Benchmarking the Performance of Q-Learning and Actor-Critic Agents on the Control of Water Systems

D. Welham
867884

Submitted to Swansea University in partial fulfilment
of the requirements for the Degree of Master of Science

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

June - September 2023

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This work is the result of my own independent study/investigations, except where otherwise stated. Other sources are clearly acknowledged by giving explicit references. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure of this work and the degree examination as a whole.

Signed ..... (candidate)

Date .....

Statement 2

I hereby give my consent for my work, if accepted, to be archived and available for reference use, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Abstract

Control of water systems is an essential and safety-critical task. Recent innovations in machine learning have been applied to water systems [1], but due to the black-box nature of many of these algorithms, actual deployment in practice is scarce. This paper applies the explainable machine learning architecture of Q-learning to water systems in an attempt to make systems which not only make good decisions, but justify those decisions in a way that is understandable to domain experts. An actor-critic network [2] is also implemented to compare performance on these problems.

Several simulated environments of varying levels of complexity are implemented to test how the agents perform on increasingly realistic versions of the water tank control problem. Knowledge from domain experts is used to ensure that the simulated environments resemble actual deployment environments, but some approximations are made to make the problem computationally tractable.

This paper demonstrates cases where Q-learning can be an effective tool for control of water systems, and also highlights cases where a simple Q-learning agent would make bad decisions that result in catastrophic failure. Where several agents are capable of performing well on the same task, this paper benchmarks them and evaluates how appropriate each agent may be for that task.

Contents

Chapter 1

Introduction

The purpose of this paper is to investigate whether Q-learning agents can perform well on various tasks for the control of water systems. Where Q-learning can perform well on a task, it is preferable to a non-explainable architecture like actor-critic networks because its q-values necessarily reflect actual experience of the action-state pair, whereas an actor-critic network may confidently assign a quality to a particular action to a particular state even if the actor-critic network has never encountered that actor-state pair before and does not actually have a good understanding of whether that action is good or not. The Q-learning algorithm's training process is also intuitively easier for a domain expert to understand since the agent effectively "memorises" how good an action is from a particular state, which matches roughly how humans think about some problems. Conversely, neural networks are trained by some optimization process, typically stochastic gradient descent [3], in a way that is opaque and very heavy on mathematics, making the process hard for domain experts to follow.

This paper establishes the background theory behind reinforcement learning [4], actor-critic networks, and Q-learning, then implements several test environments in which to benchmark the performances of each algorithm. Success is defined here as the ability to fill a tank to half-full and maintain the height of the water at this level over time. A number of problems are implemented including a simple tank with constant inflow and outflow, a gravity-drained tank in which the outflow is proportional to the square-root of the height of the water in the tank, a tank where the outflow pipe is insufficient to achieve full drainage, and a bi-modal demand problem aiming to emulate realistic data for the

demand of water usage from customers. Various edge-cases were also tested, including tanks of different shapes and different hyper-parameters for the algorithms.

It was concluded that Q-learning is an appropriate tool for the control of water systems in several different contexts. Recommendations for further work based upon this paper are also made.

Chapter 2

Background

2.1 Reinforcement Learning

Reinforcement learning is a subset of the field of machine learning. In the broadest possible sense, machine learning is about taking some class of inputs "$\mathcal{I}$" and using some algorithm "$\mathcal{A}$" to produce some class of outputs "$\mathcal{O}$", where the exact way that $\mathcal{A}$ behaves is not explicitly coded by the programmer, but rather is learned by some training process, "$\mathcal{T}$", typically by tuning the weights, "ω", of $\mathcal{A}$. That is:

$$\mathcal{A}_\omega : \mathcal{I} \rightarrow \mathcal{O} \tag{2.1}$$

and:

$$\mathcal{T} : \mathcal{A}_i \rightarrow \mathcal{A}_j \mid \mathcal{A}_j \succ \mathcal{A}_i \tag{2.2}$$

-:where " $\succ$ " is the preference relation, that is, we prefer $\mathcal{A}_j$ to $\mathcal{A}_i$.

In the case of reinforcement learning, the class of inputs is some state, $\mathcal{S}$ of the environment, and the class of outputs is some action, "**A**", which the algorithm (now the "agent") has chosen to take. We now have:

$$\mathcal{A}_\omega : \mathcal{S} \rightarrow \mathbf{A} \tag{2.3}$$

Where **A** affects the next state of the environment. The process of deciding which version of the agent is automated by setting some reward function $\mathcal{R}$ where agents which achieve higher reward are considered to be preferable.

2.1.1 Policy-Based Vs. Value-Based Reinforcement Learning

The two main reinforcement learning paradigms are policy-based and value-based learning. Policy-based learning attempts to find a policy, "π" [5], that maps states to actions directly [6]. That is:

$$\pi : \mathcal{S} \rightarrow \mathbf{A} \tag{2.4}$$

Whereas value-based learning [7] has a state-value function, "V(s)", and an action-value function, "Q(s, a)", which it uses to decide which states are desirable and to navigate towards those states. The value-based agent does (implicitly) have some policy "π" which it follows, but its explicit considerations are only V(s) and Q(s, a). V(s) represents the expected cumulative reward of being in state $s \in \mathcal{S}$ and then following policy π, whereas Q(s, a) is the expected cumulative reward of being in state $s \in \mathcal{S}$ and taking action $a \in \mathbf{A}$ and then following policy π.

Q-learning is a value-based method as it attempts to assign q-values to pairs of actions and states, where these q-values are designed to approximate Q(s, a). The critic part of the actor-critic network is also value-based, as it is given pairs of states and actions and attempts to approximate how good it is to take that action from that state.

The actor part of the actor-critic network is policy-based, it simply learns that given state $s \in \mathcal{S}$ it must take action $a \in \mathbf{A}$.

2.2 Environment

Every environment consisted of a tank model, a model for inflow and outflow, and a state. The state always contained at least the water's height in the tank and the volume of water in the tank, and in problems where the outflow/inflow vary over time, the state also contains the current time. The volume of water in the tank was only used for the environment simulation to calculate changes in height more conveniently, the agents never access the volume directly, since real-world systems typically cannot access the volume directly.

2.3 Inflow/Outflow Models

2.3.1 Basic Problem

The simplest model used in this paper is a cylindrical tank where the inflow and outflow are both constant: 0.5 and 1 units respectively. This is a useful base for the project to expand upon, but is not a realistic physical system as water doesn't typically just flow out from a tank at a constant rate. We call this "the basic problem".

2.3.2 Gravity Problem

A consequence of the Bernoulli Equation [8] is that the outflow from a gravity-drained water tank is proportional to the square root of the height of the water in the tank [9]. If the units are in metres and the liquid in the tank is pure water then the outflow is exactly $\sqrt{2gh}$ where g is the gravitational constant and h is the height, but in order to not lose generality this paper assumes that the outflow is:

$$\delta v_{out} = -\psi\sqrt{h} \tag{2.5}$$

-:where δv_{out} is the change in the volume due to the outflow, ψ is some empirical constant, and "h" is the height of the water in the tank. If we assume that gravity is the limiting factor in allowing water out of the tank and that the inflow remains constant, then we have a time-inhomogenous Markov process that the agents must learn to control, which we call the "Gravity Problem".

2.3.3 Capacity Problem

Real water systems cannot allow an arbitrarily huge amount of water to flow out of the tank at any given moment, as the outflow must pass through pipes which have a maximum capacity. Therefore, we define the "Capacity Problem" in which a gravity-drained tank is also limited by the maximum outflow that can pass through the tank, which is some constant. Unless otherwise stated, the constant used is 1.2 units.

2.3.4 Demand Problem

The tanks being simulated here are used to store water for customers, but customer demand is not constant over time: according to domain experts, there is typically a spike

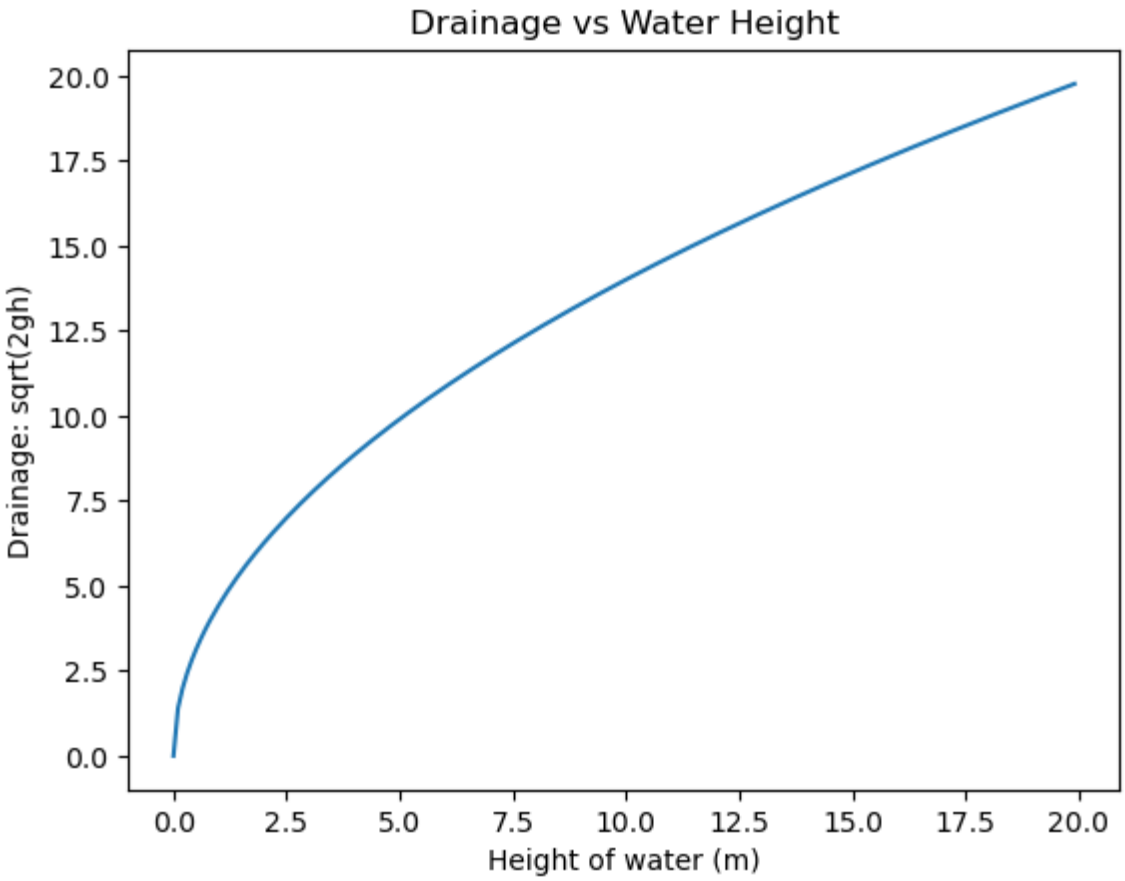

Figure 2.1: Outflow vs height (in metres) for pure water in a gravity-drained tank

in water usage at around 08:00 and 20:00, since many people are either getting ready for the day or are preparing to go to bed, and so do morning or evening routines that involve showering, brushing teeth etc. Therefore, we extend the model used in the capacity problem so that the demand of the customers varies over time. Now the agents have access to the current time to the nearest 0.01 of an hour, and the demand is no longer constant but is some stochastic variable with peaks at around 08:00 and 20:00. The tank remain gravity-drained and limited by a capacity outflow, so at each time-step the outflow is either the maximum outflow that can happen due to gravity or the maximum capacity of the pipe or the total demand from customers, whichever is smallest.

