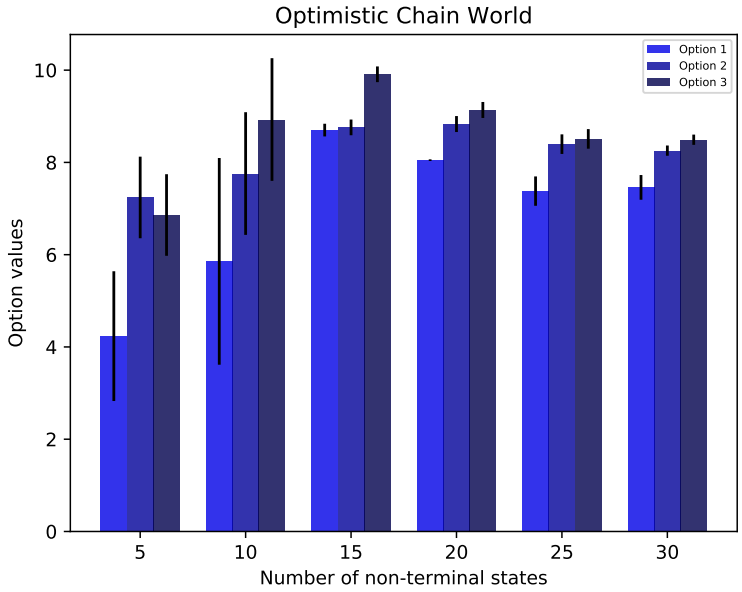

Figure 4.4: Option values of the start state of Optimistic Chain MDP. Option 1 corresponds to first quantile, Option 2 to second and Option 3 to third.

4.1.2 Analysis

We can observe in the learning curves that Deep-QUOTA is capable of discovering the optimal policy in all the chain settings. The performance of QRDQN is sub-optimal beyond chain length 5 but Deep-QUOTA and QUOTA consistently perform efficient exploration. In Figure 4.3, we observe that Deep-QUOTA discovers the optimal trajectory more frequently than QUOTA. For chain length 5, QRDQN gives the best performance, but breaks down as chain length is increased.

Figure 4.4 shows the option value for the Deep-QUOTA agent at the end of training. For chain length 5, we observe that Deep-QUOTA is not able to learn the correct option values but for higher chain lengths, the option values are consistent with our expectations. Option 3, which corresponds to the highest quantile, has the highest option value, which shows that Deep-QUOTA agent adjusts its option values to reflect the assumptions on returns needed for efficient exploration.

4.2 Pessimistic Chain World

This Chain MDP is very similar to the one described in Section 4.1, except that the rewards are modified to make a pessimistic assumption on the return values leading to efficient exploration. In this MDP, executing `UP` in a non-terminal state gives a reward sampled from $\mathcal{N}(0, 1)$ and `LEFT` gives a reward of -0.1 except in the N^{th} state, where it gives a reward of $+10$. The MDP is shown in Figure 4.5.

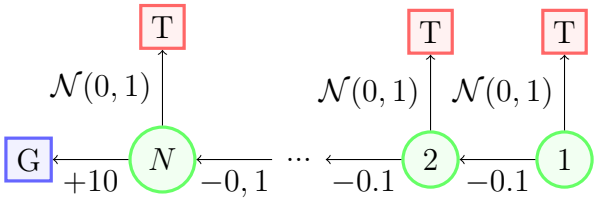

Figure 4.5: Chain MDP in which acting on lower quantiles leads to efficient exploration

4.2.1 Results

The evaluation of Deep-QUOTA in this environment was done on the same lines as Optimistic Chain World. All the experimental settings and hyperparameters were replicated from Section 4.1.1. The learning curves for chain length 5, 10, 20 and 30 are shown in Figure 4.6 and the remaining are given in Appendix A. The fraction of optimal trajectories in training are shown in Figure 4.7. The option values of the start state for various chain lengths are shown in Figure 4.8.

