Exploration Strategies for Homeostatic Agents

Continuous and dynamic exploration for homeostatic regulation using deep reinforcement learning

Master's thesis in Computer science and engineering

Patrick Andersson Anton Strandman

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Exploration Strategies for Homeostatic Agents

Continuous and dynamic exploration for homeostatic regulation using deep reinforcement learning

Patrick Andersson
Anton Strandman

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

Exploration Strategies for Homeostatic Agents
Continuous and dynamic exploration for homeostatic regulation using deep reinforcement learning
Patrick Andersson
Anton Strandman

Supervisor: Claes Strannegård, Department of Computer Science and Engineering
Examiner: Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Exploration Strategies for Homeostatic Agents
Continuous and dynamic exploration for homeostatic regulation using deep reinforcement learning
Patrick Andersson
Anton Strandman
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This paper introduces and evaluates four novel exploration strategies for homeostatic agents. Homeostatic agents have the objective of keeping some internal variables as close to a predetermined optimum as possible. Reinforcement learning is used for decision making, and the agents are given access to the optimal and acceptable values of the internal variables, giving greater flexibility for exploration and better survival chances. The new exploration strategies that utilise the internal variables are evaluated in a range of environments, showing them to outperform common reinforcement learning exploration techniques where these variables are not taken into consideration.

Keywords: artificial general intelligence, multi-objective reinforcement learning, exploration, homeostatic regulation, animat, homeostatic exploration

Acknowledgements

First of all, we would like to thank our supervisor Claes Strannegård for introducing us to the topic, as well as his feedback and guidance. We would also like to thank Chalmers for providing invaluable resources to make this project feasible.

Patrick Andersson & Anton Strandman, Gothenburg, May 2019

Contents

List of Figures

List of Tables

1

Introduction

Machine learning (ML) and Artificial Intelligence (AI) applications are becoming more and more ubiquitous and can be found in everything from cars to phones. These applications are often powerful and can in some instances even outperform humans [1]. But there are limitations to these implementations. They are so-called *narrow* AI [2], as such that they are purpose-built for a single problem. For example, solutions that can beat humans at recognising images will fail to recognise songs, as they are not designed to do so. Artificial *General* Intelligence (AGI), on the other hand, is a branch of AI that aims to solve multiple tasks with a single agent. While AGI has not yet been achieved, one approach is to simulate artificial animals, so-called *animats* [3], tasked with maintaining multiple internal variables, such as energy and hydration levels. By simulating more and more advanced animals, human level general intelligence could eventually be achieved.

These agents reside in an environment in which they explore and exploit in order to learn how to survive. However, deciding when to explore and when to exploit is not trivial. This is called the *Explore-Exploit dilemma* and is crucial for any self-learning agent to master in order to perform well [4].

In this paper, we introduce four novel exploration strategies capable of dynamically deciding when to explore and when to exploit based on the current state of the internal variables of the agent. This is done by modelling an animat as a *Homeostatic* Reinforcement Learning (HRL) problem [5], which is different from traditional Reinforcement learning (RL). In HRL the task is to maintain some optimal values of some internal variables, instead of maximising a single reward. These internal variables could be the energy and hydration of an animat, the battery charge of a robot or even the water level and electricity generation of a hydroelectric power plant, all of which have some optimal level and are affected by external events. The internal variables and optimal levels provide additional information related to the current state of the agent which can be utilised in the exploration strategy to provide continuous exploration and give the agents a better chance of success.

The typical exploration strategy used in state of the art RL is the annealed ϵ-greedy strategy [6, 7]. However, this strategy has some drawbacks: it will only perform any significant exploration at the beginning of a simulation. This requires human engineering to find the optimal amount of time to explore, and the agent

may struggle to explore new states once this training session is completed. We show that in the HRL problem simple heuristics can generate strategies that outperform traditional RL exploration strategies as such that they facilitate more exploration throughout the simulation while at the same time improve the agents' ability to maintain the internal variables, all with minimal human interaction.

In the following chapter, information is provided related to the implementation of the agent and the exploration strategies. In chapter 3 we give details of how the HRL agent is implemented, how the strategies are devised as well as how the strategies are evaluated. Finally, we present and discuss the results, and show that the novel exploration strategies, named homeostatic exploration, are able to outperform traditional RL exploration techniques.

2

Background

In this chapter, we give a brief introduction to animats and why simulating animals can be considered an HRL problem. We also give some background to reinforcement learning and homeostatic regulation, and how these can be combined into homeostatic reinforcement learning.

2.1 Animat

The animat model was proposed by Stewart Wilson [3, 8] as a means of achieving Artificial General Intelligence [2] (AGI). According to Wilson, AGI can eventually be achieved by simulating progressively more complex animals. As such the animat model attempts to imitate animals and is an adaptive system with needs, sensors and motors. The sensors and motors should not be purpose-built for a specific task, instead, the animat should learn to generalise the input and drive the motors in such a way as to fulfil its needs and survive, similar to how the internal variables should be maintained in the HRL problem. In addition, an implementation of the animat model should be capable of incremental adaptations triggered by, for example, a change in environment. As such the model has to be adaptive, flexible and be capable of continuous learning through its lifetime.

2.2 Reinforcement learning

Reinforcement Learning is a sequential decision-making problem in which an agent, at every time step, decides upon an action based on some observation and reward received from an environment. It has many similarities to the animat model, where the sensors generate observations, and actions drive motors. However, in RL the objective is to find a policy so as to maximise cumulative reward given by the environment [9]. To find this policy the agent must explore the environment, and observe which states and actions result in rewards. This is typically done in a trial-and-error fashion in which the agent will take a random action and observe the rewards received.

The RL problem is usually defined as a reward maximisation problem in some environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$. Here, the $\mathcal{S}$ and $\mathcal{A}$ are the state and actions spaces while $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $P(\cdot|s, a)$ is a transition function and $\gamma \in (0, 1)$ a discount factor. At each time step, an agent observes a tuple (s_t, r_t) and produces an action a_t whereupon the time increases and a new tuple is generated. This cycle repeats until some predefined state is reached.

In a given time step, the return is defined as $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ and the general goal of RL is to maximise the return R_t at each state s_t. The optimal action-value function $Q^*(s, a)$ is defined as

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{\pi} \left[R_t | s_t = s, a_t = a \right] \tag{2.1}$$

where $\pi(\cdot|s)$ is a policy defined as a mapping from a state to a distribution over actions. Given Q^* the optimal policy follows directly as $\arg\max_a Q(s, a)$ and in general the problem is to find Q^* and then act greedily with regards to it.

2.2.1 Q-learning

Q-learning is a method of iterative approximation of the optimal Q-value function according to Equation 2.2. Given the previous state and action s, a and the current state s', the Q-value estimate is updated towards some target $Y = r + \gamma \max'_a Q(s', a')$ with learning rate α.

$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right) \tag{2.2}$$

An agent implementing Q-learning will create an approximation of the Q-value function by utilising some exploration strategy to acquire observations from the environment. As can be seen in Algorithm 1, given a state and a Q-value approximation for each action, the strategy will decide which action to take. The agent then observes the results of the action in the environment and updates the Q-value function to improve predictions. In practice, the agent will sometimes encounter a terminal state, death. When this occurs, no future return can be obtained and the Q-value update only includes that reward for the current time step. The simulation is then reset to its initial state.

2.2.2 Deep Q-Networks

While it is possible to apply a tabular approach, storing a value for each state-action pair, to a problem with a large state space, it is often of no practical interest. Such an algorithm very quickly runs out of memory. Moreover, the tabular implementation will not be able to generalise, that is it will not take into consideration that similar states may have similar returns.

Algorithm 1 Q-learning

Require: Exploration strategy STRATEGY, step size α, function Q, environment $\mathcal{E}$, start state s, discount factor γ, number of time steps T

for T time steps **do**
$a \leftarrow \text{STRATEGY}(Q,\, s)$
$s', r \leftarrow$ Observe results of taking action a in $\mathcal{E}$
$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$
$s \leftarrow s'$
end for
output Q

In order to circumvent these problem we turn to approximating the Q-function with some parameters $\boldsymbol{\theta}$, $Q(s,a;\boldsymbol{\theta})$. While such an approximation can be done in many ways, we will focus on Deep Q-Networks (DQN) [10] for one primary reason. The capability of neural networks to approximate a wide range of functions. There is, therefore, no need to switch algorithm between environments and the number of layers of the agents can be changed instead. In general, the update rule for the parametric approach with parameters $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left[Y_t - Q(s,a;\boldsymbol{\theta}_t) \right] \nabla_{\boldsymbol{\theta}_t} Q(s,a;\boldsymbol{\theta}_t). \tag{2.3}$$

where the default Q-learning target $Y_t = Y_t^Q$ is

$$Y_t^Q = r + \gamma \max_{a'} Q(s',a';\boldsymbol{\theta}_t). \tag{2.4}$$

For DQN, however, there are two additions. First, the Q-function used in the target calculation is estimated using an old version of the current network. The target thus becomes

$$Y_t^{DQN} = r + \gamma \max_{a'} Q(s',a';\boldsymbol{\theta}_t^-). \tag{2.5}$$

The parameters, or weights, of the old version, $\boldsymbol{\theta}^-$, are updated after a number of steps to the current network. This has the effect of decorrelating the predictions and the targets as the targets are fixed for some time. This is important to improve the stability of the Q-values.

Second, all (s,a,r,s',t) experiences are saved in order to form a stored dataset. Here, t is a boolean variable indicating if the step resulted in a terminal state. The network is then trained on this dataset, using randomly sampled minibatches, instead of training on the most recent experience. This has the effect of temporally decoupling experiences, making the data less correlated and decreasing the effect of changes to the data distribution. It also allows the agent to train on the data multiple times allowing for increased data utilisation.

2.3 Homeostasis

In order to simulate animats, some model of living organisms needs to be employed. Homeostasis is a popular framework with a long history rooted in physiology [11, 12] that provides a natural starting point. A set of internal variables, together with points, or intervals, corresponding to optimal values are given to an agent who is tasked with maintaining each variable in its optimal state. As an example, consider a simple animat that needs both food and water. Eating or drinking too little would lead to starvation and dehydration, while too much may lead to overconsumption and overhydration. As such the animat must balance these needs within some bounds.

There exists various alternatives to the ideas of homeostasis such as allostasis [13] and adaptive homeostasis [12]. As these ideas can be ambiguous and overlap to a significant extent [11], when the term homeostasis is used in this paper it is taken to mean the steady state of an animats internal variables. This is in turn achieved through:

- Central control - The actions taken in order to counteract deviation in the internal needs are issued from a central actor. While there are animal behaviours that contradict this—simultaneous shivering and sweating [11]—we are here concerned with modelling a simple brain. With a single centre for control, we can also better prevent multiple courses of actions which lie at odds with each other.

- Learnable regulation - The agent should be able to learn which actions that minimize internal deviations. It should thus learn to connect action outcomes to the resulting change in the internal state. It may also learn to initiate behaviour before an internal variable has deviated from its set point in anticipation of such a deviation.

In order to maintain homeostasis, the RL framework may be applied. However, in order to do so, we need to translate the homeostatic regulation problem into the notion of reward for the RL setting. A natural approach to doing this is to turn to the subfield of Multi-Objective Reinforcement Learning.

2.3.1 Multi-Objective Reinforcement Learning

Multi-Objective Reinforcement Learning (MORL) is a subfield of RL where the agent receives multiple rewards in the form of a vector instead of as a scalar. For example, the agent may receive rewards for picking up boxes, as well as receive negative rewards for energy consumed. As such the agent is supposed to minimise the energy consumed at the same time as trying to pick up as many boxes as possible. However most existing RL algorithms use scalar rewards, that is a single number,

and do not extend naturally to multiple rewards.

In general, there are two approaches to solving MORL problems [14]. As the multiple objectives are not necessarily comparable, the first approach extends the notion of an optimal solution to include all policies which are not provably worse than any other, the Pareto frontier. The other approach is to provide a preference for each reward, allowing them to be compared and subsequently to be transformed into a scalar reward. In this case, the problem can be simplified to the regular RL problem.