2.4 Tank Models

The tanks are modelled as circular frustums. When the base area “S" is the same as the top area “s" then the frustum is a vertical cylinder. The volume of water contained in a circular frustum of base area S and top area s with height “h" is given by:

$$V = \frac{1}{3} \times \pi \times h \times (S + s + \sqrt{S \times s}) \tag{2.6}$$

And therefore the change in height δh of the water given a change in volume δv is given by:

$$\delta h = \frac{3\delta v}{\pi \times (S + s + \sqrt{S + s})} \tag{2.7}$$

In order to check generality, we also implemented rectangular frustums where the equivalent formulae are:

$$V = \frac{1}{3} \times h \times (S + s + \sqrt{S \times s}) \tag{2.8}$$

And:

$$\delta h = \frac{3\delta v}{(S + s + \sqrt{S + s})} \tag{2.9}$$

Except where otherwise stated, the tanks used were the circular frustums.

2.5 Actions

For each problem the agents were given one of two sets of actions: they either had “binary actions" where the valve must either be fully open or fully closed, or “decimal actions" where the valve can be set so that some fraction of the possible inflow is allowed, in increments of one-tenth.

For the purposes of the simulation, it is assumed that whichever action the agent picks, the valve can immediately be set to that value. Although this is not strictly true in physical systems since it takes some time for the valve to move to the new position, it should be a reasonable assumption given that valves can be moved very quickly, and the use of

Q-learning necessitates that time be discretised anyway.

Although in principle there will exist some physical configuration of a single valve such that it lets in one-tenth of the flow that it would let in if it were fully open (since the amount of inflow strictly increases with the openness of the valve, which is itself continuous), it may be difficult to calculate how open the valve should physically be in order to achieve this. However, this should not matter in practice since it is equivalent to having ten valves and, if an inflow of 0.3 is required, opening 3 of them.

2.6 Utility Functions

The goal of each problem is for the tank to be half-full (in terms of height). This allows the tank to have the optimal trade-off between being unlikely to overflow (i.e. exceed its maximum capacity) and being unlikely to become fully empty and therefore be unable to meet the demands of consumers. Therefore, a utility function, $U(s)$, was constructed for the value of any particular state of the height of the water relative to the maximum height of the tank, which is:-

$$U(s) = 100 - (Error(s) + Overflow(s)) \tag{2.10}$$

-:where Error(s) is the square of the difference between the water height and half the tank height, that is:-

$$Error(s) = (h_w - \frac{h_t}{2})^2 \tag{2.11}$$

-:and Overflow(s) is given by the square of the height of water exceeding the maximum tank height, that is:

$$Overflow(s) = (h_w - h_t)^2 \tag{2.12}$$

-:if $h_w > h_t$, and 0 otherwise.

U(s) defines a preference order of possible world states in which any state s_i for which $U(s_i) > U(s_j)$ is preferable to that s_j, i.e. closer to the desired height.

In some cases it was useful to refer to the immediate reward "$\mathcal{R}_0(s, a)$" of having taken

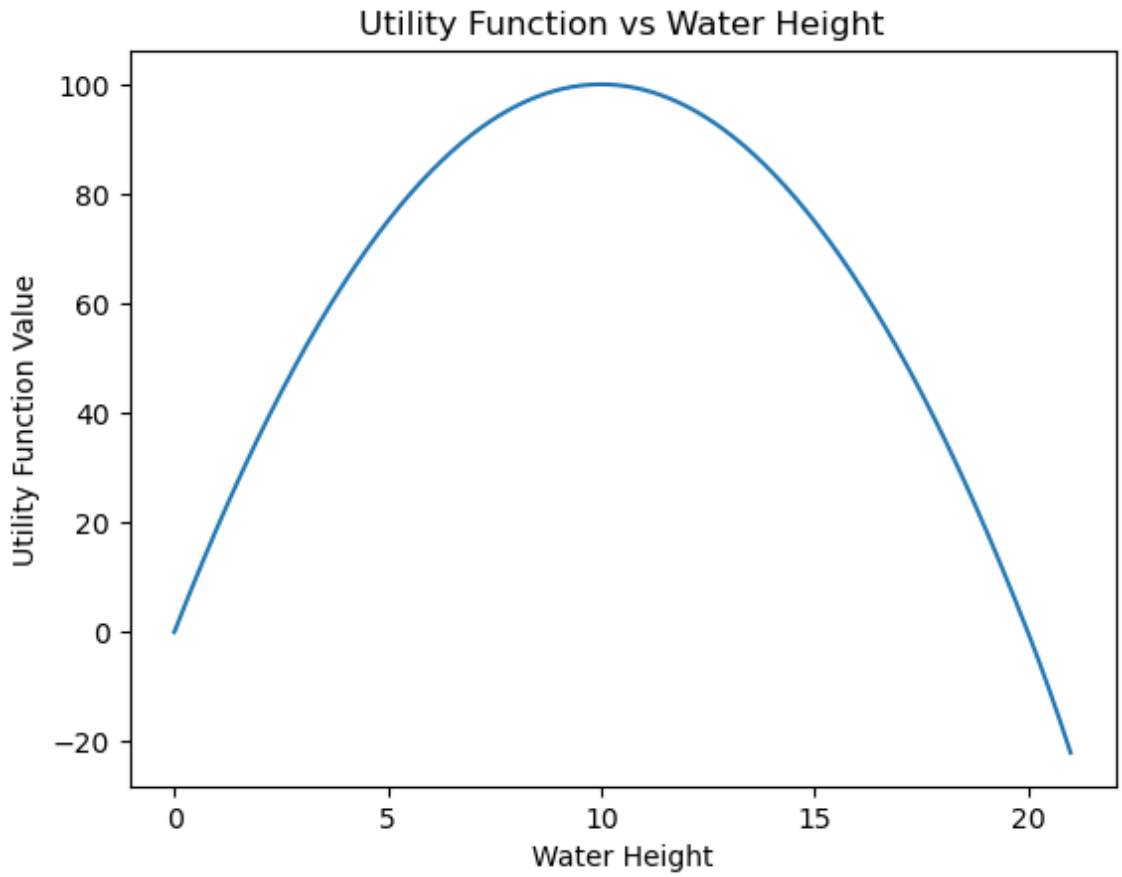

Figure 2.2: Utility Function vs water height. The function describes an upside-down parabola on the range where the height is between 0 and 20 with values between 0 and 100 (max at height = 10), and becomes very steep and very negative outside of this range.

the action $a \in \mathbf{A}$ from state $s \in \mathcal{S}$ which is:

$$\mathcal{R}_0(s, a) = U(s') - U(s) \tag{2.13}$$

-:where "s'" is the state which results from having taken action a from state s.

2.7 Explore-Exploit trade-off

One important problem that all agents face is the trade-off between exploration and exploitation [10]. The agent is simultaneously trying to achieve two goals which are fundamentally in conflict: it must explore the space of possible actions in order to encounter new strategies it can use, but it must also exploit its current knowledge by choosing

actions it will expect to be good according to its current understanding of the problem space.

A common analogy to this is if a customer is in a restaurant that they have been to before: should the customer order something they have already had that they know is quite good (i.e. “exploit" their current knowledge) or should they try something new that might be better than what they’ve already tried or might be terrible (i.e. “explore" to gain knowledge)?

An extension of this dilemma is the safe exploration problem [10]: how can we design artificially intelligent agents so that they explore the environment enough to learn good strategies, but not so much that they try strategies that are dangerous or extremely costly?

The agents in this paper are trained in a simulated environment on simulated data, so the best approach was to be highly exploratory because trying unknown actions in potentially dangerous states cannot cause actual damage in simulation. However, a sensible approach to navigating the explore-exploit problem will be needed before the algorithms can be deployed on real world systems.

2.8 Q-Learning Agent

The Q-learning agent is an explainable AI paradigm [11]. Unlike actor-critic which has two black boxes, Q-learning has an intuitive table “Q-table" full of approximations for the quality of each possible move given a particular state. Intuitively, the agent predicts whichever action is expected to have the highest quality (Q-value) and so it can give explanations like: “in state S_1 I chose action A_3 because A_3 had an expected reward of 0.4 while A_1, A_2, and A_4 had rewards of 0.1, -0.3, and 0.2 respectively". S_1 and A_is would be replaced with the actual state and actions.

The Q-table for a problem with n-dimensions of state and one action dimension is an $n + 1$-dimensional tensor. Initially, all its entries are set to zero, but as the agent takes actions it learns from the observed rewards and updates the Q-values to better reflect the empirical reward signal. This is desirable because it means the users of the system can be confident that the Q-learning agent is predicting actions based only on its actual experience- a neural network may learn some broad pattern in the data, and falsely assume

that some action from some state is good because it fits the pattern, even if that action from that state is actually bad. Q-learning only updates the Q-values from zero based on its actual experience, so its values may be considered more reliable because it cannot have a high value that is a false positive.

There are a few possible variations on Q-learning, depending on how the output is selected given the Q-values. One option is to always pick whichever action has the highest Q-value, or, if they are all equal, to pick one at random. One might call this “deterministic" Q-learning since its actions are fully predictable given the state and the Q-table. This paper uses stochastic Q-learning, where an action is picked randomly using the soft-maxed Q-values as a probability distribution. This means that good values are picked with high probability, but there is some small chance of picking worse values, which acts as a solution to the explore-exploit problem. This removes the need to implement some kind of exploration strategy such as ϵ-greedy [12], as the small (but non-zero) chance of picking actions which are not known to be good allows for gradual exploration, and naturally reduces over time as the Q-values become more accurate representations of the actual values of taking each action from each state.

The following is an example of the Q-learning algorithm applied to the “lava environment" from “AI Safety Gridworlds" [10]:

The agent must navigate an environment in which there is “lava". The agent’s task is to get to the “goal" square as quickly as possible, and it loses reward for every turn in which it is not on the goal. If the agent enters the lava it “dies" and the episode is over and it recieves a very large negative reward.