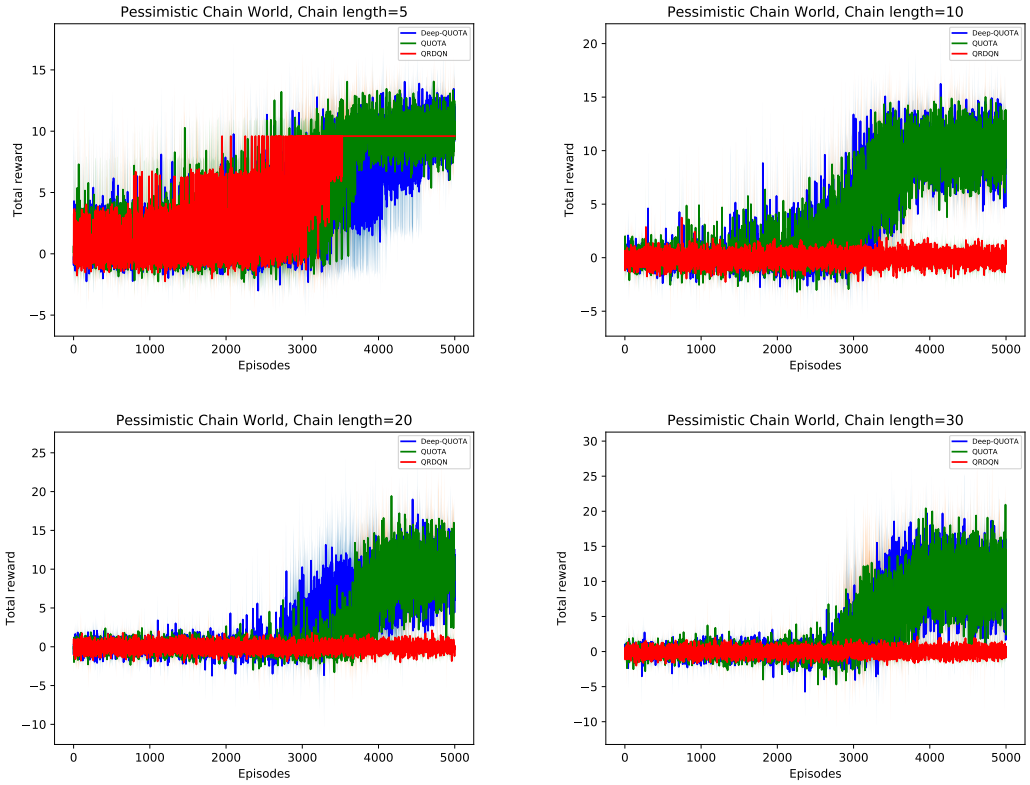

Figure 4.6: Learning curves for DeepQUOTA, QUOTA and QRDQN for Pessimistic Chain World with chain sizes = 5, 10, 20 and 30. Each curve is an average over three independent runs.

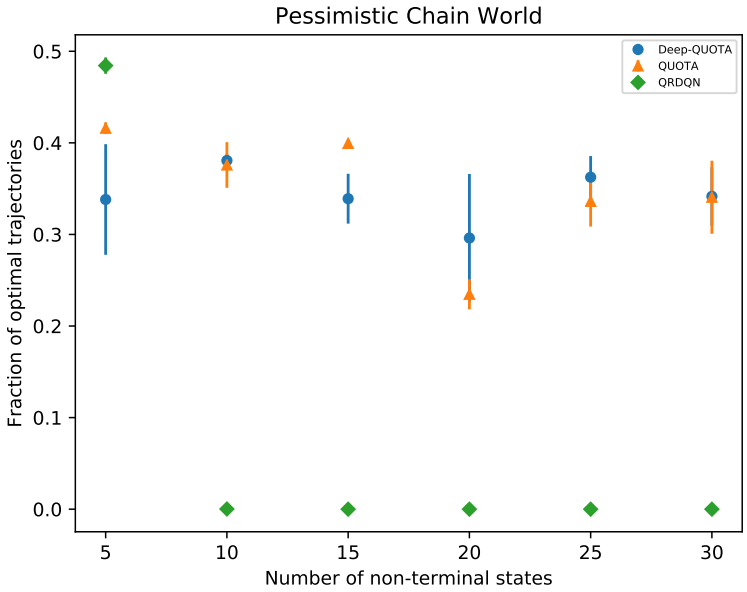

Figure 4.7: Fraction of optimal trajectories taken by each agent during training against number of non-terminal states in the Pessimistic Chain MDP.

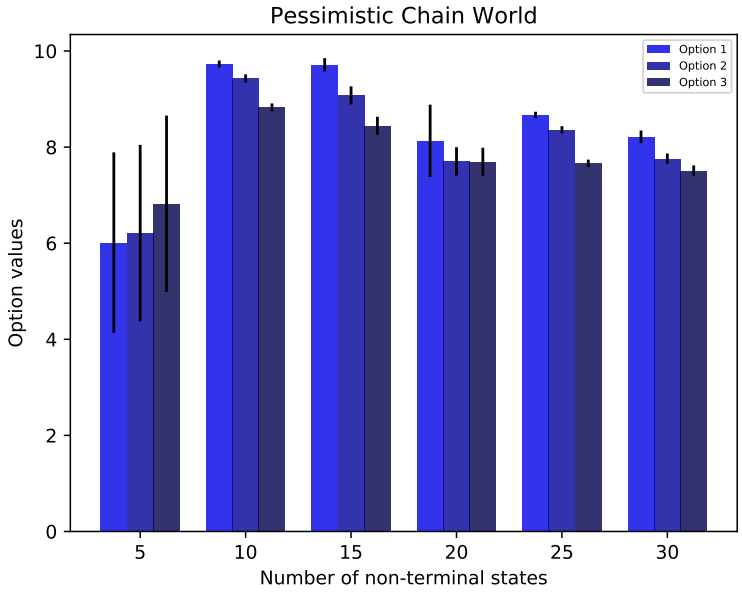

Figure 4.8: Option values of the start state of Pessimistic Chain MDP. Option 1 corresponds to first quantile, Option 2 to second and Option 3 to third.