Connected to the second approach is the reward hypothesis:

> That all of what we mean by goals and purposes can be well thought of as the maximisation of the expected value of the cumulative sum of a received scalar signal (called reward). [9, p.53]

In effect, this would mean that all MORL problems can be simplified into regular RL problems. There is one catch though, the hypothesis gives no hint on how to find a scalarisation function, and it is at this point that MORL finds its use [15].

In this paper, homeostatic drive reduction [5] is used as a scalarisation function, described in the next section, and the viewpoint that a scalar reward is sufficient is adopted.

2.3.2 Homeostatic reinforcement learning

There are multiple ways to integrate Reinforcement learning and homeostatic regulation [16, 17, 18, 19, 20]. Of interest is a method that accomplishes two things: it exposes the internal variables to the agent and it computes a scalar reward from the internal variables. One such approach is given by Keramati and Gutkin [5, 21]. They present a framework that unifies the act of reward maximisation with internal regulation and exhibits a number of relevant properties.

The framework builds on the application of drive reduction theory, motivation as a response to internal variables [22], and makes two changes to the initial RL problem. First, the observed state is now comprised of internal variables in addition to the external observation. The state space of the internal variables is defined as $\mathcal{H} = \{\mathbf{h} \,|\, \mathbf{h} \in \mathbb{R}^N\}$ where N is the number of internal variables. The new state space becomes the Cartesian product of the old state space and the internal state space $\mathcal{S}' = \mathcal{S} \times \mathcal{H}$. An observation is then the external observation concatenated with the internal state vector of length N.

Second, the reward is changed as there are no extrinsic rewards from the environment. A scalar reward is generated within the animat itself in each time step based on the changes in the internal variables. The drive, d and reward, r_t at time step t

are defined as

$$d(\mathbf{h}_t) = \sqrt[m]{\sum_{i=1}^{N} |h_i^* - h_{i,t}|^n} \tag{2.6}$$

$$r_t = d(\mathbf{h}_t) - d(\mathbf{h}_{t+1}) = d(\mathbf{h}_t) - d(\mathbf{h}_t + \mathbf{k}_t) \tag{2.7}$$

where h_i^* is the optimal level of the homeostatic need h_i, and $h_{i,t}$ is the value of the internal need h_i at time step t. In this paper we will assume $\mathbf{h}^* = \mathbf{0}$ and constrain $h_i \in [-1, 1]$ by scaling to some acceptable bounds and the optimal value, neither of which will have any impact in the theoretical sense. Intuitively, the reward between t and $t+1$ is the difference in distance from $\mathbf{h}_t$ to $\mathbf{h}^*$ and $\mathbf{h}_{t+1}$ to $\mathbf{h}^*$. $\mathbf{k}_t$ is the change in internal variables for time step t and is given by the environment.

The hyperparameters n and m control how the drive scales based on the distance from the optimum. For example when $n = m$ the drive magnitude does not scale with the distance; a step towards the optimum when nearly at the optimum will give as much reward as a step of the same length and direction when further away from the optimum. In contrast when $n > m$ the reward for two steps of identical length will be larger if the internal variables are further from the optimum, while the opposite is true for $m > n$. In effect, having $n > m$ in the case of an animal will make eating when hungry more rewarding than when almost full. This also holds for negative rewards. Not eating when hungry will give a larger negative reward than not eating when full.

The effect of the variables n and m can be observed in figure 2.1. When $m > n$ the drive increase more rapidly near the optimum (near $\mathbf{h}_t = \mathbf{0}$), while for $n > m$ the drive increase more rapidly further away from the optimum. This in turn affects the rewards received, as a larger change in drive between t and $t + 1$ generates larger rewards.

As explained in [5], for $n > m > 1$ this framework accounts for four important behaviours observed in animals. First, better outcomes are rewarded accordingly. If the change is larger, in the direction of the optimum, then the reward will in turn increase. Secondly, a change in the internal variables has a larger magnitude in drive the further from optima the current state of the agent is. Thirdly, smaller drives have an even further decreased impact on the reward when there exists a drive with a large deviation. One deprived variable has an inhibiting effect on the rewarding effects of less relevant variables. Lastly, the reward function is concave with respect to the change in internal variables which results in risk-aversion, uncertain outcomes are selected against.

With these changes to the RL problem, the agent is now aware of its internal variables. This enables it to learn that it receives rewards differently based on the current state of the internal variables.

An interesting property of the reward function, as here described, is the independence of return from trajectory when $\gamma = 1$. Any two paths taken in the space of $\mathbf{h}$ that have the same start and end point will also necessarily have the same return

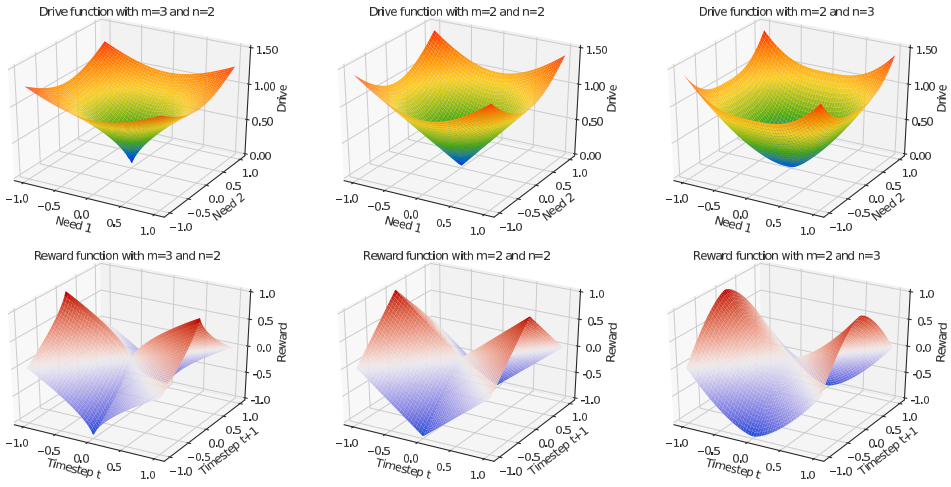

Figure 2.1: Comparison of the drive and reward function for 3 configurations of m and n. The upper row shows the resulting numerical drive as a function of two internal needs. The bottom row shows the reward obtained from a transition from one need level to another.

if the agent considers infinite horizons. Likewise, all paths that end in the same internal state as they started will have a return of zero. It is thus crucial that $\gamma < 1$ for the agent. If it does not matter *when* it receives a reward, it may choose to postpone this event indefinitely.

It is also interesting to note the similarity of the reward function to the form necessary of reward shaping functions [23]. These functions, when added to the reward of a normal RL problem have the effect of not changing the optimal policy. While there is not a one-to-one correspondence between the formulations, this might indicate how a regular RL problem can be combined with a homeostatic regulation problem.

2.3.3 Evaluation

Agents reach terminal states whenever an internal variable exceeds some predetermined allowed range i.e. the death of the animat. However, in the simulations, the agents will be allowed to continue learning in the same environment even after death. This is due to the general instability of current RL approaches and their inability to learn quickly. In general, agents that die as few times as possible are the primary objective, but as the agents are optimising for cumulative reward, this metric might be unfair.

When evaluating RL algorithms, contemporary approaches [24, 7] measure total, undiscounted, accumulated reward. As the homeostatic return after some given amount of time only depends on the start and end point, such a measure will not capture the quality of an agents life. Take for example an agent that spent its life

at the optimal point and an agent that spent its life on the brink of death, just to reach the optimal point before the simulation ended. They would both be assigned the same score, even though the former behaviour is preferred.

A better metric would be to measure how much time the agent spends in its different internal states, with closer to optimal being better. This is, however, not a good metric either. Consider two agents, one which spends its simulation close to death but never dying, and one agent which is at the optimal point for all but a short period of time in which it manages to die. The former agent, while it would have a lower score, is the one to prefer here.

The remedy to these concerns is to punish death in the latter evaluation scheme. It is not clear, though, how death is to be compared to the deviation in internal variables than an agent sustains over time. At one extreme, death is not considered bad, but this was already declared as not being a good solution. At the other extreme, death is considered infinitely worse than any life that an agent can live. For reasons given by nature, this is not necessarily the correct valuation either. An animal with low fitness but a long life is not necessarily preferable to an animal with high fitness who lives a shorter life. The true comparison between these two metrics most likely fall somewhere in between these extremes.

Here, death will be considered infinitely worse than any life an agent can live. This results in a scheme in which two agents performance will first be compared on their number of deaths, with ties being broken by their performance within each life. In practice, the agent will die many times over the course of the simulation and the extra information of how well the agent performed within a life is superfluous. Thus the final metric on which the agent will be compared is the total amount of accumulated deaths.

As a final note, the number of deaths can be linked back to the total reward as it is an approximation of it. Each death will contribute some amount to the total reward that can not be lost. As the agent is reset, it can no longer make up for this loss. In this case, counting the total number of deaths is almost the same measure of performance as counting the total returns. The only differing value is the last life of the agent which will contribute a vanishingly small amount if the number of deaths is large. Another slight difference is contributed by the fact that it is possible to reach different terminal states at different distances from the optimal internal state as the constraints form a hypercube, $-1 > h_i > 1$, and not a hypersphere, $||\mathbf{h}|| < 1$. This, however, speaks in favour of counting the number of deaths as dying is generally thought of as being equally bad, no matter how it is brought about.

2.4 Exploration

Exploration is necessary for RL in order to find an optimal policy as an agent does not know the optimal policy initially [9]. If the agent explores too much, it will

never have time to exploit the policy that it has found. This balancing problem is the *Explore-Exploit dilemma*. In this context, an *explorative* action is defined as taking any non-optimal action while the action with the highest Q-value is the *greedy* action [9].

As mentioned in section 2.2.1, Q-learning employs an exploration strategy that decides which action to take in each state. In the HRL problem the agent also has access to the underlying **h** variables. As such, there is additional information which can be utilised when attempting to balance the explore/exploit dilemma. A successful HRL agent should, therefore, be able to utilise the knowledge of its internal variables to balance its exploitation and exploration, in order to find an even better policy.

Two of the most frequently used exploration strategies are ϵ-greedy strategy [9] and the Softmax exploration strategy [25].

2.4.1 ϵ-greedy exploration

The ϵ-greedy strategy chooses a random action with probability ϵ

$$a_t = \begin{cases} \arg\max_a \ Q(s,a) & \text{if } r \geq \epsilon \\ \text{random action} & \text{otherwise} \end{cases} \tag{2.8}$$

where r is a random number in the interval $[0, 1]$.

The **annealed ϵ-greedy** strategy is an extension to the ϵ-greedy strategy with the addition of an ϵ_Δ which decreased the value of ϵ at each time step. This addresses the explore/exploit problem by initially taking random actions, exploring, and then acting more and more greedily over time.

2.4.2 Softmax exploration

The Softmax exploration strategy, sometimes known as Boltzmann exploration, involves sampling from a probability distribution defined from the Q-values. The probability of action a_i is defined as

$$\pi(a_i|s) = \frac{e^{\frac{Q(s,a_i)}{\tau}}}{\sum_j e^{\frac{Q(s,a_j)}{\tau}}} \tag{2.9}$$

when in some state s and with some temperature τ. The hyperparameter τ has a similar effect to ϵ. As $\tau \to 0$ the policy becomes greedy and when $\tau \to \infty$ the policy becomes random. Note two things in particular. First, the range of values for τ is $(0, \infty)$ while for ϵ it is $[0, 1]$. Secondly, the *relative numerical* values of the Q-function plays a role in the policy.

In the same way that annealed ϵ-greedy is an extension to the ϵ-greedy strategy, annealed Softmax decreases τ over time with τ_Δ.

2.5 Related work

The use of homeostatic variables in RL exploration appears to be a fairly unexplored area of work. That said, there are still two areas that are touched upon in this work that are worth highlighting: exploration strategies and homeostatic agent models.

2.5.1 Exploration strategies

Much research in RL is focused on *efficient* exploration, exploration that minimises costs accrued during learning, or maximising information gain.

Exploration strategies can be split into two broad main categories: undirected exploration and directed exploration. Undirected exploration acts locally, often based on randomness, while directed exploration strategies are identified as acting on global information [26].