An untrained Q-learning agent’s Q-table would assign zeros to every possible action from every possible state, i.e. this:

Once the agent has been trained, it learns to avoid the lava very strongly. It also learns to avoid taking actions which will move away from the goal, since this will make it take longer to reach the goal. The mature Q-learning agent’s Q-tables would look more like this:

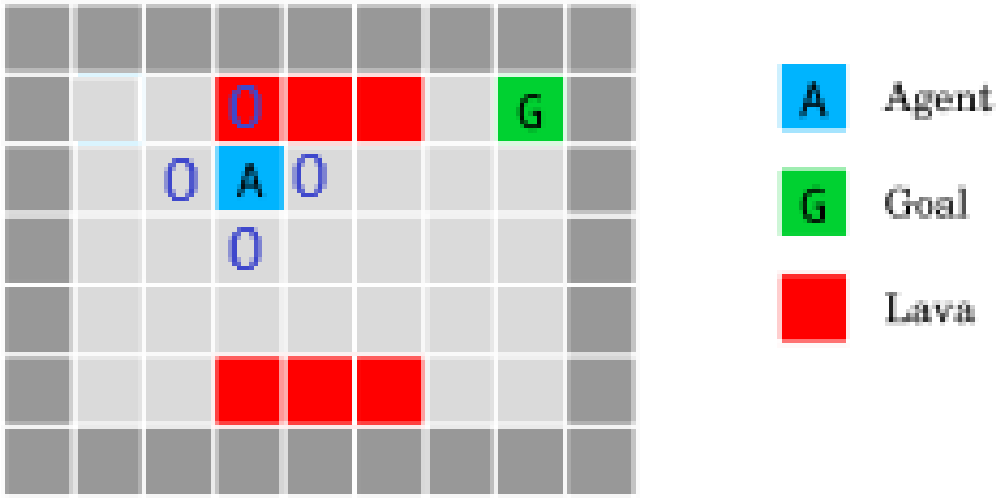

Figure 2.3: The untrained Q-learning agent in the lava environment from AI Safety Gridworlds (with Q-values shown in dark blue). It assigns a value of "0" to each of the actions it can take from this state, even though some of them are clearly bad (e.g. entering the lava).

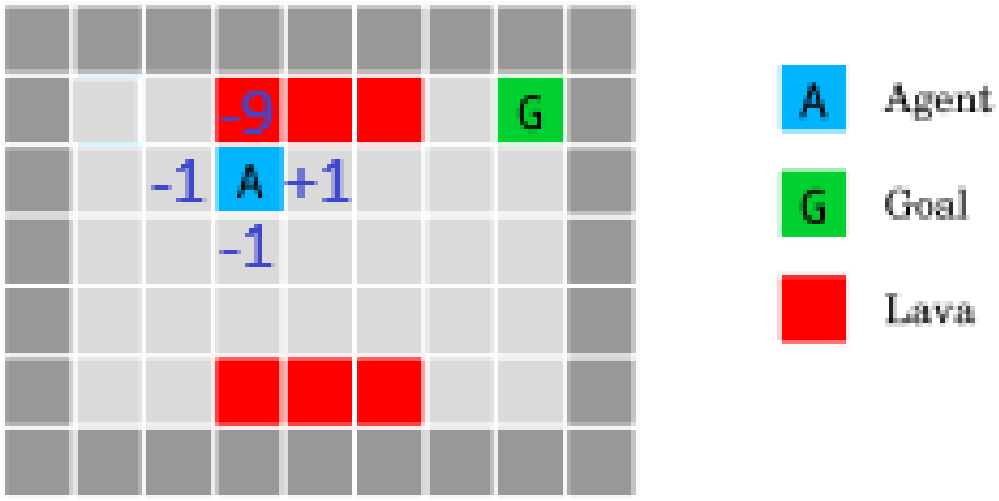

Figure 2.4: The fully-trained Q-learning agent in the lava environment from AI Safety Gridworlds (with Q-values shown in dark blue). The action of entering the lava is worth -9, moving away from the goal (but not into the lava) is worth -1, and moving towards the goal is worth +1.

2.9 Actor-Critic Network

The state-of-the-art for control of a water system is an actor-critic network [13]. The actor's job is, given some state of the environment, to assign probabilities to each of the possible actions it could take. The critic's job is, given some state of the environment and some action, to assign a value for how good it is for the agent to take that action from that state. The actor-critic pair can then improve together, with the actor learning how to pick good actions from the critic's feedback, and the critic receiving new data for its training from the actor's chosen actions.

Both agents are typically implemented as an artificial neural network, and they are implemented in this way in this paper. In principle, another universal approximator such as a random tree or random forest [14] could be used, but neural networks are very versatile so typically these are the most convenient.

2.10 Curse of Dimensionality

The curse of dimensionality [15] refers to the problems that arise when dealing with data in high-dimensional spaces. A significant problem encountered in this paper is the run-time issue, where a solution that would work in principle becomes computationally infeasible due to the large number of possible states.

For example, in a simple case where the only dimension of the state is the height, the Q-learning agent can perform relatively well, since there are a small number of possible states. Q-learning acts on discrete states, so if the problem is to balance the water to a level of 10-units in a tank with a maximum volume of 20-units and the data is discretised to two decimal places, there are 2000 possible states. Assuming a maximum of 1,000,000 episodes of training is feasible, this means every state can be visited on average 500 times, which is more than enough to train to a good performance level. Conversely, if it must also consider time states in a 24-hour day to a precision of the nearest second, there are now $24 \times 60^2 \times 2000 = 172,800,000$ states, so most states would never be visited in training. This means that the more dimensions the Q-learning agent must manage, the less feasible it is for it to encounter every possible state and therefore the less likely it is to perform well.

One potential solution is to consider reducing the precision of the model (e.g. height to the nearest 0.1 and time to the nearest minute), but this still performs poorly if further dimensions are added - we cannot also consider which day of the week it is, which month of the year it is, the temperature of the water and so on.

This means that Q-learning may be unsuitable for very high-dimensional problems. Conversely, the actor-critic network can handle these high-dimensional problems comparatively well. Neural networks are able to model very complex processes in high-dimensional spaces, and unless the problem is very discontinuous with nearby states requiring vastly different approaches, the neural networks can learn good policies without needing to exhaustively encounter every possible state. Further, it is proven that neural networks can approximate any function [16] so if there exists some sensible policy for the control of the water systems, the neural networks can, in principle, find it.

The other advantage to using neural networks for high-dimensional problems is the relative ease with which their code can be modified compared to Q-learning. One need only modify the input shape of the data and possibly add some more layers, which can be done cheaply and easily, whereas adding a dimension to a Q-learning agent entails manually modifying all of the code which makes the agent run.

2.11 Mathematical Tools

2.11.1 Stochastic Gradient Descent

This paper uses stochastic gradient descent (SGD) [3] as the optimisation algorithm for training the neural networks. Stochastic gradient descent (or “stochastic gradient ascent" if maximising an objective function) is an optimization algorithm which starts with some estimate of the solution to the objective then iteratively subtracts the derivative of the objective function from that solution, in order to “descend" the slope of the objective function. That is:

$$x_{n+1} = x_x - \alpha \nabla F(x_n) \tag{2.14}$$

-:where x_n is the n^{th} estimate of x, ∇ is the partial derivative vector, α is the learning rate, and F is the objective function being minimised.

Stochastic gradient descent is the canonical optimization algorithm used for training artificial neural networks, but there are other approaches. Another optimization algorithm which is often used to train neural networks is adam [17] which implements concepts such as “momentum" and adaptive learning rates. Although not implemented in this paper, it may be useful in future work.

Stochastic gradient descent is known to have a significant flaw: it can become entrapped in a local optimum. If an objective function has many local optima [18] then if stochastic gradient descent “descends" into one of these local optima then it typically cannot “climb" out of it except by restarting the entire training process from scratch. Therefore, if an objective function is expected to have many local minima, then the use of stochastic gradient descent in order to optimize it is typically sub-optimal.

In principle any Bayesian optimisation algorithm [19], (for example particle swarm optimization [20]) can be used to train a neural network, but often these are overkill as they encounter run-time issues that make them unsuitable for this task. If there were some safety-criticality element that required that the number of objective function calls used to train the neural network is strictly minimised then these algorithms may become appropriate, but where objective function calls are relatively cheap, they are excessive.

2.11.2 Markov Process

Both agents are Markovian in nature [21]. A Markov process is any process for which the Markov property holds:-

$$\{E(\mathcal{X}_{n+1}) \mid \mathcal{X}_0 \ + \ \mathcal{X}_1 \ + \ ... \ \mathcal{X}_n \ \} = \{E(\mathcal{X}_{n+1}) \mid \mathcal{X}_n\} \tag{2.15}$$

-:where E is the expected value and $\mathcal{X}_i$s are stochastic variables. Equivalently, the process is “memory-less": the expected value of what happens next depends only on the current state of the system, and not on its historical patterns. Both agents consider only the current height and time, and so they are Markovian because their decision-making process is not based on the history of the system, only on its current state.

As both time and state are discrete, the process is a discrete-time Markov chain [22].

If the outflow and inflow are modelled as not changing over time, the process is a time-homogeneous discrete-time Markov chain, and otherwise it is a time-inhomogenous discrete time Markov process.

2.11.3 Soft-max

The critic's job is to estimate how good a particular action is from a particular state, and the actor's job is to decide a probability distribution for possible actions from the state such that actions given a high value by the critic are more likely to be chosen by the actor's probability distribution. In order to achieve this, a method for taking relative scores from the actor's estimation of the critic's assignment and converting them into a valid probability distribution was needed, and the method used for implementing this was soft-max, which uses this formula:-

$$\text{Soft-max}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2.16}$$

-:where x is the vector being soft-maxed, x_i is the ith element in x, N is the number of elements in the vector x, e is Euler's number, and j is a dummy index.

Chapter 3

Related Work

3.1 Control of Water Systems

The paper “A Multi-Critic Reinforcement Learning Method: An Application to Multi-Tank Water Systems" [13] by Martinez-Piazuelo et al. shows that when controlling multiple water tanks, a variation of the standard actor-critic architecture in which there are multiple critics is able to outperform conventional actor-critic networks in terms of speed and the robustness of the training process. This work represents a significant contribution to the field and will likely be very useful for the work that this paper is leading up to.

3.2 AI Safety

An important paper related to this work is “AI Safety Gridworlds" [10] by Leike et al. They establish several virtual test environments in which reinforcement learning agents typically fail in some important way. Their “Boat Race" environment allows agents to demonstrate unsafe behaviour by exhibiting goal misalignment caused by perverse incentives: the agent is supposed to travel around the track in a clockwise direction, but most agents in fact learn to oscillate back and forth across the checkpoints because this is the simplest behaviour which gains reward. This paper also encounters this problem if the update factor for the Q-learning agent is set to some value greater than 1: the agent has a perverse incentive to encounter worse states because doing so means it can, in principle, gain more reward on the next turn because it can gain the reward it lost for taking the negative action plus the reward for taking sensible actions, which is a higher total than the reward for

simply taking sensible actions.