4.2.2 Analysis

The learning curves for Pessimistic Chain World show very similar trends as the Optimistic one. For chain length 20, we observe a steeper increase in total reward for Deep-QUOTA than QUOTA. For all chain lengths, the performance of Deep-QUOTA is consistently competitive to QUOTA. In Figure 4.7, we observe that Deep-QUOTA either has higher or almost equal fraction of optimal trajectories as QUOTA, except for chain length 5 and 15. Figure 4.8 shows that Option 1, corresponding to the first quantile, has the highest option value for all chain lengths except 5, which is consistent with the pessimistic assumption needed for efficient exploration. This confirms that Deep-QUOTA agent adaptively changes its option values to suit the environment.

4.3 Montezuma's Grid World

This environment is inspired from one of the most challenging games in Arcade Learning Environment by Bellemare *et al.* [2013] : *Montezuma's Revenge*. It is difficult to solve due to its extremely sparse reward function. Most of the rewards

obtained by the agent is 0, which makes the feedback difficult to propagate through the long trajectories taken by the agent. In our environment, we have a four room grid world, shown in Figure 4.9, where $(1, 1)$ is the start state for every trajectory and possible actions are `LEFT, RIGHT, UP` and `DOWN`. All transitions are deterministic and reward for all transitions is 0, except for transitioning into state $(20, 1)$ or $(1, 20)$ which gives a reward of $+4$, or transitioning into state $(20, 20)$ which gives a reward of $+80$.

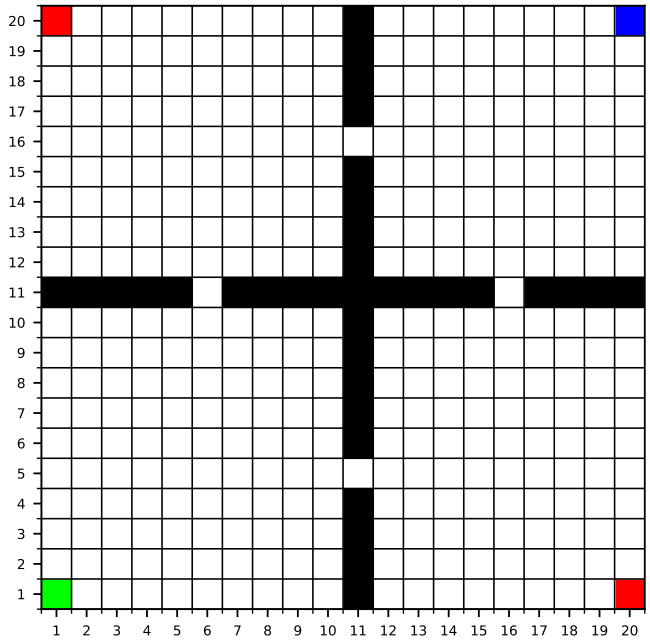

Figure 4.9: Montezuma's Grid World: A sparse reward four-room grid world. The grids colored black are blocked states, colored green is the start state, colored red are the sub-optimal terminating states and colored blue is the optimal terminating state.

4.3.1 Results

We evaluated the performance of Deep-QUOTA against QUOTA and QRDQN over three independent runs. We set the number of quantiles, N to 15, with 5 options, $\Gamma(i) = \{3i - 2, 3i - 1, 3i\}$, $\forall i \in [5]$, for all experiments. We use three fully connected layers of size 64, 64 and 32 respectively for state representation and add linear layers of appropriate sizes for QRDQN and Option-value network. ϵ and ϵ_Ω was exponentially decayed from 1.0 to 0.01 over 6×10^5 training steps.

Here, we found $\beta = 0.001$ better than our previous setting of $\beta = 0$. The batch size was kept 64, χ was chosen from $\{0.4, 0.5, 0.6\}$, among which we found 0.6 to give the best performance. ϵ_1 and ϵ_2 was kept 10^{-3}, β_c was linearly increased from 0.6 to 1.0 over 6×10^5 training steps. The network was trained every 10 steps and synchronization frequency of network was set to 100. The entire network was trained using RMSProp optimizer with an initial learning rate of 10^{-4}.