Two prominent examples from the undirected category are the strategies presented in section 2.4, ϵ-greedy and softmax exploration. In the directed exploration category, a wide range of heuristics may be found. An illustrative example is that of count-based methods. A count of how many times each state has been visited [27] or how many times each state-action combination has been tested [28] is kept and higher importance is assigned to those with fewer counts.

While the undirected strategies are commonly used in research [10, 24, 7, 25] they are recognised as being suboptimal as compared to directed strategies [26]. However, in this thesis, only undirected strategies are evaluated as these are more commonly used and does not require additional overhead such as counting states.

2.5.2 Homeostatic agent models

Attempting to use RL to maintain homeostasis is not a new idea. Bersini [20] show how an agent can maintain endogenous (cf. homeostatic/internal) variables within predetermined bounds using Q-learning. Likewise, in connection to drives and needs, multiple models have been suggested [16, 17, 18, 19, 29] that balance their multiple objectives using RL.

We stress here that, in this thesis, the specific underlying model, while it may affect the results, are not of too much importance. It is the difference in performance

amongst the exploration strategies that use the *same* underlying model that is important.

3

Methods

The homeostatic reinforcement learning problem consists of two main parts, the agent and the environment. The agent gets input from the environment, selects an action which is then performed in the environment, which in turn generates a new input. While this is similar to a typical RL problem, there is a significant difference: there is no reward from the environment for the agent to maximise. Instead, the agent is tasked with maintaining several internal variables, which may be affected by the environment, within some bounds of optimal values.

In this chapter, we describe the properties of the agent, and the different exploration strategies employed. We then describe a number of evaluation environments designed to highlight different aspects of the explore/exploit dilemma. Finally, we describe the measurements used for evaluation, as well as how the evaluation is conducted.

3.1 Agent

An RL agent typically consists of two components: a Q-value estimator and an exploration strategy, to provide an action a_t for a given state s_t. When converted into an HRL agent, these are extended with two additional components: the internal variables and the drive reduction framework, which can be seen in Figure 3.1. In addition, the HRL agent no longer receives any reward r_t, but instead, observe some change $\mathbf{k}_t$ in the internal variables.

As the internal variables have a direct effect on the reward, these are used as input to the Q-value estimator in addition to the external state s_t, in order to provide accurate estimations. As the internal variables are real values, the state space of the Q-value function becomes infinite. Because of this a deep Q-network (DQN) [10] was used to approximate the Q-value function.

Once the Q-values for each action have been approximated in a given state, an action is selected by the agent based on the exploration strategy. I.e. an ϵ-greedy policy will select a random action with probability ϵ, else it will select the action with the highest Q-value. However, in our HRL agent the strategy has access to the internal

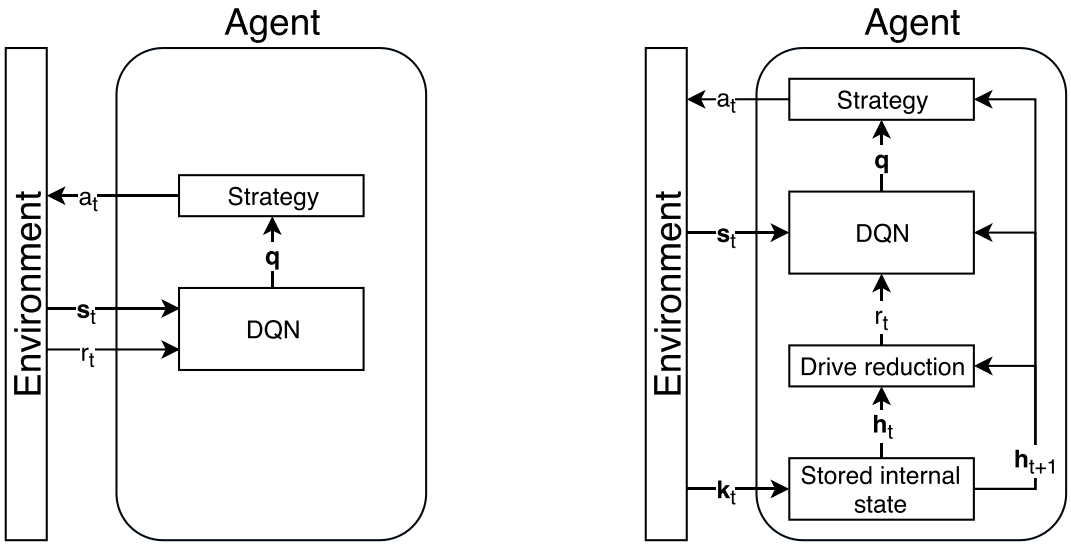

Figure 3.1: Diagrams visualising the difference between a RL agent (left), and a HRL agent (right).

variables, and as such can use more sophisticated exploration strategies.

As the purpose of the evaluations is to evaluate the performance of different strategies, the hyperparameters for the DQN were kept identical for all evaluations. However, due to the complexity difference between environments the number of hidden layers and the number of units in each layer where changed between environments. The hyperparameters and size of hidden layers were found through a grid search using the baseline strategies. See Appendix C for a full specification of hyperparameters.

The agents were randomly initialised in each simulation and shared no memory between simulations. Upon death the environment would reset, the state was recorded as a terminal state, and the simulation would continue.

3.2 Exploration strategies

Two sets of exploration strategies were used in the evaluation: baseline strategies and homeostatic exploration strategies. The baseline strategies are commonly used strategies found in the literature.

Using the internal variables of the agent, we introduce the homeostatic exploration heuristics. These heuristics generate a scalar value, which combined with either ϵ-greedy exploration strategy, or the Softmax exploration strategy, create a total of four new homeostatic exploration strategies.

3.2.1 Baselines

As the homeostatic exploration strategies can be combined with either ϵ-greedy or Softmax, traditional implementations of the same strategies were used as baseline strategies. In total four types of baseline strategies were evaluated: Constant ϵ-greedy, Annealed ϵ-greedy as well as Constant Softmax and Annealed Softmax exploration. In addition, fully random and a fully greedy strategy were included to give some indication of the properties of the environments.

As all the aforementioned exploration strategies have hyperparameters which can greatly affect how well an agent perform, each strategy was evaluated for multiple different settings. The Constant strategies, as the name suggest, had constant ϵ and τ, while the annealed strategies decay with ϵ_Δ and τ_Δ to a minimum of $\epsilon_{\min}$ and $\tau_{\min}$, every time step. A complete list of all baseline strategy hyperparameters can be found in Table 3.1.

3.2.2 Homeostatic exploration

When considering the internal variables of the agent, a great many different exploration strategies can be devised. In this paper, we present two simple heuristics based on the assumptions that one can either explore when you are doing good, or when you are not.

The hypothesis for these strategies is simple. If the agent is doing well it will have the capacity to explore and can manage potential negative effects. However one could also argue that if the agent is not doing good, then the current policy is not good enough and the agent needs to explore in order to find a better one.

Based on these hypotheses we present the Explore When Good (EWG) and Explore When Bad (EWB) exploration heuristics for the HRL problem. The basic implementation of these heuristics can be found in Equation 3.1 and 3.2 respectively. Intuitively EWG will return a higher value when the internal variables are near the optimum level, near 0, while EWB will return a higher value when the internal variables are near the bound of acceptable levels, near -1 or 1.

$$\text{EWG:} \qquad 1 - \max_i |h_i| \tag{3.1}$$

$$\text{EWB:} \qquad \max_i |h_i| \tag{3.2}$$

As $h_i \in (-1, 1)$ these heuristics will always return a value in the range $[0, 1]$. This is beneficial as it makes integration into existing strategies easy, the values can be used either as a probability, or a scale.

In the following sections, the implementations of these heuristics in the ϵ-greedy and Softmax strategies are explained.

Table 3.1: A list of the hyperparameter settings of the baseline strategies. The strategy Constant ϵ-greedy with $\epsilon = 1.0$ is also called the Random strategy.

	Hyperparameters		
Constant ϵ-greedy	ϵ		
	0		
	0.01		
	0.1		
	0.2		
	0.3		
	0.4		
"Random"	1.0		
Annealed ϵ-greedy	ϵ	ϵ_{min}	ϵ_{Δ}
	1.0	0.1	0.001
	1.0	0.1	0.0001
	1.0	0.1	0.00001
	1.0	0.01	0.001
	1.0	0.01	0.0001
	1.0	0.01	0.00001
Constant Softmax	τ		
	0.001		
	0.01		
	0.1		
	1.0		
	10		
	100		
Annealed Softmax	τ	τ_{min}	τ_{Δ}
	0.001	0.0001	0.001
	0.01	0.0001	0.001
	0.1	0.0001	0.001
	1.0	0.0001	0.001
	10	0.0001	0.001
	100	0.0001	0.001

3.2.2.1 Homeostatic ϵ-greedy

As ϵ is a probability, and the heuristics return a value in the same interval, these heuristics can easily be adapted to be used with the ϵ-greedy strategy. This was done by setting the value of ϵ in accordance to Equation 3.3 and 3.4

$$\text{EWG } \epsilon\text{:} \qquad \epsilon = 1 - \max_i |h_i| \tag{3.3}$$

$$\text{EWG } \epsilon\text{:} \qquad \epsilon = \max_i |h_i| \tag{3.4}$$

This creates two ϵ-greedy strategies which are not affected by time, but instead the

current internal state of the agents. This can be seen in Figure 3.2, which shows how ϵ change based on the internal variables, h_1 and h_2, of the agent. There is an issue with these strategies however; both strategies will always explore to some extent and will, in practice, never act optimally.

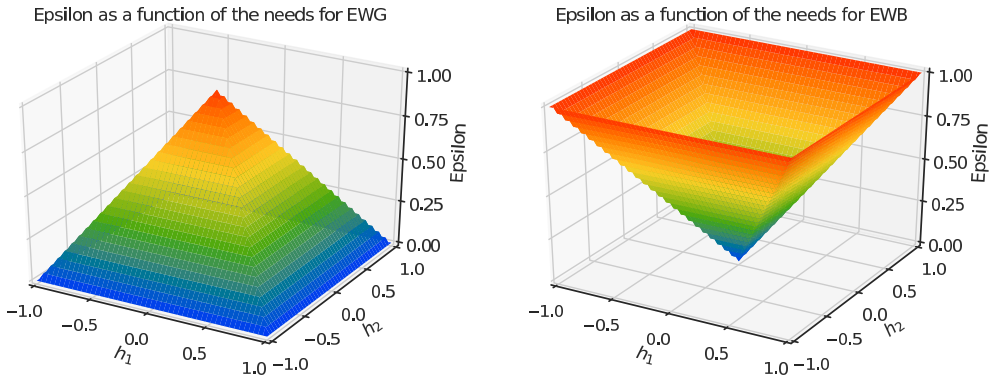

Figure 3.2: The value of ϵ based on Equation 3.3, left, and 3.4, right. EWG acts optimally only at the edge of the acceptable levels, while EWB acts optimally only at the optimal value of the internal variables.

To counter this behaviour the strategies are extended with a simple hyperparameter. The threshold $\theta \in [0, 1)$ parameter change at which level of the internal variables the EWG agent will act optimally, or begin to explore in the case of an EWB agent.

The final equations for the EWG/EWB ϵ-greedy strategies can be seen in Equation 3.5 and 3.6. The threshold parameter in these equations is visualised in Figure 3.3 using two internal variables. Here it is revealed that the threshold indeed makes the agents greedy in the aforementioned intervals.

$$\text{EWG } \epsilon\text{:} \quad \epsilon = \max\left(0, 1 + \frac{\max_i |h_i|}{\theta - 1}\right) \tag{3.5}$$

$$\text{EWB } \epsilon\text{:} \quad \epsilon = \max\left(0, \frac{\theta - \max_i |h_i|}{\theta - 1}\right) \tag{3.6}$$

The homeostatic ϵ-greedy strategies were evaluated with multiple settings of the hyperparameter settings. A full list of all evaluated hyperparameter settings can be found in Table 3.2.