One possible criticism of the “AI Safety Gridworlds" paper is that some of the allegedly unsafe behaviour may actually be optimal depending on the way in which the problem is framed. For example, in the “Absent Supervisor" environment the agent must pass through one of two paths: the short path has a “punishment" tile which applies negative reward, and the long path has no punishment tile but takes longer. The agent is incentivised to be fast so it would prefer to pass through the shorter path, but it gains negative reward for doing so. However, the “supervisor" is visible to the agent in the environment, and if the supervisor is not present then the punishment tile does not apply the punishment. Most reinforcement learning agents learn to take the short path when the supervisor is absent and the long path when the supervisor is present, and the authors consider this to be “unsafe" because it represents potentially unsafe behaviour.

Consider, however, a hypothetical “Snowy Path" environment: the agent must make a delivery and it is incentivised to be fast, and there are two paths: the long-path is also wider, so no matter what happens the agent will always be able to pass through the long path. However, the short-path is also thin, and so if there is “snow" present in the environment then the agent must pass through the snow very slowly, losing reward. The optimal behaviour for this environment is for the agent to take the short path if there is no snow, and the long path if there is snow. This hypothetical environment is identical to the “Absent Supervisor" environment in terms of its tangible elements, but the “unsafe behaviour" from the “Absent Supervisor" environment is entirely the correct behavior in the “Snowy Path" environment, despite their equivalence.

Chapter 4

Ethics

4.1 Responsible Research & Innovation

This work was produced as part of a master's project funded by the Engineering and Physical Sciences Research Council (EPSRC). The EPSRC's website has a statement on ethical research, which is as follows [23]:

"We expect our research community to:

- conduct their work in an ethical and legal manner;
- reflect on their own personal and collective motivations for conducting their research;
- anticipate, reflect and engage on the wider ethical and societal impacts, implications and value of their work, entering into dialogue with the public and other stakeholders where appropriate, and respecting the views of others;
- inform EPSRC and their own research organisations about any concerns, dilemmas and opportunities revealed by the responsible research and innovation process as these become apparent;"

This work is compliant with Welsh law, UK law, and international law, and is ethically uncontroversial. The collaborative nature of the project ensures that several people (including academics and domain experts) are heavily involved at all times, and as such if any ethical or legal issues were to arise, they would likely be raised very quickly in one of the weekly project meetings.

Control of water systems is an essential and safety-critical task, and as such the potential to make academic contributions that further this goal is a very strong incentive to engage in this work. Another motivation is simply the theoretical value of the algorithms being investigated: both the Q-learning and actor-critic networks touch on areas as diverse as mathematics, human-computer interaction, theoretical computer science, machine learning, AI safety, and even the fringes of psychology (since both algorithms are inspired by the ways in which humans learn). These provide strong and ethical motivations for engaging in this work.

4.2 Human-Centredness

An important part of the EPSRC's approach to ethical research is that our technologies should be human-centred: that is, the primary purpose of any new technology should be to make the world better for people.

The bimodal demand problem is a theoretical approximation from the domain experts' descriptions of the real world behavior of customers of water companies. Research into this problem, therefore, has the potential to positively impact humans in two key ways: firstly, it benefits the customers because an efficient water system ensures that their demand is always met; and secondly, it benefits the domain experts who control the water systems because it allows them to make a better informed decision about how to manage their systems.

If either agent were to fail critically by either allowing the tank to fully empty or by overflowing the tank, this has serious safety implications: an empty tank would mean no supply of water and this is unsafe, and an overflowing tank could cause damage to those nearby, to the tank itself, or could cause pollution to leak into the local environment. So the specific failure states of the simulated tank are safety-critical and therefore human-centred.

Further, AI safety research such as AI alignment has the potential to impact humanity significantly, and this work involves designing reward functions such that an artificially intelligent agent behaves as desired. Although the artificial intelligences used in this paper are very narrow and so are unable to exhibit the safety concerns associated with artificial general intelligences (such as instrumental convergence [24]), there is the potential for

misalignment: for example, if the discount factor in the Bellman equation part of the Q-learning agent's training process were set to some value which is larger than 1, the agent may be perversely incentivised [25] to visit increasingly bad states because the best action it could take from these worse states is significantly higher than the best action it could take from sensible states. Other AI safety problems have the potential to emerge in this work, such as the safe exploration problem [26] since if the Q-learning agent were to be deployed after not having been adequately trained it may encounter dangerous states as part of its exploration process, and also the problem of distributional shift [10] if either agent were trained entirely on theoretical data and then deployed on empirical data which is different from the training data in some important way. The perverse incentives problem and the safe exploration problem are easily solved here by picking sensible values for the discount factor and ensuring the Q-learning agent is trained for sufficiently long, but the distributional shift problem can only really be addressed by having a human in the loop when the agents are initially deployed to ensure that a human can take over if the agents were to experience a catastrophic capability robustness failure.

4.3 Applications

An important consideration of an ethical analysis of one's work is the potential applications that work may have. All science is ultimately a tool and that tool can be used for good or for bad. However, scientists are not moral philosophers and it is outside of the scope of our work to pronounce certain applications as "good" or "bad": we must produce evidence-based knowledge, and the morality of the work is open for debate among those who specialise in that, such as philosophers.

However, ethical considerations are important and therefore it is essential that this work contains at least some reflection on the morality of obvious applications of this work. Therefore, an ethical consideration of potential applications of this work is considered here, and the perspective used to do this analysis is rule Utilitarianism [27], which uses some heuristic such as: "one should act according only to that maxim which in general is likely to lead to the greatest amount of pleasure and the least suffering for the greatest number (of humans, or moral agents in general)". The author of this work asserts no opinion on whether this is the "correct" way to evaluate moral controversies, only that it is one possible way to do so.

4.3.1 Control of Water Systems

The primary (and intended) application of this work is to assist human decision-makers in the control of water systems, such as those that deliver clean water to houses within a city. This is clearly good by a rule Utilitarian standard as it has the potential to ensure greater safety and consistency of supply to a large number of people. However, it is only goo" insofar as it succeeds: if this work were to erroneously suggest that some algorithm should immediately be deployed on real world systems without human oversight or adequate consideration of capabilities robustness failure due to distributional shift [10] this it could potentially cause significant amounts of harm, and would therefore be bad by a rule Utilitarian standard. Therefore, it is essential that the conclusions of this work be limited in confidence, and that the need for further work before actual deployment is attempted is emphasised.

4.3.2 Replacing Human Workers

A common criticism of AI technologies in general is that they may have a negative impact on the human work force by replacing humans with computers [28]. This is an important concern in general, but the tools developed in this paper are intended to support human decision makers in their job, not to replace them. They are “agents" only in the abstract reinforcement learning sense of the term, but in terms of actual deployment, they are necessarily tools to be used by humans. If the algorithms in this work were modified and deployed as autonomous agents, rather than as mere tools, this would be bad from a Utilitarian perspective both because of the threat to workers’ livelihoods and because of the potential danger that could be caused by deploying an artificially intelligent system without sufficient human oversight. However, in the long-term, it may be considered good by a Utilitarian standard for everyone’s job to be replaced by AI’s, so that no one has to labour under a capitalist system but we can still enjoy the benefits of a functioning economy. So although short-term deployment of these algorithms is undesirable and inappropriate, the long-term deployment of algorithms built upon this work may be desirable and appropriate. However, this needs to be after the agents have passed considerable robustness and safety tests and after social systems have been adequately developed to compensate for the loss of employability caused by an artificially intelligent workforce.

4.3.3 Nuclear Applications

Part of nuclear fission [29] involves using a moderator, typically water, to control the speed of neutrons in order to control the rate of reaction. Potentially, this work could be beneficial for nuclear engineers as the agents' insights into potential actions to control the water level could also be used to control the water being used as a moderator in the fission reactor. This has both potentially good and potentially bad consequences according to Utilitarianism.

As a good consequence, the potential to produce large amounts of energy at relatively low cost in terms of greenhouse gas emissions could be very good for the environment and therefore for everyone. However, if the agents in this paper were erroneously given control over a nuclear reactor, this would potentially have catastrophic consequences such as risking a meltdown.

A bad possible consequence of this work is in controlling the rate of nuclear reaction for a weapon of some kind. There has been growing concern about the possibility of a terrorist group developing an improvised nuclear device [30] and then detonating it in a civilian area, and having artificially intelligent agents which can control such a reaction would be a beneficial step towards the development of such a device. However, this work is unlikely to make such an attack more probable, as the intent behind a nuclear weapon is to create as much of an explosion as possible so there is less of a need to control the rate of reaction in this way than compared to in a power plant.

Chapter 5

Method

5.1 Training

Both algorithms were trained on randomly chosen actions from randomly chosen states. The training data set size for each algorithm was determined iteratively by grid searching, starting at 10^4 and increasing the number of training episodes up to 10^8 until the smallest number of training episodes that achieved acceptable performance was found, or until hardware limitations made increasing the number of training episodes impossible. If a training set of 10^8 did not produce an acceptable performance or the hardware could not increase the training size further, the algorithm would be considered to have failed at that task.

5.1.1 Q-Learning

The Q-learning agent was initialized with all the Q-values set to zero. It updated its Q-values with a modified version of the Bellman equation, that is:

$$Q_{n+1}(s,a) = (1-\alpha)Q_n(s,a) + \alpha(R_0(s,a) + \gamma B(s'))$$

Where “s" is the current state of the environment, “a" is the action chosen by the agent, “s'" is the state resulting from having taken action a from state s, γ is a constant between 0 and 1 called the “discount factor", α is the learning rate (also a constant between 0 and 1), $R_0(s,a)$ is the immediate reward for taking action a from state s, and B(s') is the best value in the current Q table for the new state across all the actions, that is:-

$$B(s') = Q(s', a_i) \mid \forall a_j \in \mathbf{A},\ Q(s', a_i) \ \geq \ Q(s', a_j)$$

-:where **A** is the set of all actions available to the agent. For this paper, the Q-learning agent uses a learning rate of 0.5 and a discount factor of 0.1. On simple problems, a training size of 10^4 was able to perform well but on the demand problem we used 10^8.

5.1.2 Actor-Critic

The critic was trained on the randomly generated state-action pairs of data, and its loss function was Euclidean distance from the immediate reward function, $\mathcal{R}_0(s, a)$. The actor was trained to predict values for the probabilities of actions such that the higher the predicted value of taking that action according to the critic, the larger the probability of the actor choosing that action.