The learning curves for each agent is shown in Figure 4.10. The fraction of optimal trajectories taken by each agent during training is shown in Figure 4.11. The best option for every grid in the Montezuma Grid World at the end of training, calculated with option values averaged across three independent runs, is shown in Figure 4.12. Each plot represents an average over three independent runs of the experiment where standard errors are shown as shadow/bars.

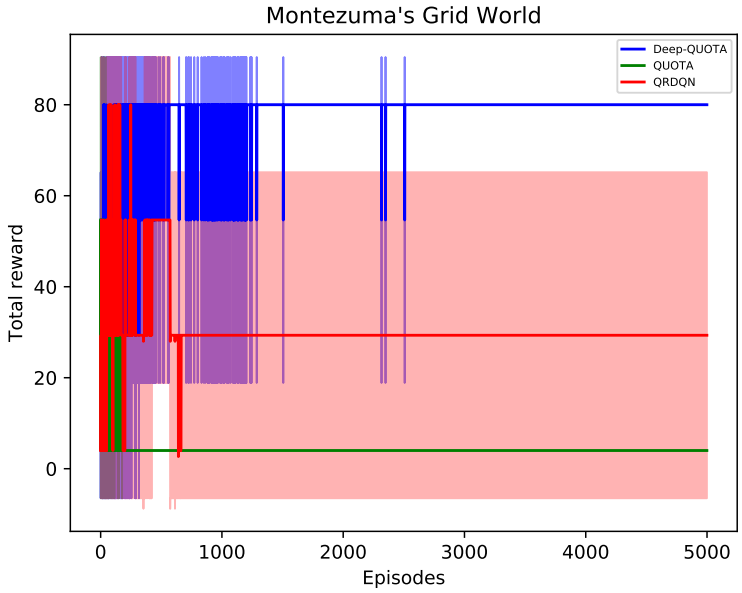

Figure 4.10: Learning curves for Deep-QUOTA, QUOTA and QRDQN in the Montezuma's Grid World. Each curve represents an average over 3 independent runs.

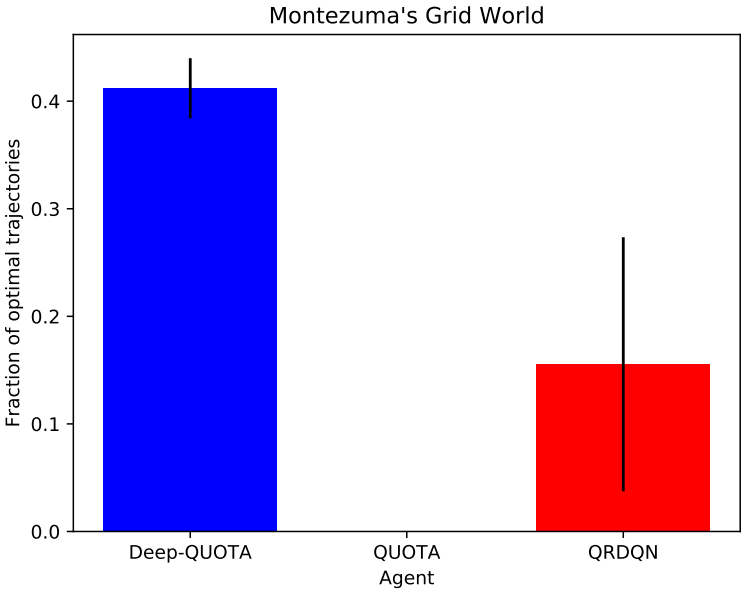

Figure 4.11: Fraction of optimal trajectories taken by each agent during training in the Montezuma's Grid World