3.2.2.2 Homeostatic Softmax

Applying EWG and EWB on the Softmax exploration strategy is slightly more complicated. Looking back to Equation 2.9, increasing the temperature τ will affect the

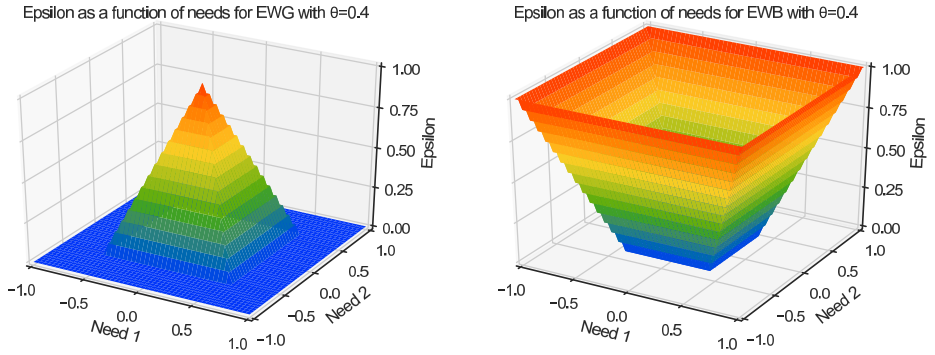

Figure 3.3: The threshold θ makes the EWG strategy act optimally at the edge of optimal values. EWB, however, only starts exploring when the threshold is exceeded, and else it will act optimally.

Table 3.2: A list of the hyperparameter settings of the EWG ϵ-greedy and EWB ϵ-greedy strategies.

	Hyperparameters
EWG ϵ-greedy	θ
	0
	0.2
	0.4
	0.6
	0.8
EWB ϵ-greedy	θ
	0
	0.2
	0.4
	0.6
	0.8

probability distribution generated by the algorithm towards a uniform distribution while decreasing τ will increase the probability for actions with higher Q-values. In the Homeostatic Softmax exploration strategy, the temperature τ is set by the EWG and EWB heuristics. The naïve implementation can be seen in 3.7 and 3.8.

$$\text{Naïve EWG Softmax:} \qquad \tau = 1 - \max_i |h_i| \tag{3.7}$$

$$\text{Naïve EWB Softmax:} \qquad \tau = \max_i |h_i| \tag{3.8}$$

However, in Softmax exploration, the size of τ is not necessarily limited to (0,1). Due to this, the EWG and EWB heuristics are not naturally on the correct scale. In order to find the correct scale the hyperparameter τ_h as is shown in Equation 3.9 and 3.10 is introduced.

Table 3.3: A list of the hyperparameter settings of the EWG and EWB Softmax exploration strategies.

	Hyperparameters
EWG Softmax	τ_h
	0.001
	0.01
	0.1
	1.0
	10
	100
EWB Softmax	τ_h
	0.001
	0.01
	0.1
	1.0
	10
	100

$$\text{EWG Softmax:} \qquad \tau = \tau_h(1 - \max_i |h_i|) \tag{3.9}$$

$$\text{EWB Softmax:} \qquad \tau = \tau_h \max_i |h_i| \tag{3.10}$$

To find a scale of the temperature suitable for the environments in which the strategies were evaluated in, the homeostatic Softmax exploration strategy was evaluated for multiple values of τ_h. A full list of these hyperparameter settings can be found in Table 3.3.

3.3 Environments

In order to properly evaluate the strategies, a number of evaluation environments were required. While the Atari games have become the de facto benchmark environments for evaluating the performance of RL algorithms [10, 30, 31], they are limited to scalar rewards and do not contain a notion of internal variables. To resolve the issue of scalar reward, one could look to Multi-Objective Reinforcement Learning (MORL) problems for which there are environments sets for evaluation [32]. However, neither of these environments support homeostatic regulation and cannot be used to evaluate HRL problems. Instead, a number of custom environments specifically designed for the HRL problems were created using the Open AI Gym framework [33].

A total of three different types of environments: Simple, Gridworld, and Expanding Color Gridworld, were created for evaluation of the strategies. The types were

differentiated by the size of the external input, from no external input to a large external input. Each environment type contained a number of different configurations, resulting in a total of 13 unique environments.

While all environments were different in some aspect they all had some properties in common. All environments were based on the concept of an animat, i.e. the agent had some motors (actions), external sensors, as well as some internal variables, in this case, two: food and water.

The environments did not produce any explicit reward. Instead, each action taken by the agent would reduce the internal needs slightly, simulating energy consumption. In addition performing specific actions in specific states would increase or decrease some internal variables by a larger amount, simulating eating, drinking or performing some detrimental action.

The properties and expectations of each environment are described below. For a full specification of how the environments were balanced, see Appendix A.

3.3.1 Simple environment

The Simple environments were designed to evaluate the ability of the policy to adhere to its explore and exploit behaviour as well as dynamically switch between them. In these environments, the agent is a very simple organism with no external input. The only information the agent has to take decisions upon are the internal needs. From this information, the agent should in each time step decide on one of four different actions. Each action can either give an increase in food or water, a decrease in water or do nothing (which reduce both needs slightly).

A graphical visualisation of the environment can be seen in Figure 3.4. The top left box displays the current levels of the internal needs (food - red, water - blue) of the agent, the levels of which is the only input to the agent. The arrow points at the action last taken by the agent, and the small red and blue bars indicates the outcome of each action 1-4. The goal of the agent is to balance the internal variables as close to 0 as possible, at the current state the agent has eaten a bit too much and is slightly thirsty.

There were three different Simple environments called Greedy, Random and Dual. The Greedy environment was designed to test the ability of strategies to exploit. In this environment, the result of each action is constant and predetermined. That is action 1 always give food, action 2 does nothing, action 3 gives water and action 4 reduces water. The environment is balanced as such that an agent utilising a fully random strategy would either thirst to death by selecting the water reducing action too often or eat itself to death by eating too often. A successful agent has to balance eating and drinking by doing nothing in between, but no noteworthy exploration is needed in order to find the optimal policy.

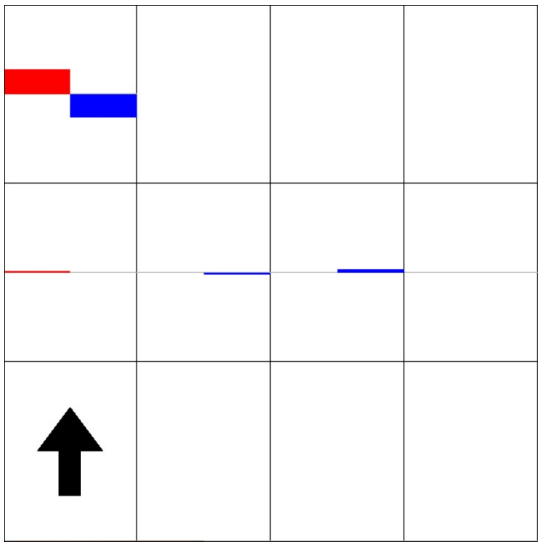

Figure 3.4: A visualisation of the simple environments. The top left bars indicate the current internal variables, smaller is better, the values of which is the only input to the agent. The bars in the middle indicate what payout each action will have, in this state a_1 will give food, and a_2, a_3 will decrease and increase water respectively. The arrow shows the last action of the agent. The agent has 4 different available actions in all states. The agent can only observe the current internal variables and selects among the 4 actions.

The Random environment is slightly different. This environment is specifically designed to test the ability of the strategies to explore. Each time the agent selects an action that increases or reduces any of the needs, that action result is set to a random action. I.e. if the agent selected action 1 and it gave some food, the food resource would move to any action uniformly at random. As such all resources could eventually end up on the same action, or be arranged over the different actions. This world was balanced such that taking random actions would be sustainable, but it was still possible to die if some action was never chosen. In this environment, it would typically not be beneficial to be greedy as the Q-value estimator would not manage to keep up with the constant changes in the environment, and may suggest the same action multiple times in a row. As the same action is not probable to give reward multiple times in a row, this type of behaviour is detrimental. In this environment, an agent must take exploratory actions frequently to survive.

The third Simple environment, Dual, mixed the properties of both the Greedy and the Random environment. The Dual environment switches back and forth between acting like Greedy and Random every few steps. As such, for a strategy to be successful in the Dual environment, it would have to be capable of switching focus between exploring and exploiting dynamically in order to survive. As such this environment tests if the strategy is capable of both exploration and exploitation based only on the internal needs of the agent.

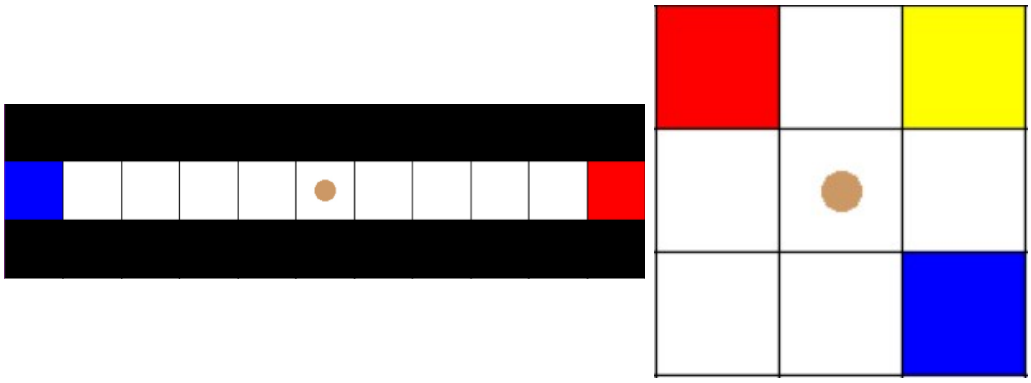

Figure 3.5: Depicted is the short corridor, left, and Gridworld Static Hostile, right. The agent is the brown dot, the red tiles are food, the blue tiles are water, while the yellow tile reduces the water variable.

3.3.2 Gridworld

The Gridworld environments were designed to give very basic external input. The concept of these environments was to evaluate the strategies in slightly more complex environments where multiple actions in a specific order are required in order to survive. The external input to the agent in these environments where the coordinates in the world, as such the agent always knows exactly where it is. In these environments, the agent has to learn which tiles are beneficial at different internal states and how to get to them. A total of six variants of the Gridworld environments were used.

Two of these six environments were corridors, where food and water tiles are located in the opposite ends of the corridor, as can be seen in the leftmost image in Figure 3.5. These environments are designed to punish undirected exploration and test the ability of the strategy to exploit for many steps in succession. The corridors were balanced in such a way that the agent would be required to take many actions of the same type to reach one end of the corridor, and then quickly return to the other end else it would starve or thirst to death along the way. If the agent decides to explore (stand still or move in the wrong direction) more than a few times it would not manage the return trip. The two different corridor environments differ in that one is longer than the other and thus requires more consistent behaviour.

The remaining four environments are 3×3 tile grids, where the food and water tiles are located in opposite corners. An example of these environments can be seen in the rightmost image of Figure 3.5. The different versions of this environment were called Static Safe, Static Hostile, Dynamic Safe and Dynamic Hostile. The safe environments contain only food and water tiles, while the hostile environments have an additional tile which reduces the water variable once entered. These hostile tiles make random exploration more dangerous. In the static environments, the position of all tiles remains fixed throughout the simulation, while in the dynamic environments the food could only be consumed a limited number of times upon

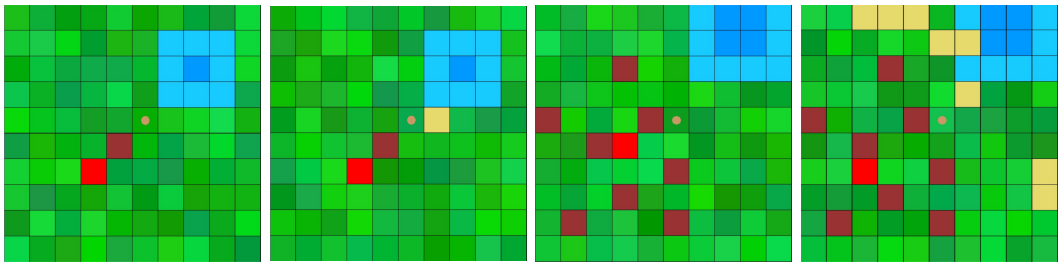

Figure 3.6: The expanding colour gridworlds, from left to right, Static Safe (SS), Static Hostile (SH), Dynamic Safe (DS), Dynamic Hostile (DH). All ECG environments are initialised as the static environments. However, the dynamic environments add additional content over time as the environment expands. In this figure both of the dynamic environments have expanded several times.

which it would disappear and respawn in another tile uniformly at random. In addition to respawning, after each consumption, the amount gained by the agent is lowered. This would force the agent to explore in order to find the new position of the food once fully consumed. As the input to the agent is only the position, and the respawn position is random, each tile would have to be revisited in order to find the new food location, thus promoting random exploration.