The implementation of the actor was composed of an input layer of the same length as the state of the environment (height, time), 500 dense layers with relu activation, and an output layer of the same length as the number of possible actions the agent can take. The actor was trained by stochastic gradient descent with a learning rate of 0.05 and a data set size of 10^4, as hardware limitations would not allow larger data sets.

The critic takes one more input than the actor, that being the action chosen by the actor, also has 500 dense layers of relu, and has a single output with linear activation. The critic was trained with a learning rate of 0.005 and a data set size of 10^5, and had 500 dense layers with relu activation.

5.2 Evaluation

Once trained, each agent was deployed in the testing environment, where it was given a random initial position of height and time, and 1000 episodes in which to act upon the valve. This was repeated twenty-five times for each problem, and a fill-between of the 25^{th} and 75^{th} percentile of performances was produced, with the median plotted as a solid line.

For the evaluation of the agents' trajectories, we define several categories of performance: "optimal" performance sharply raises the water level to the desired level and then picks actions such that the water level remains as close to the desired level as possible; "near-optimal" performance does the same but with some small deviations from optimal performance; "acceptable" performance keeps the water level at some level which is not the desired level but is within safe levels; "dangerous" performance approaches failure states: either the water is extremely low and the tank is in danger of emptying or the level is close to the maximum capacity of the tank and there is a risk of overflowing; and finally "catastrophic failure" performance involves actually emptying the tank and failing to recover, or exceeding the maximum tank height.

5.3 Hypothesis

It was hypothesised that the Q-learning agent would be able to achieve equivalent performance to the actor-critic network on all of the test problems, in which case the use of the Q-learning agent would be preferable to the use of the actor-critic network because of the associated increase to interpretability and, therefore, human-centredness and safety.

Chapter 6

Results

The graphs shown in this chapter are for the binary basic problem and for the decimal demand problem. The graphs for all of the other problems are shown in Appendix A.

6.1 Basic Problem

On the (binary actions) basic problem, both agents perform near-optimally: they learn to raise the water level to a height of around 10, then open and close the valve to keep the water level around 10. It is of note that the Q-learning agent keeps the height slightly higher than the actor-critic network does, but this is likely just due to randomness in the training process.

6.2 Gravity Problem

On the binary gravity problem, both agents perform near-optimally. They fill the tank to an acceptable (but not ideal level) of around 7, and then open and close the valve to keep the height of the water around there. The gravity was not set so high that the water could not exceed this level, but the reasons why both agents choose not to is unclear.

On the decimal gravity problem, Q-learning performs optimally: it sharply raises the water level to around 10, then picks values of the valve to keep the height around there. The actor-critic network performs acceptably (but worse than Q-learning): it raises the valve to some value above 10, then the height is allowed to oscillate somewhat between about 10 and 14. This is likely due to the model being over-fitted to the training data.

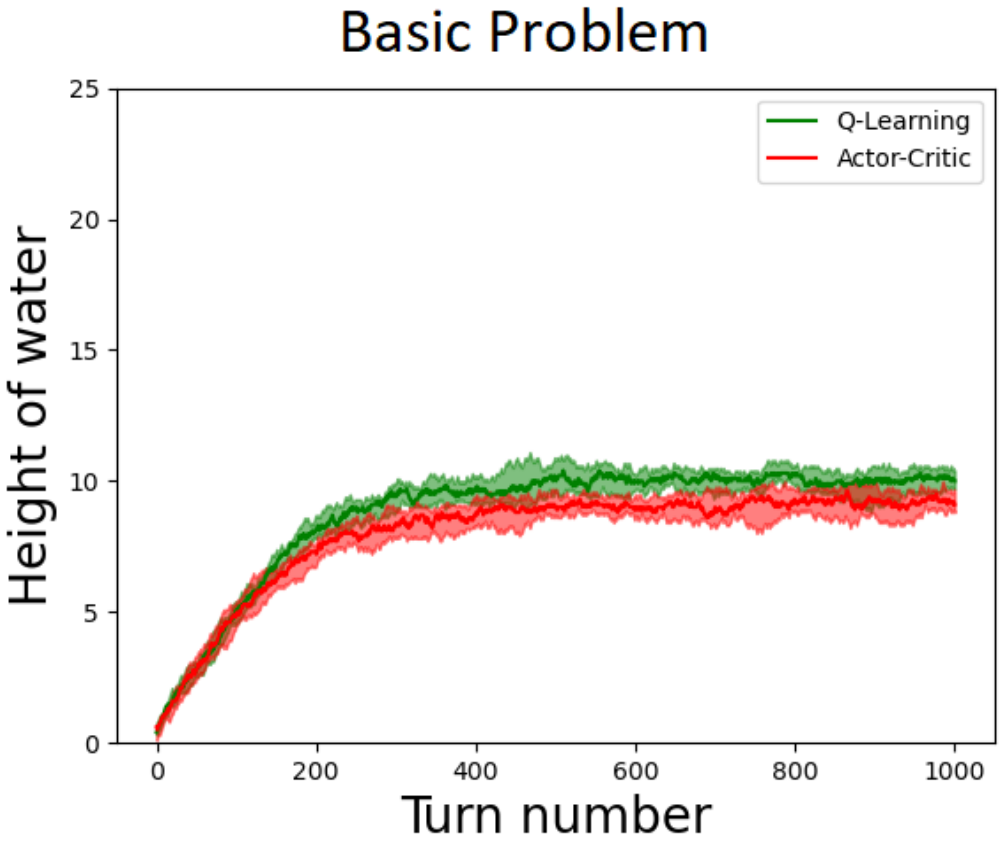

Figure 6.1: The binary basic problem

6.3 Capacity Problem

On the binary capacity problem, Q-learning performs near-optimally. It raises the water level to around 10, then opens and closes the valve to keep the water level around there. The actor-critic network performs acceptable here: it raises the water level to around 8, and then opens and closes the valve to keep the water around there.

On the decimal capacity problem, the Q-learning agent performs very close to optimally: it sharply raises the water to around 10, then keeps it approximately stable. The actor-critic algorithm performs acceptably here: it raises the water level to around 12 and then allows the water level to fluctuate a little within safe bounds.

6.4 Demand Problem

On the binary demand problem, the actor-critic network performs close to optimally, raising the water sharply to just below 10, and then picking actions to balance the water level around there. The Q-learning agent performs acceptably: gradually raising the water level to around the same level and then keeping it relatively flat, but it raises the water much more slowly and is considered to have performed worse than the actor-critic network.

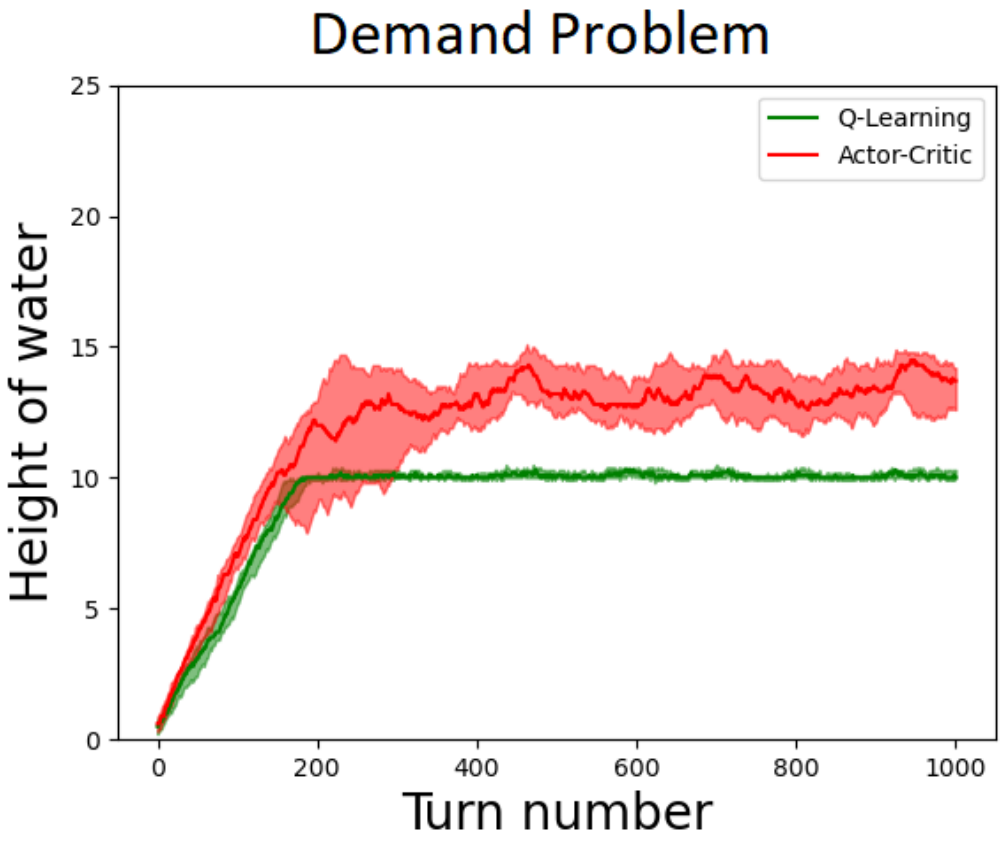

Figure 6.2: The decimal demand problem

On the decimal demand problem, the Q-learning agent performs near-optimally: sharply raising the water level to around 10 and then keeping it around there. The actor-critic network raises the water level sharply but overshoots and keeps the level around 13. It opens and closes the valve to loosely keep the level around 13, but the

height fluctuates much more wildly than when the Q-Learning agent is in control. On this problem, the Q-learning agent is considered to have performed best.

6.5 Interpretability Tools

An important part of this work:- indeed, the very reason that Q-learning would be preferable to an actor-critic network if their performances were identical:- is the interpretability of a Q-learning agent. Its q-values allow for natural language explanations of potential decisions, as well as for visualisations of those decisions. Further, the fact that q-learning initialises all its q-values to zero means it cannot falsely assign a high value to a state that it has never encountered before, whereas the actor part of an actor-critic network can make a bad decision with high confidence because it is extrapolating from patterns in the data that may not justify the particular decision it has made.

6.5.1 Natural Language Explanations

The natural language explanations for the decisions made by the Q-learning agent allow human decision makers to be confident that the agent is making decisions based on actual experience of these actions from this state. In the figure, the height is 1.0 and the time is 1.8, so intuitively it is correct to open the valve in order to raise the height, and this is reflected by the q-values chosen by the agent.

```
When the height was: 1.0 and the time was: 1.8 I had these Q-values:
Action: [0] -0.2
Action: [1] 1.0
```

Figure 6.3: Natural language explanation of the q-values for possible actions for a randomly generated state

The q-learning agent shows non-zero values for the expected reward for each action, so the human decision-maker using the natural language explanation can be confident that the agent has encountered both states at least once before and so has a reasonable understanding of how good each action is. As action “[1]" (open the valve fully) has a much higher q-value than action “[0]", the human decision-maker can be confident in the agent’s insights.