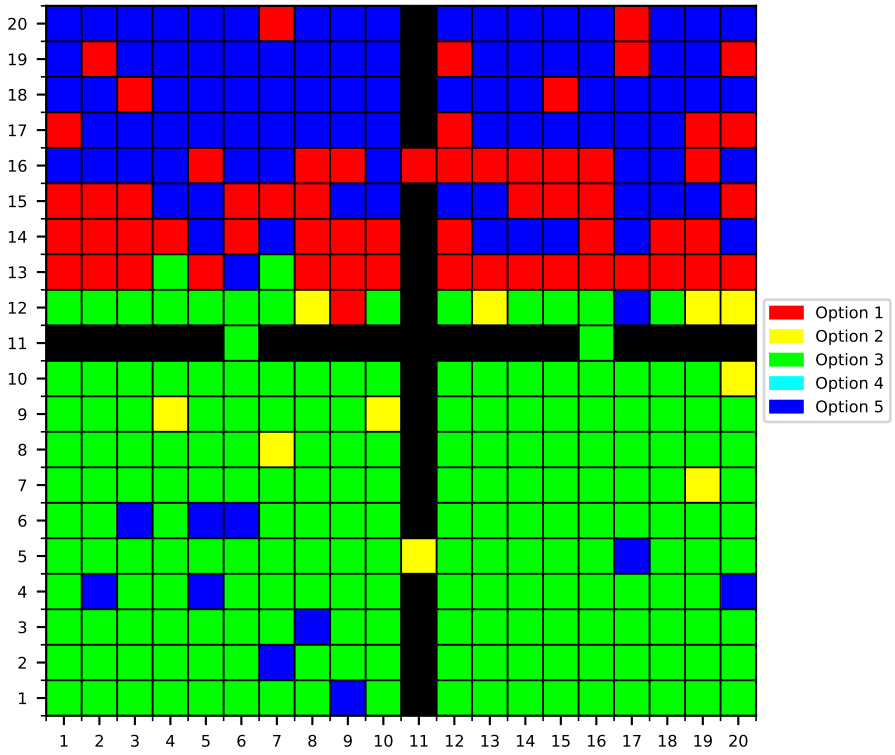

Figure 4.12: Best option in each grid of the Montezuma's Grid World. The optimal options are calculated with option values averaged across three independent runs.

4.3.2 Analysis

As evident in Figure 4.10, Deep-QUOTA outperforms QUOTA and QRDQN by a significant margin. While QUOTA does not converge to the optimal policy even at the end of 5000 episodes, one run of QRDQN discovers the optimal policy, thus giving higher performance than QUOTA. However, the shadowed error for QRDQN is very large which shows that the agent converged due to some rare favorable initialization. Deep-QUOTA, on the other hand, consistently converges to the optimal policy within 2500 episodes in all the runs. Figure 4.11 shows that around 40% of trajectories taken by Deep-QUOTA are optimal, which is significantly higher than QUOTA and QRDQN.

Figure 4.12 yields some interesting observations. Most of the bottom grids have Option 3 (close to the median) as the optimal option. This might be due to overpopulating the buffer with frequent zero reward transitions obtained in these states, which makes it difficult to propagate the final positive reward signal to these states. The grids in top two rooms recommend a diverse set of options, which might be due to frequent transitions of the agent into these states, leading to more training of option values here. The blue grids near the optimal and sub-optimal states recommend acting on highest quantile, which is rational, as in these states, the agent is close to a positive reward. The strip of red states, recommending acting on lower quantiles, are the ones which prevent the agent from running into the sub-optimal corner state. This is because the lower quantiles of action leading to sub-optimal state would be lesser than the lower quantiles of action leading to the optimal state. This makes the actions recommended in the red states push the agent to the optimal state.

4.4 Ablation Study of Improvements

We have employed three improvements in the Deep-QUOTA algorithm: Prioritized Experience Replay, Double Q-Learning and Multi-step learning. In this section, we will demonstrate the benefit obtained from each of them through an ablation study on Montezuma's Grid World. We choose this environment because Deep-QUOTA demonstrates a clear improvement in performance in it, thus making the

study of its ablations more systematic.

4.4.1 Results

We evaluate the Deep-QUOTA agent against three of its variants. In the first variant, we remove the Multi-step learning component and set the horizon length, n to 1. In the second variant, we remove the Double Q-Learning component, and choose the target values for Huber quantile loss and Option value loss as given in Equation 4.1 and 4.2 respectively.

$$\begin{aligned} \text{Huber-quantile loss target} &\doteq \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n q_j(s_{t+n}, a^*; \theta^-) \\ a^* \doteq \arg\max_a \frac{1}{K} \sum_{k \in \Gamma(o)} q_k(s_{t+1}, a; \theta^-), \;\; o &\doteq \arg\max_{o'} Q_\Omega(s_{t+1}, o'; \psi^-) \end{aligned} \tag{4.1}$$

$$\begin{aligned} \text{Option loss target} \doteq \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n (\beta \max_o Q_\Omega(s_{t+n}, o; \psi^-) \\ + (1-\beta) Q_\Omega(s_{t+n}, o_t; \psi^-)) \end{aligned} \tag{4.2}$$

In the third variant, we remove the Prioritized Experience Replay component and instead sample uniformly randomly from the buffer. Figure 4.13 shows the learning curves for each of these ablations. In Figure 4.14, we show the fraction of optimal trajectories taken by the agent during training. We again note that each plot represents an average over three independent runs of the experiment where standard errors are shown as shadow/bars.