3.3.3 Expanding Colour Gridworld

The third type of environments was the Expanding Colour Gridworld (ECG) type. While similar to the Gridworlds, these environments had more complex external input and the size of the environments was much larger.

A total of four different ECG environments were created. Similar to the 3×3 gridworld environments, the ECG environments had four settings: Static Safe, Static Hostile, Dynamic Safe, Dynamic Hostile all of which can be seen in Figure 3.6. However, the implementation was different. All ECG environments were initialised as a 5×5 grid, with a single dark blue water source in one corner, and a single red food source next to a brown tree in the other corner. The hostile environments also contained an additional yellow tile, a desert tile, that reduces the water variable. The remaining tiles were "empty" and would either be a random hue of green or light blue if next to a water source.

Every 10,000 steps the environments would expand, increasing the grid size with one tile in each cardinal direction. This, over time, created a larger and larger area which could be explored. The area was limited in that if the agent selected an action which would make it leave the allowed area, the agent would not move but instead remain in place. The available area is visualised as a black box in Figure 3.7. However, this "border" was not visible to the agent.

In addition, as the environments expanded, the dynamic environments would spawn

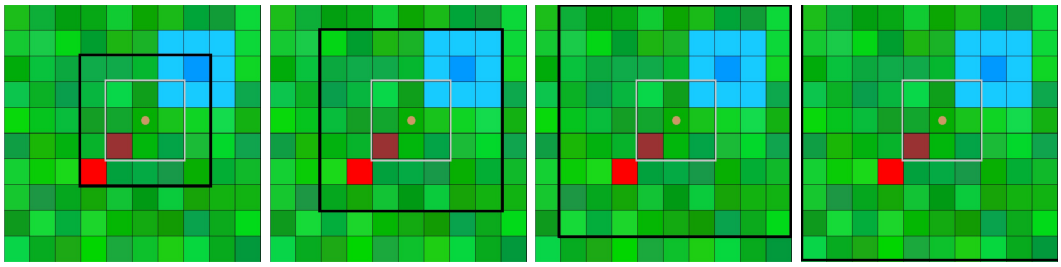

Figure 3.7: The external input of the agent is highlighted as a grey box, and the area available for movement is highlighted as a black box. The available area expanded, in all environments, every 10,000 steps. Notice that the available area is larger than what is visualised in the rightmost figure. Only a part of the environment can be visualised at a time as it is in fact infinite.

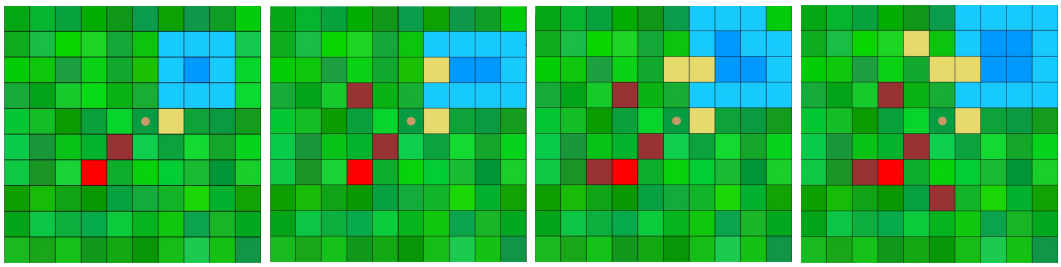

Figure 3.8: The dynamic environments added additional trees and water tiles as the available area expanded, and in ECG DH additional desert tiles were also added. In this figure the first 4 stages, from left to right, of ECG DH is depicted.

additional trees and water sources, and the Dynamic Hostile environment would also spawn additional desert tiles. However, only a single food source would ever exist at any given time. Once consumed this food source would respawn at a random location near one of the trees. The expansion of the dynamic hostile environment can be observed in Figure 3.8. Here, additional trees, water sources and desert tiles are added over time.

The external input of the agent was limited to the RGB colour of the nine tiles surrounding the agent. This was done in order to make the environments more complex than the Simple and Gridworld environments, while also fulfilling what Wilson calls a class 2 environment, i.e. "reliable prediction can no longer be based on the current sensory stimulus and action" [8]. The observation available to the agent is highlighted with a grey box in Figure 3.7.

While reliable predictions can no longer be made on the input, there were still visual cues which the agent could learn. The food tiles always spawned next to a tree tile, and light blue tiles only exist next water tiles. As such the environment contained indicators for the agent which could be used to guide it towards resources once close enough. This type of information was intended to promote guided exploration as well as providing richer input to the agent without expanding the external input size.

The static environments were designed to promote exploitation as the new tiles which were made available over time would contain nothing useful, and it would be hard for the agent to find its way back to the food and water sources if it moved away from them. The dynamic environments, on the other hand, required exploration near the trees to find the food sources, but not elsewhere. As such, it required a balance between exploitation and exploration. The hostile environments were once again used to verify that the agent could balance the different internal variables.

3.4 Evaluation

The baseline and homeostatic exploration strategies, when counting all hyperparameter settings, form a total of 47 unique agents. For each agent, 10 simulations were run in each of the 13 different environments, each simulation lasting a total of 1,000,000 steps. The objective was to find agents which maintained homeostasis throughout the simulations and were capable of balancing exploration and exploitation without human intervention.

While more and longer simulations will increase the significance of the results, the number of simulations and their length is at the upper end of what is achievable by the hardware available for the experiment. Two Intel Xeon Gold 6126 CPUs were utilised and all the simulations took roughly 4 days to run.

To measure these properties two different metrics were used: the number of deaths, and the rate of exploration during a simulation. The number of deaths was the primary metric as one can not set an optimal value for the rate of exploration; as long as homeostasis is maintained, any rate of exploration is acceptable. However, some environments may require more or less exploration and as such the agent must be capable of adapting to this fact.

The average number of deaths for each agent in each environment was normalised according to Equation 3.11 to generate a score. This method is a modification of the normalisation method used in *Deep Reinforcement Learning with Double Q-Learning* [24], where the human level performance is set to 0 deaths. The result is a score for each agent and environment where 0 is the average performance of the random agent, and 1 is the optimal possible performance. A negative score indicates worse than random performance.

$$\text{score}_{\text{normalised}} = \frac{\text{deaths}_{\text{agent}} - \text{deaths}_{\text{random}}}{0 - \text{deaths}_{\text{random}}} \tag{3.11}$$

The ideal agent should be able to perform well in many different environments with the same hyperparameters, as this would indicate that the agent is capable of adapting to different environments. To evaluate this property, a total score was

calculated for each agent according to Equation 3.12, where a score of 0 is equal to the average performance of the random agent, and 1 is equal to not dying a single time in any of the simulations. This is also why the normalised scores serve a purpose, it allows for comparison between environments. A high value indicates that an agent performed well in many environments, or exceptionally well in some and poorly in others.

$$\text{score}_{\text{total}} = \frac{\sum_{\text{environment}} \text{score}_{\text{normalised}}}{\#_{\text{environments}}} \tag{3.12}$$

The average rate of exploration was used to verify that the agents adapt their behaviour depending on the environment. This was found by dividing the total number of exploratory actions with the total number of steps in a simulation. An exploratory action is defined as taking a non-optimal action [9], i.e. an action which does not have the highest Q-value.

4

Results

The full results of the evaluations can be found in Appendix B. The results presented in this chapter are summaries.

The results chapter is divided into three major parts. First, each environment category is analysed in order to verify its correctness. Five well-performing strategies are selected in each category and are compared to the random baseline. The difference between the strategies and the random baseline is then analysed, giving insight into whether the agent is capable of learning in the environments. Furthermore, it is possible to discern whether the strategies contribute to the performance of the agent.

Secondly, the strategies are compared between themselves in order to arrive at a ranking. Calculating the normalised score for each environment produces a ranking for each environment. By averaging the normalised score over environments, a final rank is reached, which allows the strategies to be compared.

Lastly, the rates of exploration between the strategies are compared. This is done in order to gain an insight into the dynamic behaviour of the homeostatic exploration strategies.

4.1 Environments

Each environment was created with some properties in mind. Here, we tie back to this and verify that the environments produce sensible results that are of use.

4.1.1 Simple environments

Observe in Figure 4.1 that, for all the simple environments, the strategies outperform the Random strategy. In the Random and Dual environments there does not seem to be any but linear rates of death. The agents perform equally good early in the simulation, as compared to late. It does not seem possible for the strategies to learn

a sustainable policy for these environments.

In the Random environment, if we look at all the strategies, we find that many of them performed worse than the Random strategy. However, there are strategies which do find policies that work better. In particular, the best performing EWG Softmax strategy manages the near-perfect score of 0.8, closely followed by the lowest threshold settings of EWG ϵ and EWB ϵ at a score of 0.72-0.78.

In the Greedy environment, some strategies perform *worse* over time. There are also those, EWB Softmax and EWG Softmax for example, that manage to survive for the whole simulation. Despite the static nature of the environment, survival is not guaranteed.

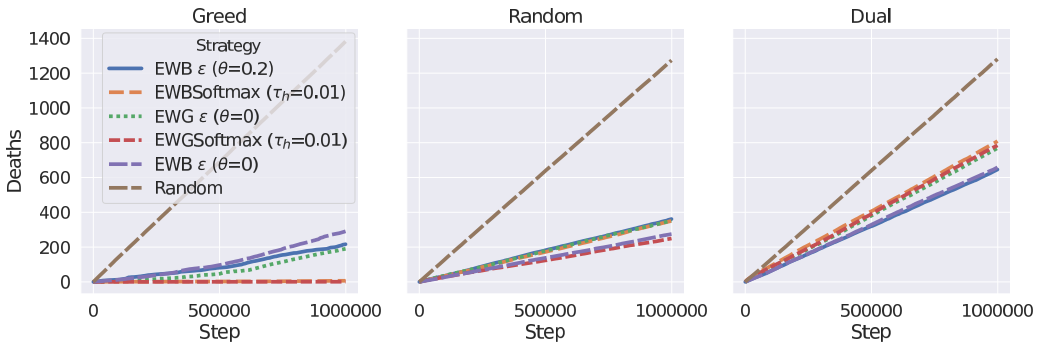

Figure 4.1: The average number of deaths achieved by five well performing strategies and the random strategy in the Simple environments.

4.1.2 Gridworld environments

In both static instantiations of the Gridworld environments, it is clear from Figure 4.2 that there is a large difference between the Random baseline and agents. For both the static safe and the static hostile version, strategies which survive near indefinitely can be found. Note that the hostile version doubles the number of deaths for the random strategy, showcasing its inability to learn the adverse effects of stepping on detrimental tiles.

The dynamic safe and dynamic hostile environments show less of a difference when comparing against the random baseline. The dynamic nature of the environment requires unlearning of old knowledge, which is not directly facilitated by the used RL architectures. Despite this fact, there does exist a noticeable gap in the performances. The best strategy from Figure 4.2 performs twice as good as the random baseline, indicating that it is possible to learn something in the environment.

By viewing the normalised scores, it becomes clear that the Gridworlds manage to test different things. Static hostile and static safe share results to some extent,

something which is less true for dynamic safe and dynamic hostile, but in general, the results vary significantly.