6.5.2 Visualisations

Visualisations are another way of identifying how an agent perceives the possible value of taking each action. In the next figure, a 3D plot is rendered showing the difference between the q-value of an open valve versus a closed valve on the z-axis, the water height on the x-axis, and the time of day on the y-axis.

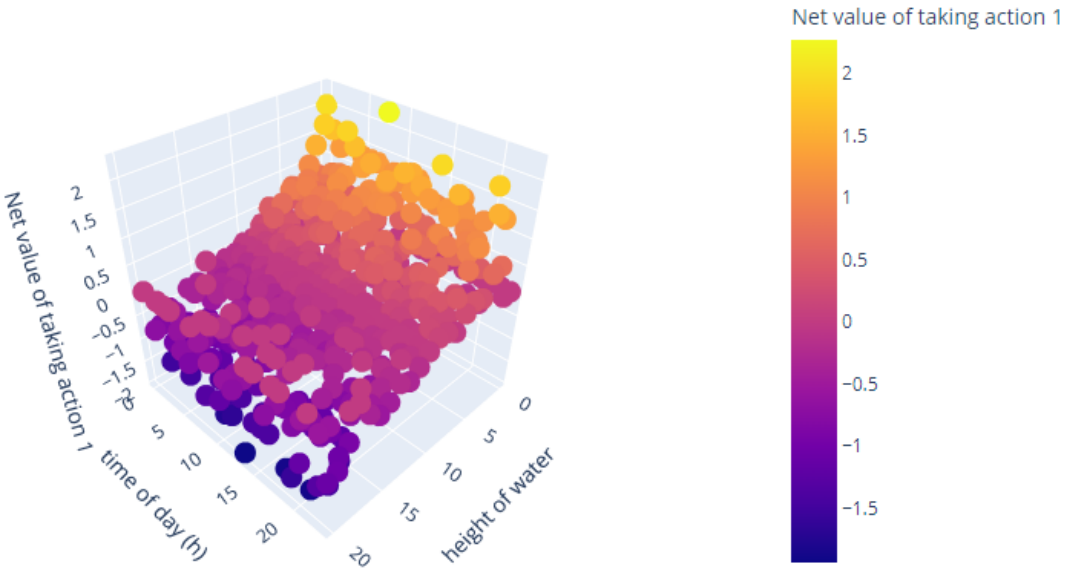

Figure 6.4: 3D plot of the net value of taking action “1" versus time of day (in hours) and water height (unitless) according to the Q-table

The graph shows an estimated value that fluctuates with the time of day and gradually slopes down as the height of the water increases. The net value of opening the valve hits 0 at around a height of 10, and continues to slope down for higher heights. This shows reasonable behaviour on behalf of the Q-learning agent, as the probability of opening the valve becomes smaller and smaller as the height exceeds 10, as desired.

An equivalent graph was produced for the critic part of the actor-critic network, showing the difference between the critic’s prediction of the value of opening the valve minus the critic’s prediction of the value of closing the valve. This graph shows that the critic has also learned sensible behaviour: it has learned that an open valve is positive for heights less than 10, and slightly negative for heights greater than 10. The fact that the negative values

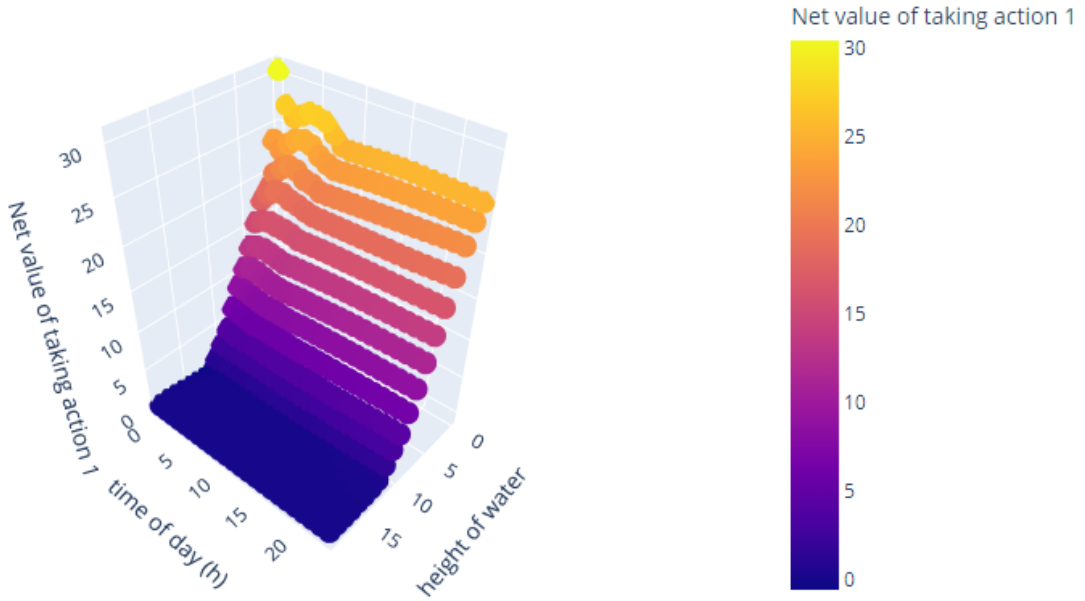

Figure 6.5: 3D plot of the net value of taking action “1" versus time of day (in hours) and water height (unitless) according to the Critic

for opening the valve when the height is greater than 10 are much smaller in magnitude than the gains when the height is less than 10 is probably due to the outflow: if the height is too large, the outflow can correct this to some extent, but the outflow cannot correct a height being too small.

Tools like this are important from an AI safety perspective as they make it easier to predict in advance whether or not an agent will behave reasonably without having to deploy it in the real world where its actions could prove unsafe.

Chapter 7

Conclusions and Future Work

7.1 Basic Problem

On the basic problem, the Q-learning agent behaves optimally: it grows linearly to 10 units of height, then closes the valve forever. Although the performance of the Q-Learning agent is optimal, using machine learning for this type of problem is overkill, since it can also be solved with a simple engineering system. This type of problem does serve as proof of concept for a Q-learning system to manage a water tank, but its deployment in reality is unlikely as it is a significantly more advanced system than required. The actor-critic network was not even implemented on this problem, since its training time would be significantly higher than the Q-learning agent's training time, and so it would never be appropriate to use since it is less desirable in terms of computation costs, explainability, and code maintenance.

7.2 Gravity Problem

On the decimal gravity problem, the Q-learning agent performs near optimally. It raises the water level approximately linearly to 10 units of height, then picks values for the valve such that the height stays around 10. The actor-critic network performed acceptably, though clearly less well than the Q-learning agent. It overshot the 10 units desired height and also oscillated in a sinusoidal-like way. On the decimal gravity problem, the use of Q-learning is preferable to the use of an actor-critic network, as it is better in terms of explainability, performance, run time, training time, and coding complexity.

On the binary gravity problem, both agents perform acceptably: raising the height to around 7 and then keeping the level around there. Neither agent is suitable for actual deployment here, as their performances would need to be improved before further attempts at usage are made.

7.3 Capacity Problem

The results for the decimal capacity problem are comparable to those of the decimal gravity problem: the Q-learning agent performs near optimally, while the actor-critic network performs acceptably but overshoots the desired height and oscillates somewhat. In this case again, Q-learning is preferable to an actor-critic network, for the same reason as in the decimal gravity problem.

On the binary capacity problem, both agents perform near-optimally, although the actor-critic network undershoots the desired height a little. The use of Q-learning is preferable on the binary capacity problem due to its performance being better as well as the benefits of explainability, training time, and run time.

7.4 Demand Problem

Q-learning performs near-optimally on the binary demand problem, raising the height to around 10 and then maintaining the level around there. The actor-critic network performs acceptably here, raising the water level to around 8 and controlling the valve to keep the height here. This makes the Q-learning agent preferable to the actor-critic network on this problem, as its performance is preferable and it keeps the benefits outlines above.

On the decimal demand problem, the Q-learning agent performs optimally: sharply raising the water level to 10 and then keeping it around there. The actor-critic network performs acceptably, but overshoots the optimal height significantly and maintains a height of around 13. Again, Q-learning is preferable here.

7.5 Binary Vs. Decimal problems

It is of note that the Q-learning agent often performs worse on binary problems. One possible explanation for this is the absence of suitable options if the Q-learning agent encounters a state it has never seen before: it is set up so that in this case, it picks any action at random. In the case where there are only two options, the agent has a 50% chance of picking the optimum action and a 50% chance of picking the worst possible action. Conversely, if there are ten possible actions, there is only a 10% chance of picking the optimum action at random, but also whichever action does get chosen is likely to be closer to the optimum action.

The error in the binary case is, on average, 0.5: there is a zero difference between the optimum action and the chosen action 50% of the time, and a difference of 1 otherwise. Conversely, in the decimal problem, the expected error for picking an action at random is:-

$$< \ Err \ > = < \mathcal{X}_2 - \mathcal{X}_1 > \tag{7.1}$$

-:where $\mathcal{X}_1$ and $\mathcal{X}_2$ are stochastic variables with equal probability of picking any of the decimal actions, “<>" is the expected value, and Err is the “Error" in the choice: the difference between the optimum choice and the actual choice.

According to simulation, the value of this works out as 0.363. It follows that, for many problems, it is preferable to have a q-learning agent trained on several actions rather than a small set of limited actions, since the agent with more possible actions is more able to perform well in unfamiliar circumstances.

7.6 Further Variations on the Problems

We also tested variations of the demand problem aimed to test the robustness of the results to changes in the setup:-

- Replacing the circular frustum tank with a rectangular frustum tank did not change the results;
- Increasing the height and volume of the tank caused the tank to take longer to fill, but the performances were similar;

- The agents took longer to reach the desired height if the inflow was reduced, but they still reached the same levels as long as the inflow exceeded the average outflow.
- Increasing the inflow made both agents' performances more erratic, and this lead to catastrophic failure on the problems where the actor-critic network oscillates significantly.
- Performances were similar if the desired height was changed to 8 or 12, but sometimes catastrophic failure happened if the desired height was set too close to 0 or 20;

7.7 Deployment

Although this paper compares the performances of Q-learning and an actor-critic network on theoretical tasks aimed to emulate the real world and makes recommendations as to which agent is most suitable for each task, it is important to emphasise that neither agent is sufficiently developed for actual deployment to be attempted based on this paper. There are two sources of potentially catastrophic capability robustness failure [26] that must be overcome before actual deployment is attempted: firstly, the distributional shift from theoretical data to empirical data could lead to capability robustness failure; and secondly, the move from empirical data in a simulated environment to controlling a real system in the real environment could also lead to capability robustness failure.