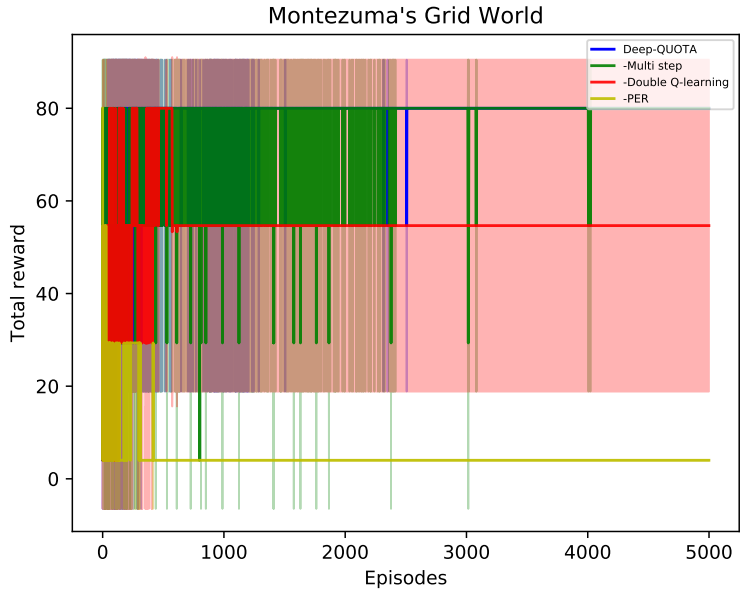

Figure 4.13: Learning curves for the ablations of Deep-QUOTA in the Montezuma's Grid World. The ablations considered in this figure are Deep-QUOTA with no Multi-step learning, no Double Q-learning and no Prioritized Experience Replay respectively.

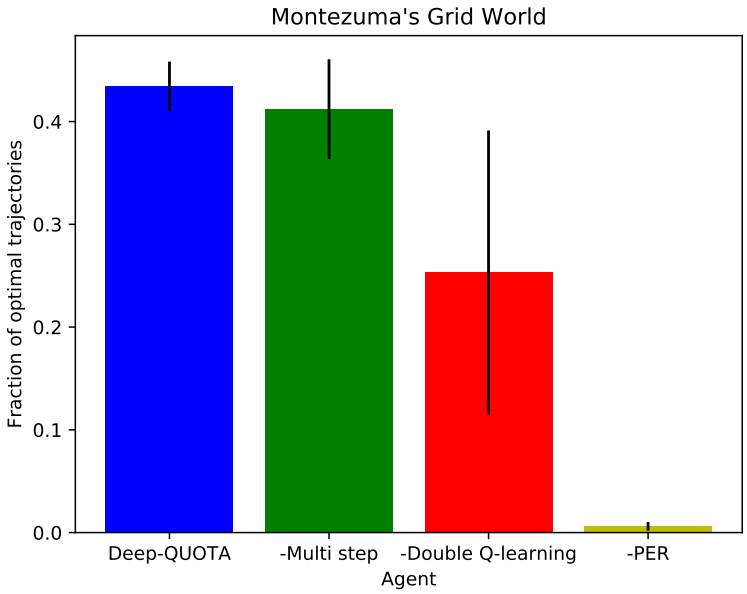

Figure 4.14: Fraction of optimal trajectories taken by each ablation of Deep-QUOTA during training in the Montezuma's Grid World

4.4.2 Analysis

From the learning curves in Figure 4.13, we observe that Deep-QUOTA and Deep-QUOTA without multi-step learning are the only two agents which are able to

determine the optimal policy in all three runs. However, the multi-step ablation takes slightly more steps than Deep-QUOTA. Deep-QUOTA without Double Q-Learning is able to determine the optimal policy only in two of its runs, thus showing that Double Q-Learning is an important improvement. The most significant improvement employed in Deep-QUOTA is very evidently Prioritized Experience Replay buffer, without which, none of the runs converge to the optimal policy.

In Figure 4.14 also, we observe that the Multi-step learning ablation reduces the fraction of optimal trajectories slightly. However, removing Double Q-learning causes a significant dip in this fraction. Without prioritized experience replay, the fraction of optimal trajectories is very low which shows that the biased sampling based on quantile loss priority and option loss priority is an important aspect of efficient training and optimal solution discovery in Deep-QUOTA.

4.5 Summary

In the previous sections, we looked at various environment settings in which Deep-QUOTA agent performed either at par or significantly better than its counterparts. In the two Chain MDPs, we have demonstrated that Deep-QUOTA adapts its option values based on the environment setting, to explore efficiently and obtain high returns. We also find that Deep-QUOTA provides a significant improvement in performance in sparse reward setting like Montezuma's Grid World. With our ablation study, we have quantified the importance of each component of Deep-QUOTA empirically.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this chapter, we will conclude this work by summarizing our problem, solution approach and experimental findings. We will then provide directions in which this work can be extended as future work.