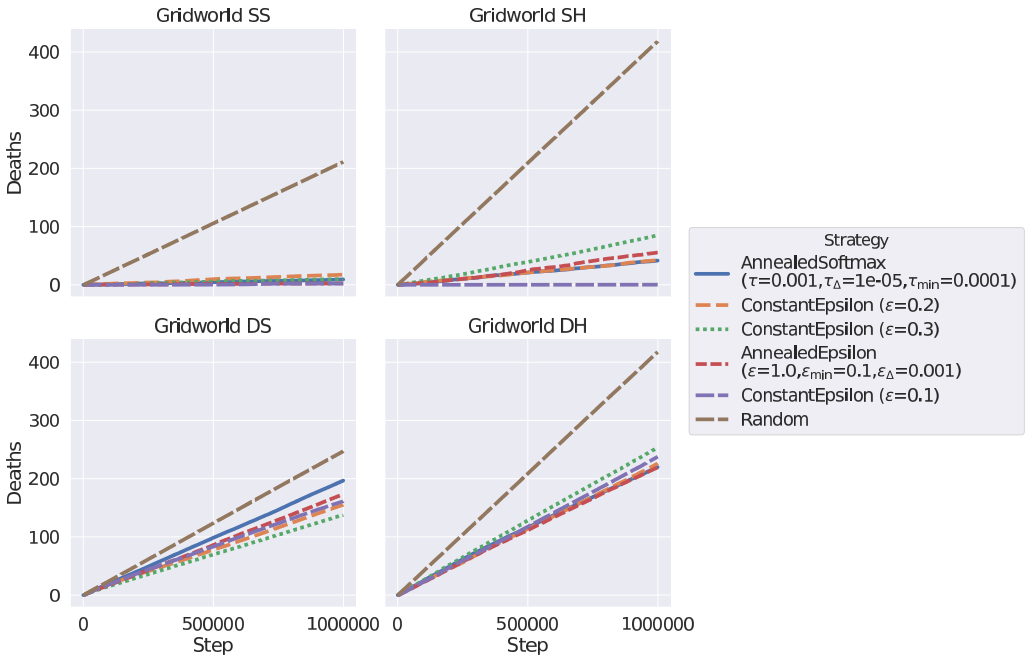

Figure 4.2: The average number of deaths achieved by five well performing strategies and the random strategy in the Gridworld environments.

The Corridor gridworlds are very harsh in comparison to other environments. In Figure 4.3 the random baseline performs worse than the selected strategies to such a degree that any variation in strategy performance is lost. Looking closer at the final score for these two worlds we can see that the baseline strategies perform well here. All three baseline strategies, ConstantEpsilon, AnnealedEpsilon, Softmax and AnnealedSoftmax have multiple scores above 0.9. In contrast, the EWB ϵ strategy struggles here, and EWG ϵ falls in between. This ties back into the property that the corridor environments should test. Too much exploration should be punished.

4.1.3 Expanding Color Gridworld environments

From Figure 4.4 the same result as for the corridor environments is observed in the Expanding Color Gridworlds. The selected strategies convincingly outperform the Random baseline. Interestingly, the static versions are more difficult to survive according to the Random baseline. Most likely owing to the increased amount of water in the dynamic worlds.

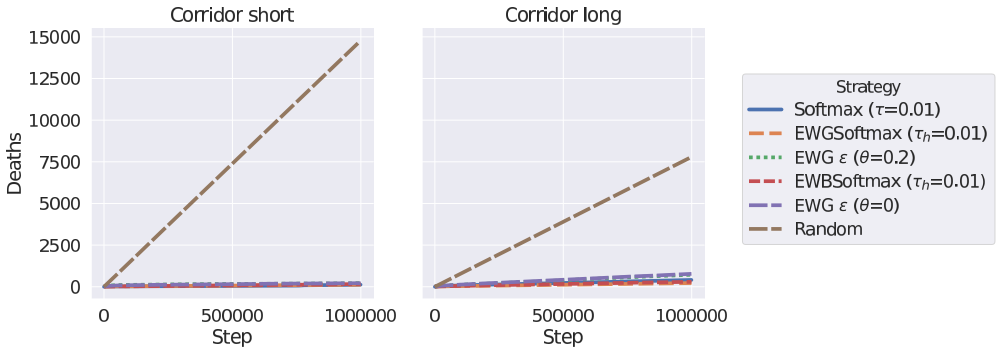

Figure 4.3: The average number of deaths achieved by five well performing strategies and the random strategy in the Corridor environments.

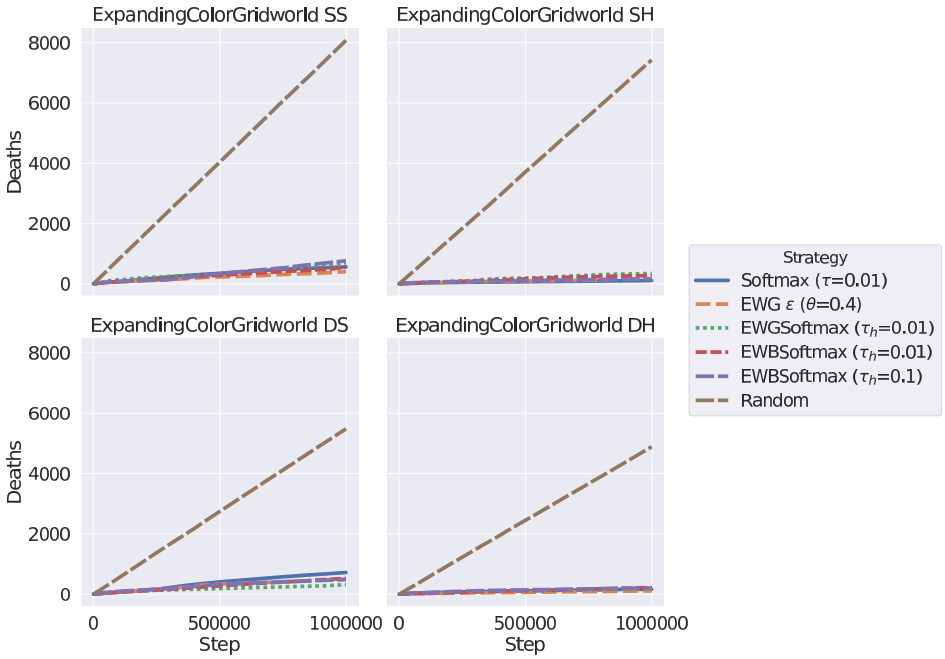

Figure 4.4: The average number of deaths achieved by five well performing strategies and the random strategy in the Expanding Color Gridworld environments.

4.2 Combined results

The final combined score across all environments can be seen in Figure 4.5. The strategies that take into account the internal variables outperform those that do not. At rank 10 we find the first non-homeostatic strategy, Constant $\epsilon = 0.4$. The first Annealed ϵ is found at rank 22.

The top strategy, EWG ϵ ($\theta = 0.2$), has a score of 0.77. The only environments

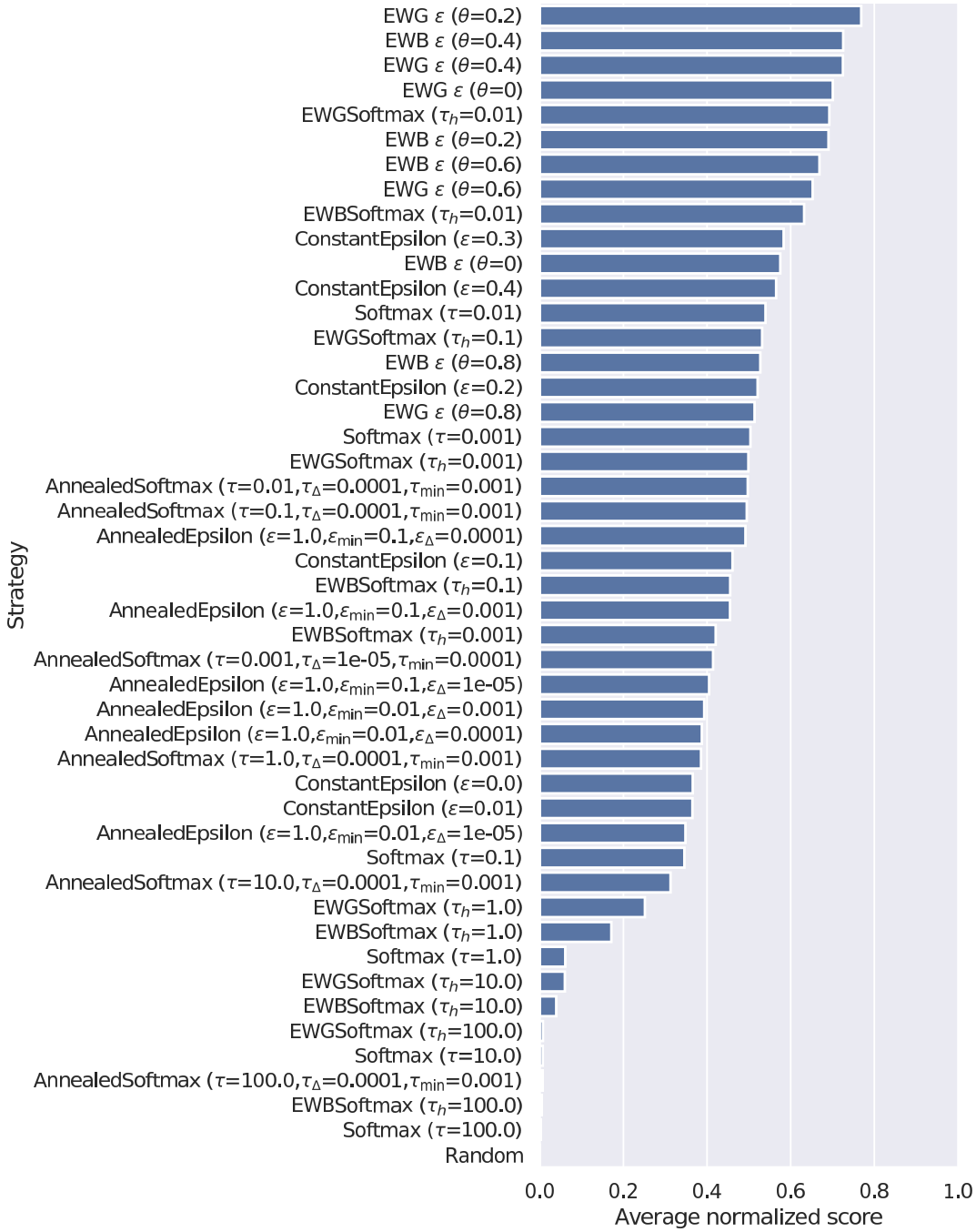

Figure 4.5: Averaged normalised scores for each 47 strategies tested. The first baseline strategy, ConstantEpsilon ($\epsilon = 0.3$), ranks number 10. The first AnnealedEpsilon is found at rank 22.

where it is not a top contender are the Dual and Random environments, but even there, it is only beaten by other homeostatic strategies. This is also in line with what is to be expected. The Random and Dual environment punishes greedy actions, and EWG will be primarily greedy in a world it performs poorly in.

Interestingly, we see a multitude of θ values in the top ranks. For EWG ϵ, the lower values of θ seem to be favoured, while for EWB ϵ, the mid-range values appear to be favourable. In general, the performance of these strategies seems to not depend too much on the value of θ.

If EWG and EWB are compared, we find no conclusive evidence towards either heuristic being better or worse than the other. The top two spots are claimed by both strategies respectively, and in the top 10, there are 4 EWG strategies and 3 EWB strategies.

Looking at the per environment score of the top performing hyperparameter settings of each strategy type, Figure 4.6, it is clear to see that all strategies struggle with the dynamic Gridworlds. For the static Gridworlds, the results are almost equal among all strategies, apart from the EWG Softmax, EWB Softmax and Softmax baseline, which all appear to fail. This appears to be due to the τ value being too high for this specific environment. In contrast, Softmax ($\tau = 0.001$) receives a score more similar to the Annealed Softmax and ϵ strategies.