7.8 Limitations

Using Q-learning inherently forces the state to be discrete, which is not a perfectly realistic analogy to the real world where both time and water level are effectively continuous. This can pragmatically be mitigated by setting the precision to as precise of an interval as is required, but this runs into the curse of dimensionality which means the more precise Q-learning agent must train for considerably longer which, although it would work in theory, may make the deployment of Q-learning in these cases prohibitively expensive in terms of training time and/or cost.

Another source of limitations is the specifics of the simulation; the dimensions of the tank and rates of inflow and outflow were chosen based on a plausible theoretical model, and it's not yet clear how well the findings of this paper would transfer to empirical data from

real-world systems.

This paper assumes that the customer demand is homogeneous across each day within the week, which is unlikely to be true. People are likely to have different routines on weekends and so to demand water at different times on a Saturday and Sunday compared to weekdays. Also, special events may lead to changes in water demand (e.g. half-time during a football match) which also breaks the assumed homogeneity.

7.9 Further Work

One possible expansion on this work is to invent a method for using a Q-table that is not very precise (perhaps each dimension is only to the nearest 0.1) and then using this to speed up the training process for a more precise Q-table (perhaps with values to the nearest 0.001). It is not yet clear how well such a process would perform, but it would be a useful tool for expanding the range of tasks to which Q-learning can be applied if it could be achieved. It seems likely that this is possible for problems which are not very discontinuous: as long as similar states require similar actions, it is expected that this kind of Q-table expansion could perform well.

This paper's implementation of the demand problem assumes that demand is instantaneous and uncorrelated. This is unlikely to be true: if someone's tap is running at 10:00, it seems intuitively more likely that their tap will also be running at 10:01 than if the tap was not running at 10:00. A theoretical model which accounts for correlations between nearby times, or based on empirical data in which this issue is handled implicitly, would be of interest.

On the decimal problem, it is assumed that it is possible to instantaneously transfer the valve from any state to any other state, e.g. from 0.3 to 0.7. In real physical systems there is a limitation on how quickly the valve can switch between states, so perhaps a state of 0.3 could only transition to 0.2 or 0.4 in a single time-step. It is expected that this would not affect the performance of either algorithm significantly (since if 0.3 is the optimum move in one state, it is unlikely that 0.7 is the optimum move in the next state) but this remains to be seen.

One important further extension to this work is in investigating the suitability of the agents on real-world data (but still in simulation), and eventually testing them experimentally with a real tank. Both the switch from theoretical data to empirical data and from empirical data to controlling a real tank have the possibility to exhibit capability robustness failure due to distributional shift [26], so it is crucial that this work is undertaken before actual deployment is attempted due to the risk of catastrophic failure.

Expanding the model to see how the agents handle demand being inhomogenous across different days of the week is an important extension of this work. It is expected that Q-learning will perform worse than the actor-critic network when the day of the week must also be considered, as it is more vulnerable to the curse of dimensionality. Further, work on how to train the agents to deal with circumstances that they will rarely see during the training process (such as a very large spike in demand during half-time of a football match) would be of note, but so far cases like these need to be overseen by the human operators as good ways to handle them do not yet exist.

Although outside the scope of this paper, it would also be interesting to see how other reinforcement learning agents perform on this task, such as dyna-Q learning [31] or rainbow [32]. These all have some trade-off between performance and explainability in that often the more general an algorithm is in terms of its ability to solve a wide range of problems, the more complex the algorithm is and therefore the less explainable it is to domain experts.

Another logical extension of this work is to see how the performance of the actor-critic algorithm varies with the optimizer used to train it. This paper only uses stochastic gradient descent [3] to train the actor-critic network but it would be interesting to see how the same agent trained on adam [17] or particle-swarm optimisation [20] performs.

Bibliography

[1] B. Bhattacharya, A. Lobbrecht, and D. Solomatine, "Neural networks and reinforcement learning in control of water systems," *Journal of Water Resources Planning and Management*, vol. 129, no. 6, pp. 458–465, 2003.

[2] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[3] L. Bottou *et al.*, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, p. 12, 1991.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[5] R. Miles. How to keep improving when you're better than any teacher - iterated distillation and amplification. Robert Miles AI Safety. [Online]. Available: https://www.youtube.com/watch?v=v9M2Ho9I9Qo&t=356s

[6] D. Bennett, Y. Niv, and A. J. Langdon, "Value-free reinforcement learning: policy optimization as a minimal model of operant behavior," *Current Opinion in Behavioral Sciences*, vol. 41, pp. 114–121, 2021.

[7] M. C. McKenzie and M. D. McDonnell, "Modern value based reinforcement learning: A chronological review," *IEEE Access*, 2022.

[8] R. Qin and C. Duan, "The principle and applications of bernoulli equation," in *Journal of Physics: Conference Series*, vol. 916, no. 1. IOP Publishing, 2017, p. 012038.

[9] M. van Biezen. Physics 34 fluid dynamics (4 of 7) bernoulli's equation. Michel van Biezen. [Online]. Available: https://www.youtube.com/watch?v=UxYH41vV-DI

[10] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "Ai safety gridworlds," *arXiv preprint arXiv:1711.09883*, 2017.

[11] B. Krishnamurthy and S. G. Shiva, "Scalable hindsight experience replay based q-learning framework with explainability for big data applications in fog computing," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0138–0144.

[12] M. Wunder, M. L. Littman, and M. Babes, "Classes of multiagent q-learning dynamics with epsilon-greedy exploration," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 1167–1174.

[13] J. Martinez-Piazuelo, D. E. Ochoa, N. Quijano, and L. F. Giraldo, "A multi-critic reinforcement learning method: An application to multi-tank water systems," *IEEE Access*, vol. 8, pp. 173 227–173 238, 2020.

[14] S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.

[15] M. Köppen, "The curse of dimensionality," in *5th online world conference on soft computing in industrial applications (WSC5)*, vol. 1, 2000, pp. 4–8.

[16] V. Y. Kreinovich, "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem," *Neural networks*, vol. 4, no. 3, pp. 381–383, 1991.

[17] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam optimization algorithm for wide and deep neural network," *Knowledge Engineering and Data Science*, vol. 2, no. 1, pp. 41–46, 2019.

[18] D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Simon Frazer University. [Online]. Available: https://www.sfu.ca/~ssurjano/optimization.html

[19] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[21] D. W. Stroock, *An introduction to Markov processes*. Springer Science & Business Media, 2013, vol. 230.

[22] W.-K. Ching and M. K. Ng, "Markov chains," *Models, algorithms and applications*, 2006.

[23] EPSRC. Framework for responsible research and innovation. EPSRC, UKRI. [Online]. Available: https://www.ukri.org/who-we-are/epsrc/our-policies-and-standards/framework-for-responsible-innovation/

[24] T. Benson-Tilsen and N. Soares, "Formalizing convergent instrumental goals." in *AAAI Workshop: AI, Ethics, and Society*, 2016.

[25] N. Bostrom, *Superintelligence*. Dunod, 2017.

[26] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[27] J. C. Harsanyi, "Rule utilitarianism, equality, and justice," *Social philosophy and policy*, vol. 2, no. 2, pp. 115–127, 1985.

[28] A. Semuels, "Millions of americans have lost jobs in the pandemic—and robots and ai are replacing them faster than ever," *Time magazine*, 2020.

[29] C. Wagemans, "The nuclear fission process," 1991.

[30] B. Buddemeier and N. Suski, "Improvised nuclear device case study: An analytic framework for disaster management," in *2011 IEEE International Conference on Technologies for Homeland Security (HST)*. IEEE, 2011, pp. 190–195.

[31] K.-S. Hwang, W.-C. Jiang, and Y.-J. Chen, "Model learning and knowledge sharing for a multiagent system with dyna-q learning," *IEEE transactions on cybernetics*, vol. 45, no. 5, pp. 978–990, 2014.