5.1 Conclusion

We started with the problem of *Explore-Exploit dilemma* faced by RL agents. To tackle it, we investigated various methods in which deep exploration strategies can be generated by the agent. One such way is using Distributional RL in Options framework, proposed by Zhang *et al.* [2018] in QUOTA. However, we established that QUOTA is not a purely hierarchical setup as its QRDQN training completely disregards the hierarchy maintained by the agent in the form of options. To solve this, we proposed using Hierarchical QRDQN, which uses the action recommended by the best option in the subsequent state to construct its target for training. Thus, it ties the Distributional RL component to the Hierarchical component in QUOTA.

Many improvements to the original DQN algorithm (Mnih *et al.* [2015]) have been suggested and fruitfully combined to aid learning. We adpated Double Q-Learning, Prioritized Experience Replay and Multi-step learning to our framework and developed an agent which performs deep exploration by fruitfully combining each of these. We evaluated our agent, Deep-QUOTA against QUOTA and QRDQN on three challenging environments- Optimistic Chain World, Pessimistic Chain World and Montezuma's Grid World. In both Chain World, our agent either performs at par or better than QUOTA and QRDQN for long chain lengths. In Montezuma's Grid World, our agent provides a significant improvement in performance through efficient exploration. In ablation study, we empirically es-

tablished that Prioritized Experience Replay is the most crucial component of Deep-QUOTA, followed by Double Q-learning.

5.2 Future Work

In this work, we have only proposed a heuristic to generate efficient exploratory policies. The problem of efficient exploration is far from solved. One possible extension of this work could involve maintaining the quantile approximation for return distribution of options. This would require developing a QRDQN equivalent for option return function. We can notice here that defining option return function itself would require considering other complications like intra-option policy and option termination function.

We can also take a variational approach by maintaining a distribution over parameters of Deep-QUOTA. Then, we can optimize our loss functions over the parameters of this distribution. We can also maintain the Option-value network with multiple heads, similar to Bootstrapped-DQN. This would be a Thompson sampling based approach, but at option level. Instead of QRDQN component in Deep-QUOTA, we can also use full parameterization quantile approximation (Yang *et al.* [2019]). All these are possible directions to explore, which we propose as future work.

APPENDIX A

APPENDIX

Figure A.1 below shows the learning curve for Optimistic Chain World for chain length 15 and 25.

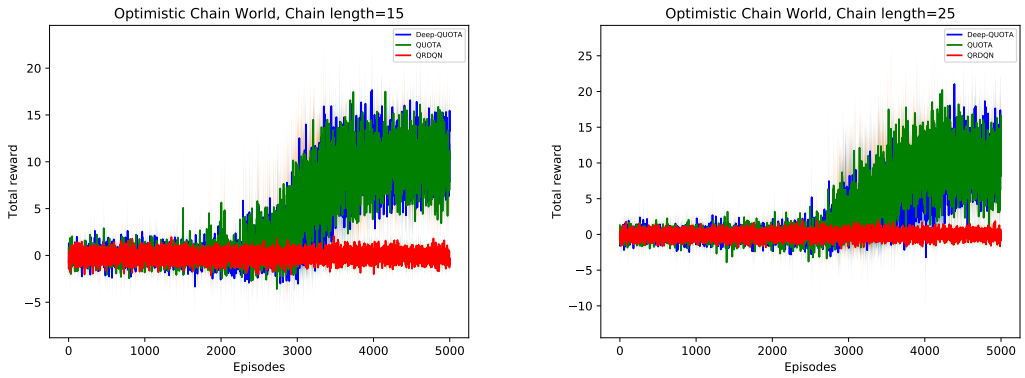

Figure A.1: Learning curves for DeepQUOTA, QUOTA and QRDQN for Optimistic Chain World with chain sizes = 15 and 25. Each curve is an average over three independent runs of the agent.

Figure A.2 below shows the learning curve for Pessimistic Chain World for chain length 15 and 25.

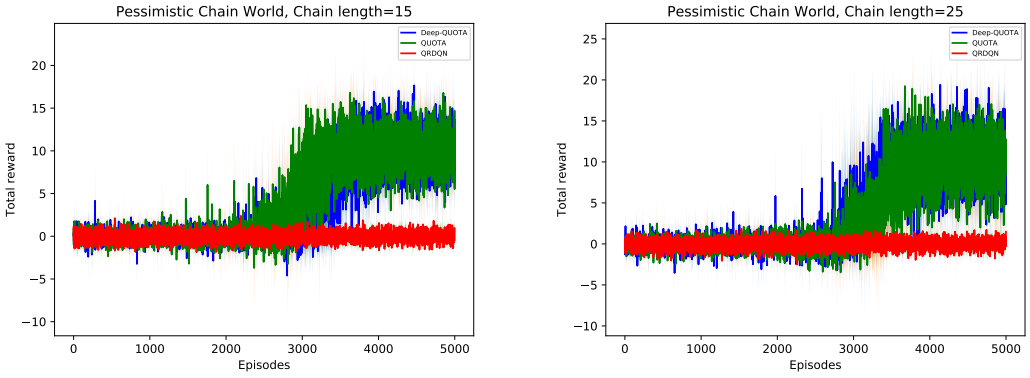

Figure A.2: Learning curves for DeepQUOTA, QUOTA and QRDQN for Pessimistic Chain World with chain sizes = 15 and 25. Each curve is an average over three independent runs of the agent.