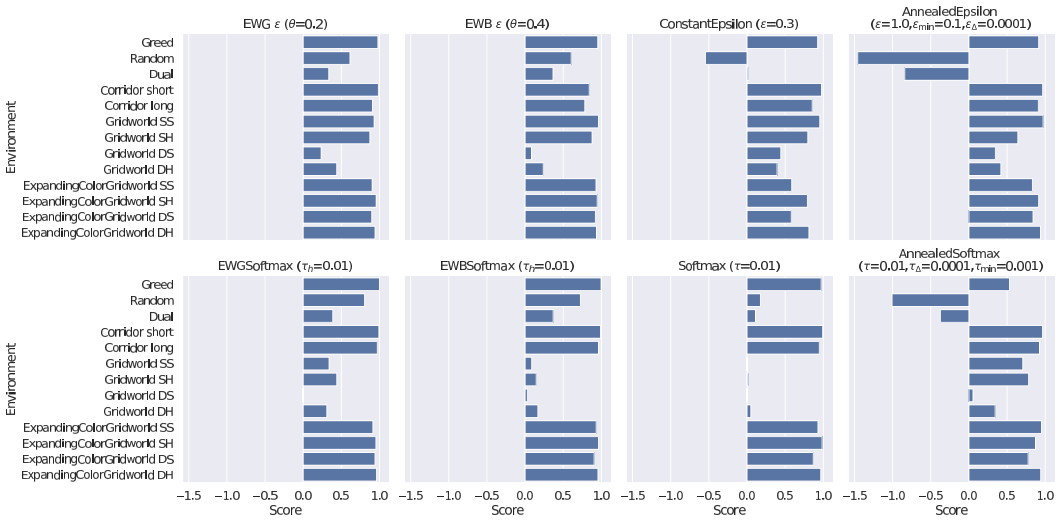

Figure 4.6: Normalised scores for the top performing strategy of each type in each environment, using the Random agent as baseline. The top row contains ϵ-based exploration strategies, and the bottom row contains Softmax exploration strategies.

For the Random and Dual environments, the baseline strategies all struggle and sometimes fail miserably, while the EWG and EWB strategies perform fairly well. This, however, does not appear to be related to the hyperparameters as almost all baseline strategies have a zero or negative score in these environments.

In the Expanding Color Gridworld environments, all strategies perform well. However, the EWG and EWB strategies do perform better than their baseline alternatives.

If we compare the results of the ϵ-greedy strategies with those of the Softmax strategies we see that, in general, the ϵ-greedy strategies seem to perform better. Do note

that the homeostatic Softmax strategies outperform the non-homeostatic baseline strategies, in general.

4.3 Exploration comparison

One of the anticipated effects of the homeostatic strategies is their dynamic exploration rates. In Figure 4.7 we observe the fraction of non-optimal actions taken in each environment. Only the top-performing hyperparameter configuration of each strategy type is illustrated. In addition, the 95% confidence interval is displayed as a black line.

4.3.1 ϵ-based strategies

What we observe from Figure 4.7 is a constant rate of exploration from the Constant ϵ and Annealed ϵ strategies. While there is a small deviation in the averages between environment types, this is explained by the differing number of actions available.

The homeostatic strategies, in contrast, have wildly varying exploration rates. Exploration rates between environments differ with up to 40 percentage points for EWB and 38 for EWG. Moreover, exploration rates within an environment differ. For example, in the static Gridworld environments EWG has a very high variance indicating that each simulation had a different exploration percentage, and the rate of exploration is much higher than for the dynamic Gridworld environments.

Overall, EWG ϵ ($\theta = 0.2$) explores much more than EWB ϵ ($\theta = 0.4$), even though they are both ranked number one and two respectively. For example, in the static Gridworld environments, both EWG ϵ ($\theta = 0.2$) and EWB ϵ ($\theta = 0.4$) have roughly the same scores: 0.87, 0.93 and 0.88, 0.96. Their exploration rates are however very different: 43%, 49% and 4%, 1%.

We also find that the rate of exploration of the top performing homeostatic strategies in the Random environment, an environment in which only the homeostatic strategies were successful, is increased (as compared to other environments), with rates between 30-75%. However, it appears that the rate of exploration by itself is not the key to success, as the baseline Softmax strategies have similar rates in this environment but still fail to find a policy that works and perform on par with the Random baseline strategy.

4.3.2 Softmax strategies

When compared to the ϵ-based strategies, the Softmax strategies exhibit the opposite exploratory behaviour. In Figure 4.7 we see that the Annealed Softmax strategy

has the most difference in exploration, both across environment types and within. Following the Annealed Softmax strategy is EWB Softmax, which also displays some variance in behaviour. Both Softmax and EWG Softmax, on the other hand, have similar rates of exploration across environments.

Despite the dissimilarities, both EWG Softmax and EWB Softmax are placed high in the ranking with spot 5 and 9. The Softmax and Annealed Softmax, dynamic behaviour aside, have rank 13 and 20 respectively.

One thing worth pointing out for the Softmax strategies is the Corridor environments. For all four strategies, the rate of exploration as compared to the other environments is systematically lower.

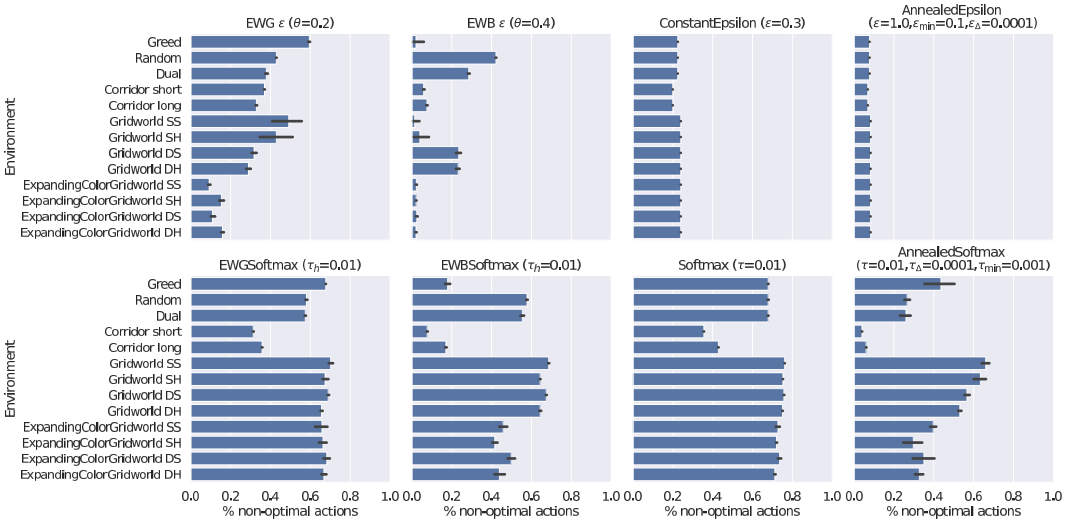

Figure 4.7: The percentage of exploratory actions taken in each environment for the best performing hyperparameter configuration of each strategy type. The top row contains ϵ-based exploration strategies, and the bottom row contains Softmax exploration strategies. Note the near constant amount of exploration done by constant ϵ and annealed ϵ across environments as compared to EWG and EWB. The slight variation in exploration is caused by the differing number of actions in the environments. The black lines indicate the 95% confidence interval, indicating the variability of the exploration rates within an environment.

5

Discussion

The animat path to AGI naturally involves simulating animats. A general objective for the animat is to regulate its internal state, to maintain homeostasis. This objective can, in turn, be computationally solved through the application of reinforcement learning. When such a solution is attempted, the RL problem finds itself with extra information not normally included in RL problems. This raises the question: By utilising this extra information, can one construct RL solutions that include this information, and will it perform better than traditional solutions?

We have answered these questions for the RL component of exploration strategies. Through 8 unique exploration strategies and 47 total hyperparameter configurations in 13 environments, the results indicate that homeostatic exploration strategies perform better than traditional RL strategies. The internal variables made available through the problem formulation encode relevant information to the solution of the HRL problem.

While the agents are unable to find sustainable policies for all environments, we see that they are capable of finding policies which provide significant improvement compared to the Random agent. The dynamic environments, in particular, are hard for the DQN algorithm to find policies for. However, as real-world environments can be dynamic, this type of environment is still relevant for evaluations. In addition, we find results that indicate that, while a DQN may struggle with this type of environments, the choice of strategy may affect how well the agent can perform in highly dynamic and stochastic environments.

In this chapter, we highlight certain properties of the homeostatic exploration strategies, insights that can be found, drawbacks to the homeostatic strategies proposed as well as future avenues of research.

5.1 Dynamic exploration

There are multiple instances of strategies that have an equal score but a vastly different fraction of exploratory actions. There are also cases of which the opposite is true, equal rate of exploratory actions but different scores.

In general, the fraction of exploratory actions does not seem to be a good indicator of the score. We are thus rather led to believe that it matters more *when* a strategy chooses to explore rather than *how much*.

This becomes apparent in the simple environment Random where continuous exploration is necessary. In this environment the baseline strategies are outperformed even by the random strategy, while several EWG and EWB strategies perform fairly well, finding a good policy in an environment which it was not assumed to be possible if one were only looking to traditional strategies. This appears to be achieved by increasing the rate of exploration *at the right time*, which these strategies do dynamically without modifying the hyperparameters.

5.2 Homeostatic ϵ hyperparameter robustness

Hyperparameter sensitivity is an important practical property of any algorithm. Only a finite amount of resources and time can be spent in finding good values for the hyperparameters.

From the results, we find the homeostatic ϵ strategies high in the ranking for a large range of θ values. This would seem to indicate their robustness and practical use. Even if the Random and Dual environments, which contribute a large negative factor to the baseline's performance, were to be removed, this still holds true. There are still some hyperparameter choices that seem to work less well for the Homeostatic ϵ strategies. The outer edges of the intervals, for example EWB $\theta = 0, \theta = 0.8$ and EWG $\theta = 0.8$ have less convincing performances. In general, the inner parts, $0.2 \leq \theta \leq 0.6$, seem to be the safer values in regards to overall performance.

5.3 Softmax strategy performance

The Softmax exploration strategies have a fundamental difference in comparison to the ϵ exploration strategies. They are able to consider how *good* each action is, at least according to the learned Q-function. We might thus expect to see them perform better than the ϵ exploration strategies. However, this is not the result that we observe. On the contrary, the Softmax strategies seem to consistently perform worse than their ϵ counterparts.

How come that, despite their extra potential, the Softmax strategies find themselves with lower scores? While it is hard to point to a specific reason, we can highlight some potential ones.

First of all, Softmax exploration is known to be hard to tune [34]. From the results, we find that $\tau = 0.01$ works better for the Softmax strategies, with both the lower

step $\tau = 0.001$ and the higher step $\tau = 0.1$ having less score. In the experiments that have been performed, only a crude gridsearch has been used, and it is fully possible that we might have found better scoring policies in the interval $(0.1, 0.001)$. This can be seen as a flaw in the experiments, or it can be seen as a flaw in the Softmax hyperparameter sensitivity.

We also have to tune the Softmax strategies to a multitude of environments. If the hyperparameters are very sensitive to one environment, finding a good parameter value for many environments might be impossible. This becomes especially clear when each environment has different relative values for the Q-function. Attempting to find a single τ that generalises across all of them might then be a fruitless endeavour.

5.4 EWG vs. EWB

It is interesting to ask the question of whether it is better to explore when in good health or when in poor. When one is in good health, one has the resources to explore, but not an immediate benefit. On the opposite, when in poor health one may not have the resources to do so, but one does have an immediate need for it.

From our results, we see that there is no obvious answer to such a question. The top two strategies are from each type. Such a result can by no means be seen as conclusive, but if taken at face value it leads to the next logical question; If both EWG and EWB work, can we combine their positive properties somehow? We postpone such a discussion to section 5.6.

5.5 Caveats

Assuming there is an optimal policy that will keep the agent at the optimal state, the EWG strategy will be able to find, but not follow, this optimal policy. This is because the closer the agent is to the optimal state, the more exploratory actions the agent will take. To counter this behaviour, in theory, many extensions could be made, e.g. annealing a maximum ϵ value or the threshold θ could be increased. These changes should make the agent more greedy in general, as well as allowing the agent to follow the optimal policy to a larger extent, but they come at a price. Increasing the number of hyperparameters, or letting them decay over time, comes with additional hyperparameter tuning.