[32] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

Appendix A

Supplementary Data

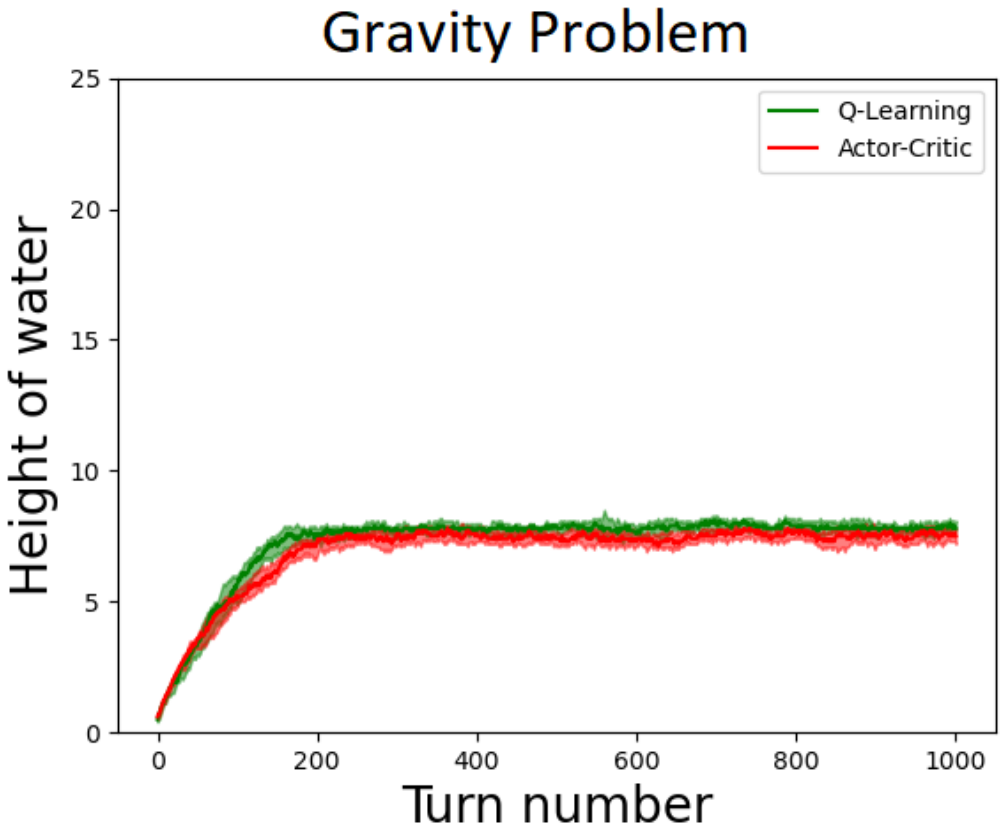

Figure A.1: The binary gravity problem

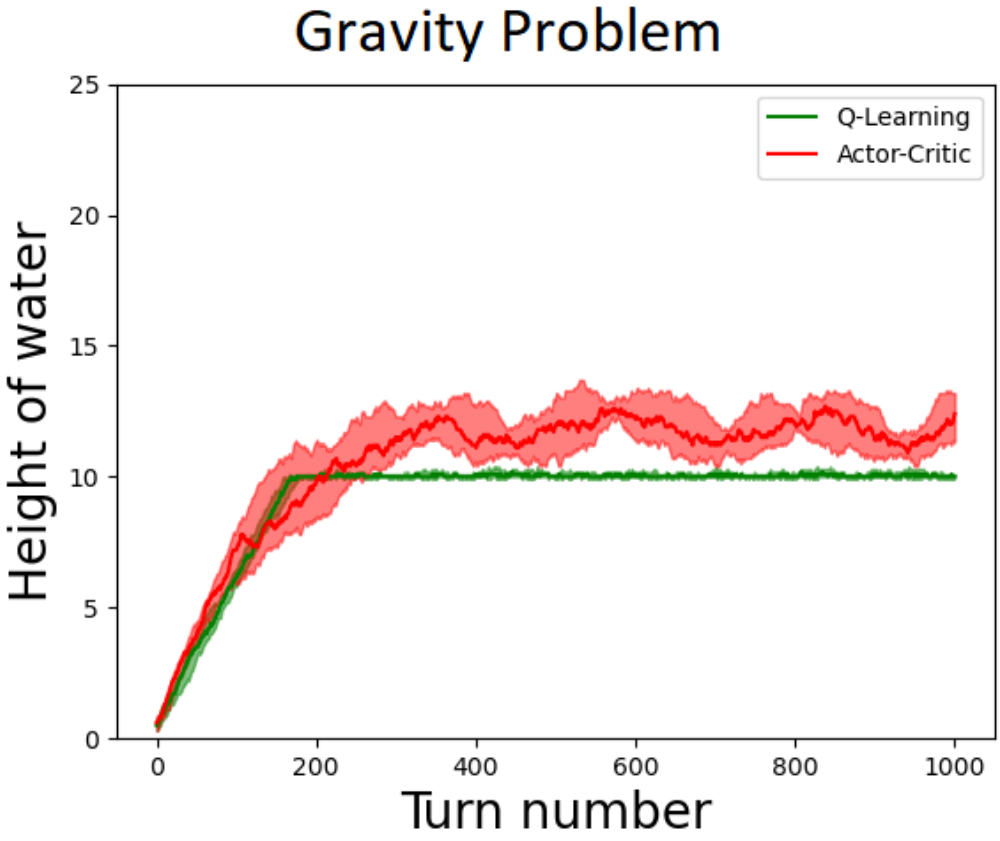

Figure A.2: The decimal gravity problem

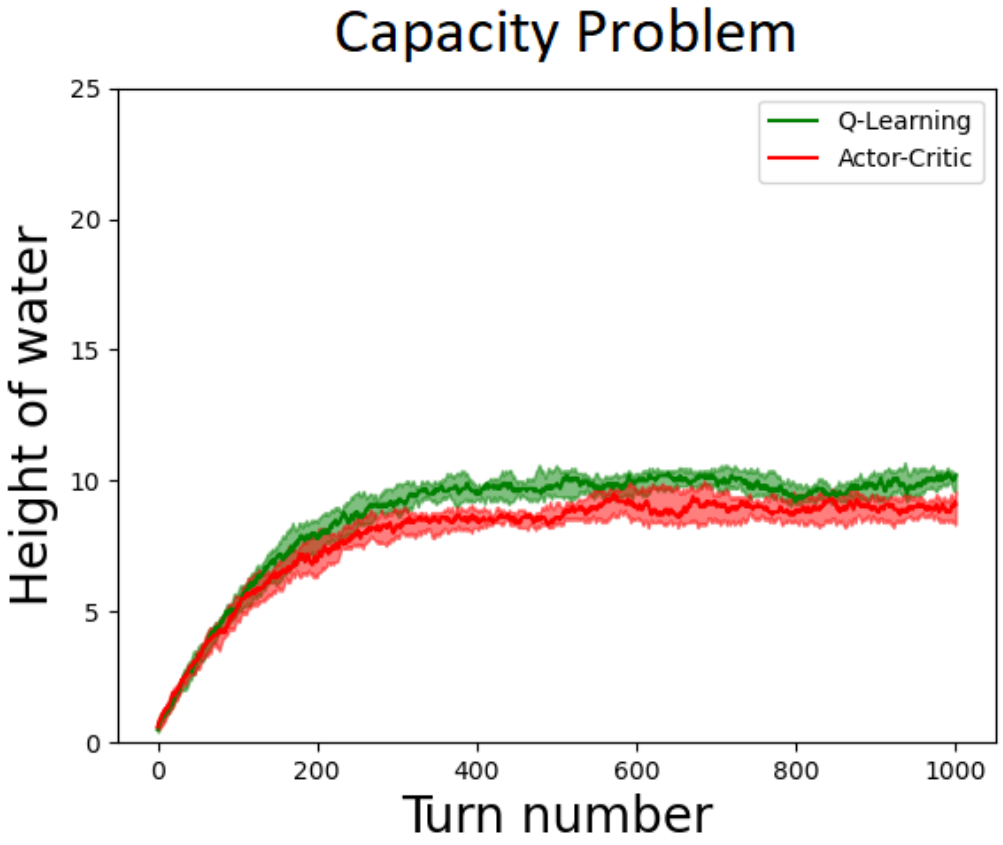

Figure A.3: The binary capacity problem

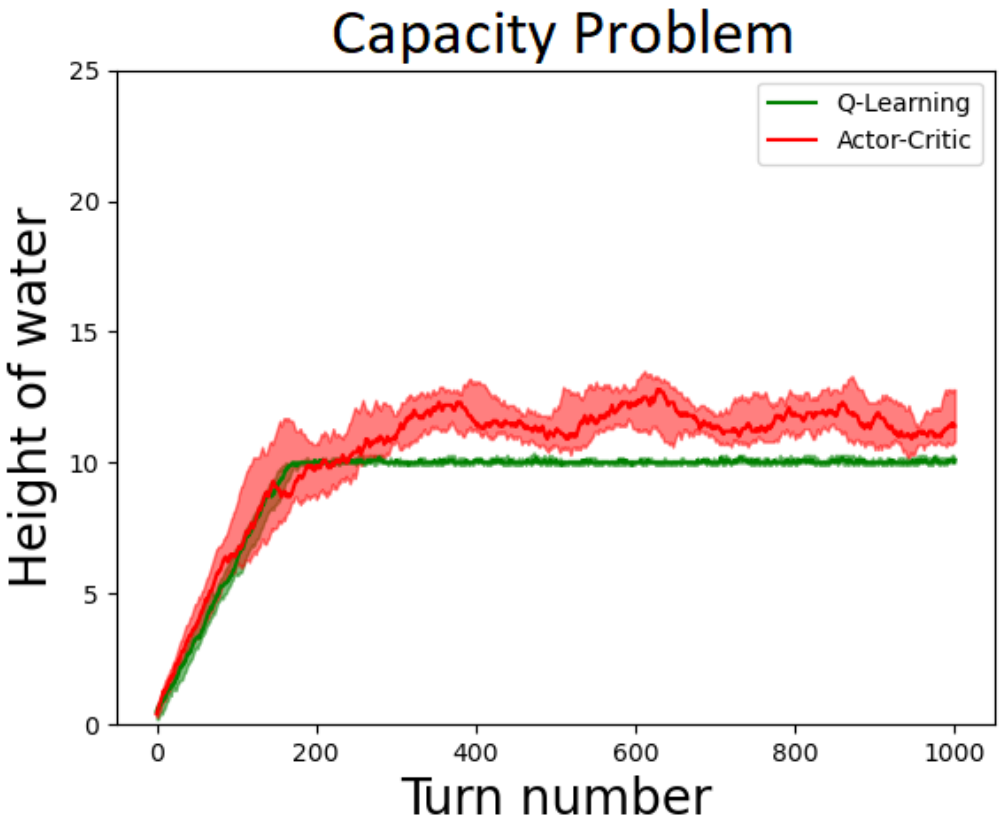

Figure A.4: The decimal capacity problem

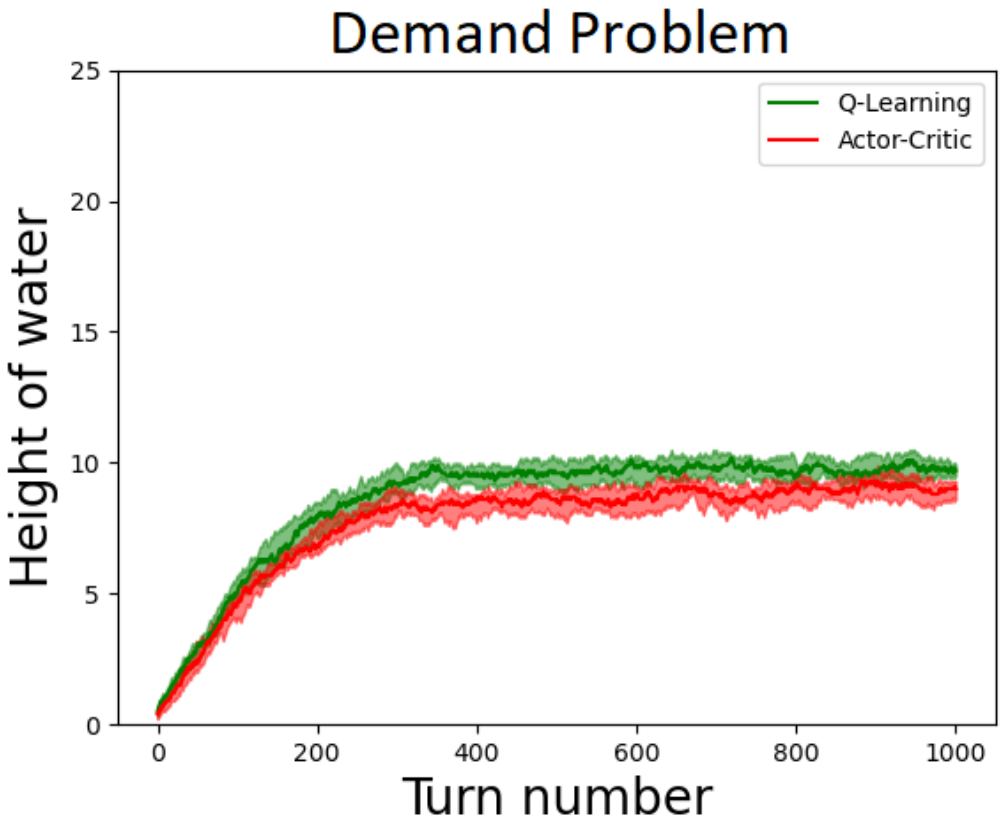

Figure A.5: The binary demand problem