REFERENCES

1. **Barth-Maron, G.**, **M. W. Hoffman**, **D. Budden**, **W. Dabney**, **D. Horgan**, **D. TB**, **A. Muldal**, **N. Heess**, and **T. Lillicrap** (2018). Distributed distributional deterministic policy gradients.

2. **Bellemare, M. G.**, **W. Dabney**, and **R. Munos** (2017). A distributional perspective on reinforcement learning.

3. **Bellemare, M. G.**, **Y. Naddaf**, **J. Veness**, and **M. Bowling** (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, **47**, 253–279. ISSN 1076-9757. URL `http://dx.doi.org/10.1613/jair.3912`.

4. **Bellman, R. E.**, *Dynamic Programming*. Dover Publications, Inc., USA, 2003. ISBN 0486428095.

5. **Dabney, W.**, **M. Rowland**, **M. G. Bellemare**, and **R. Munos** (2017). Distributional reinforcement learning with quantile regression. *CoRR*, **abs/1710.10044**. URL `http://arxiv.org/abs/1710.10044`.

6. **Gruslys, A.**, **W. Dabney**, **M. G. Azar**, **B. Piot**, **M. Bellemare**, and **R. Munos** (2017). The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning.

7. **Hessel, M.**, **J. Modayil**, **H. van Hasselt**, **T. Schaul**, **G. Ostrovski**, **W. Dabney**, **D. Horgan**, **B. Piot**, **M. Azar**, and **D. Silver** (2017). Rainbow: Combining improvements in deep reinforcement learning.

8. **Mavrin, B.**, **S. Zhang**, **H. Yao**, **L. Kong**, **K. Wu**, and **Y. Yu** (2019). Distributional reinforcement learning for efficient exploration.

9. **Mnih, V.**, **K. Kavukcuoglu**, **D. Silver**, **A. A. Rusu**, **J. Veness**, **M. G. Bellemare**, **A. Graves**, **M. Riedmiller**, **A. K. Fidjeland**, **G. Ostrovski**, **S. Petersen**, **C. Beattie**, **A. Sadik**, **I. Antonoglou**, **H. King**, **D. Kumaran**, **D. Wierstra**, **S. Legg**, and **D. Hassabis** (2015). Human-level control through deep reinforcement learning. *Nature*, **518**(7540), 529–533. ISSN 00280836. URL `http://dx.doi.org/10.1038/nature14236`.

10. **Nikolov, N.**, **J. Kirschner**, **F. Berkenkamp**, and **A. Krause** (2018). Information-directed exploration for deep reinforcement learning.

11. **Osband, I.**, **C. Blundell**, **A. Pritzel**, and **B. V. Roy** (2016). Deep exploration via bootstrapped DQN. *CoRR*, **abs/1602.04621**. URL `http://arxiv.org/abs/1602.04621`.

12. **Schaul, T.**, **J. Quan**, **I. Antonoglou**, and **D. Silver** (2015). Prioritized experience replay. URL `http://arxiv.org/abs/1511.05952`. Cite arxiv:1511.05952Comment: Published at ICLR 2016.

13. **Sutton, R. S.** (1988). Learning to predict by the methods of temporal differences. *Mach. Learn.*, **3**(1), 9–44. ISSN 0885-6125. URL https://doi.org/10.1023/A:1022633531479.

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

15. **Sutton, R. S.**, **D. Precup**, and **S. Singh** (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, **112**(1–2), 181–211. ISSN 0004-3702. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

16. **Tang, Y.** and **S. Agrawal** (2018). Exploration by distributional reinforcement learning.

17. **van Hasselt, H.**, **A. Guez**, and **D. Silver** (2015). Deep reinforcement learning with double q-learning. *CoRR*, **abs/1509.06461**. URL http://arxiv.org/abs/1509.06461.

18. **Watkins, C. J. C. H.** and **P. Dayan**, Q-learning. *In Machine Learning*. 1992.

19. **Yang, D.**, **L. Zhao**, **Z. Lin**, **T. Qin**, **J. Bian**, and **T. Liu** (2019). Fully parameterized quantile function for distributional reinforcement learning.

20. **Zhang, S.**, **B. Mavrin**, **L. Kong**, **B. Liu**, and **H. Yao** (2018). Quota: The quantile option architecture for reinforcement learning.