Another issue is the initialisation of the simulations. If the EWG agent is initialised with an internal variable far from the optimum, the agent will have a very low rate of exploratory actions and will act greedily. If the underlying Q-value estimator has not yet found a good enough policy the agent will simply act greedily towards a

poor policy and the agent may fail to improve it. This issue was sometimes observed in the Corridor environments, where the internal variables would drop too quickly for the agent to ever take enough random actions to encounter a food or water source, resulting in the agent simply standing still. To counter this we can use the same idea as before, the θ could be decreased initially. For example, an annealed ϵ approach could be applied to the θ value to increase the rate of exploration early in the training phase, if this issue is encountered. But again, this requires human intervention, something we wish to minimise.

A concern with the EWB strategy is that it can potentially make the agent unstable when it reaches the threshold θ. When an EWB agent is performing badly it will start performing more exploratory actions, in which case it may perform even worse, resulting in even lower internal variables and a higher ϵ. As such, once the agent reaches the threshold, it may be unable to return to a better state and instead follows a downward spiral, resulting in its demise.

This drawback of EWB might also give us a hint on the question of whether the EWB or EWG heuristic is better overall. We are ultimately interested in agents that manage to not die at all in their environments. If the concern of EWB spiralling to its own death holds true, we might expect to see the EWG heuristic perform better in a more realistic setting.

5.6 Future work

During the development of this paper, a number of possible improvements and alternative approaches were found, but not fully investigated. In this section, we introduce some of these ideas as potential future work.

5.6.1 Exploration heuristics and extensions

We have tested two exploration heuristics, EWG and EWB. They both map from the space of internal variables onto the interval $[0, 1]$ with a hyperparameter θ to control the shape of the function. It is possible that the space of such functions contains exploration heuristics that have an even greater performance. We noted previously that both EWG and EWB seem to be reasonable heuristics when it comes to performance. Perhaps it would be reasonable to consider an exploration heuristics that *explore when good or bad*, which is greedy in between these internal states.

Another idea is to extend the current heuristics with additional hyperparameters. An example of this is ϵ_{min} and ϵ_{max}. These parameters can be used to counter some of the caveats alluded to in section 5.5. For example, ϵ_{max} could be used to mitigate, to some extent, the potential death spiral of EWB, and ϵ_{min} could be used in EWG to force the agent to explore early.

The equations including ϵ_{min} and ϵ_{max} can be found in Equation 5.1 and 5.2.

$$\text{EWG } \epsilon\text{: } \epsilon = \max\left(\epsilon_{\text{min}}, \left(1 - \max_i |h_i|\right) \cdot \left(\epsilon_{\text{max}} - \frac{\epsilon_{\text{min}} - \theta \cdot \epsilon_{\text{max}}}{1 - \theta}\right) + \frac{\epsilon_{\text{min}} - \theta \cdot \epsilon_{\text{max}}}{1 - \theta}\right) \tag{5.1}$$

$$\text{EWB } \epsilon\text{: } \epsilon = \max\left(\epsilon_{\text{min}}, \left(\max_i |h_i|\right) \cdot \left(\epsilon_{\text{max}} - \frac{\epsilon_{\text{min}} - \theta \cdot \epsilon_{\text{max}}}{1 - \theta}\right) + \frac{\epsilon_{\text{min}} - \theta \cdot \epsilon_{\text{max}}}{1 - \theta}\right) \tag{5.2}$$

The minimum and maximum values $0 \leq \epsilon_{\text{min}} < \epsilon_{\text{max}} \leq 1$ provide the ability to set hard limits on the level of exploratory actions. The effects of these hyperparameters are visualised in Figure 5.1 using two internal variables. Here ϵ is a linear function from ϵ_{min} to ϵ_{max}.

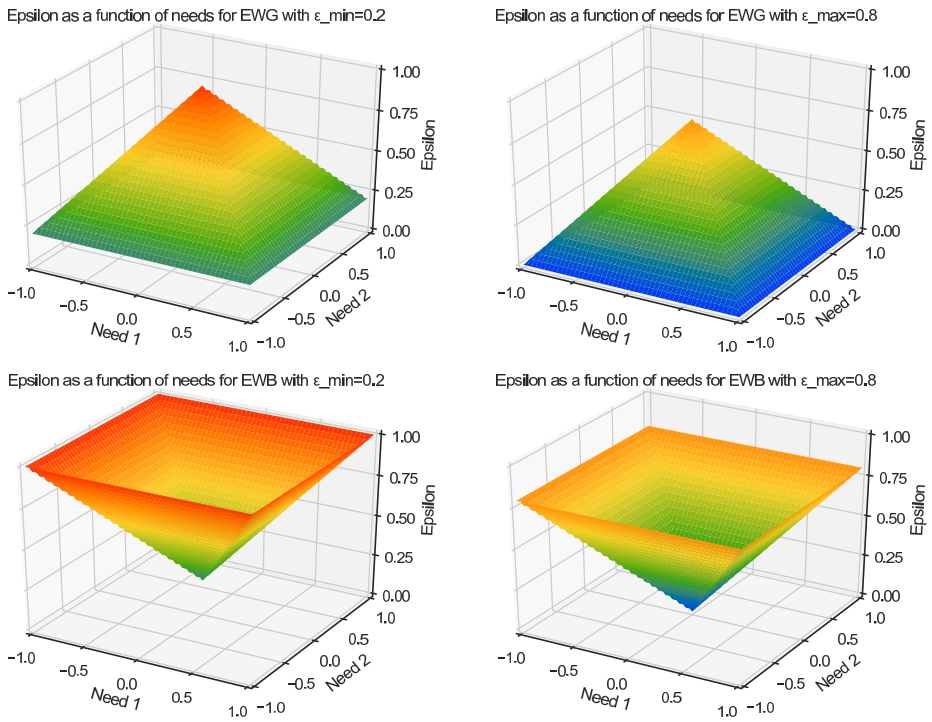

Figure 5.1: ϵ scales linearly between the ϵ_{min} and ϵ_{max} bounds.

These parameters can also be used in tandem with the threshold θ for more fine-grained control, i.e. Figure 5.2 show a risk-averse setting, where both a threshold and a maximum ϵ is used to have the agent explore only a little when it is doing well. The original implementation can be recreated by setting the hyperparameters to $\theta = 0$, $\epsilon_{\text{min}} = 0$ and $\epsilon_{\text{max}} = 1$.

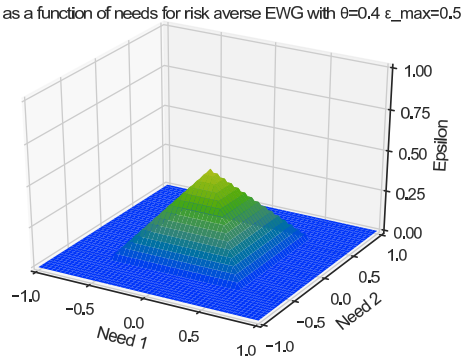

Figure 5.2: An example of a risk averse strategy which explores at a reduced rate and only when internal variables are near optimal.

5.6.2 Sustainability-based exploration

In the Homeostatic Reinforcement Learning problem we can reason about what the optimal Q-values should be. An agent with an optimal Q-function, with all its internal variables at optimal levels, should not predict any positive return, no matter the action. The best achievable return in such a state is in fact zero. This can be observed in the second row of images in Figure 2.1. Here, when the internal variables at time step t are 0, the reward for all possible values of the internal variables in $t + 1$ are negative or zero.

While larger than zero Q-values may still be found during learning and may persist indefinitely due to approximation and other errors, the Q-function should eventually be bounded from above by zero in the optimal case.

Likewise, we can interpret the Q-function in our Homeostatic setting. If the return in a given state is above or equal to zero it indicates that it is a step that will eventually come closer to the optimum, and the agent believes that it is sustainable. On the other hand, if a Q-value is less than zero it indicates that this action will eventually end up further away from the optimum. This indicates that the Q-values can be seen as a measure of the agent's beliefs of its own sustainability.

With this in mind, we can form a class of strategies with the same hypothesis as for EWB, which only explores when the agent believes itself to have no sustainable action. In other words, exploration only occurs when all expected returns, Q-values, are strictly less than zero.

We can define multiple policies which build on this idea by simply not exploring if there exists a Q-value larger than or equal to zero, indicating that the given policy is sustainable, and using one of the previous exploration strategies when all Q-values are less than zero.

The pseudocode for the sustainability-based exploration strategy is given in Algo-

rithm 2. Note that a separate exploration strategy is needed.

Algorithm 2 Sustainability-based strategies

Require: Exploration strategy STRATEGY
input function Q, internal variables $\mathbf{h}$, current state s,
  if $\exists a : Q(s, a) \geq 0$ **then**
    $a \leftarrow \arg\max_a Q(s, a)$
  else
    $a \leftarrow \text{STRATEGY}(Q, \mathbf{h}, s)$
  end if
output a

A small number of trials were performed with this strategy, however, the results were negative. This may be due to the tendency of the DQN to overestimate the Q-values [24]. As such, improvements to the DQN algorithm [7] may improve the results of the sustainability-based strategy. However, trials with DDQN [24] did not show promising results either.

Rather than use the Q-function as a measure of sustainability, it is also possible to use the agent's history. A downward trend in the internal variables signifies unsustainable behaviour and an upward trend signifies sustainable behaviour. There are a few considerations to take into account here though. First, it is not clear how to define sustainability. How far back into history one looks and how to compare different needs need to be answered. Does an upward trend in one variable cancel a downward trend in another? Secondly, there may well exist environments that require the agent to temporarily deviate from its internal state. For example, a bear in hibernation. To then assign significance to such an event may be incorrect, as the bear most likely has a belief in its ability to find food once spring arrives.

6

Conclusion

We find that replacing the typical ϵ-greedy strategies with either EWG ϵ or EWB ϵ can improve performance of the HRL agent. Agents implementing these strategies appear to be able to find better policies. In addition, agents using the new strategies appear better equipped to handle dynamic and stochastic environments without human intervention or hyperparameter tuning.

While the Softmax strategies did perform fairly well for some settings in some environments, we did not see any significant improvement in the performance with the EWG and EWB heuristics, when compared to baseline Softmax implementations. As such implementing the EWG and EWB heuristics as a Softmax exploration strategy does not seem worthwhile as it does not significantly improve performance.

Bibliography

[1] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[2] Cassio Pennachin and Ben Goertzel. “Contemporary approaches to artificial general intelligence”. In: *Artificial general intelligence*. Springer, 2007, pp. 1–30.

[3] Stewart W Wilson. “Knowledge growth in an artificial animal”. In: *Adaptive and Learning Systems*. Springer, 1986, pp. 255–264.

[4] Robert C Wilson et al. “Humans use directed and random exploration to solve the explore–exploit dilemma.” In: *Journal of Experimental Psychology: General* 143.6 (2014), p. 2074.

[5] Mehdi Keramati and Boris Gutkin. “Homeostatic reinforcement learning for integrating reward collection and physiological stability”. In: *Elife* 3 (2014), e04811.

[6] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[7] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[8] Stewart W. Wilson. “The Animat Path to AI”. In: *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats*. Paris, France: MIT Press, 1990, pp. 15–21. ISBN: 0-262-63138-5. URL: http://dl.acm.org/citation.cfm?id=116517.116519.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[10] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), p. 529.

[11] Douglas S Ramsay and Stephen C Woods. “Clarifying the roles of homeostasis and allostasis in physiological regulation.” In: *Psychological Review* 121.2 (2014), p. 225.

[12] Kelvin JA Davies. “Adaptive homeostasis”. In: *Molecular aspects of medicine* 49 (2016), pp. 1–7.

[13] Peter Sterling. “Allostasis: a model of predictive regulation”. In: *Physiology & behavior* 106.1 (2012), pp. 5–15.

[14] C. Liu, X. Xu, and D. Hu. "Multiobjective Reinforcement Learning: A Comprehensive Overview". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.3 (Mar. 2015), pp. 385–398. ISSN: 2168-2216. DOI: 10.1109/TSMC.2014.2358639.

[15] Diederik M Roijers et al. "A survey of multi-objective sequential decision-making". In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 67–113.

[16] Mohamed Oubbati, Christian Fischer, and Günther Palm. "Intrinsically motivated decision making for situated, goal-driven agents". In: *International Conference on Simulation of Adaptive Behavior*. Springer. 2014, pp. 166–175.

[17] George Dimitri Konidaris and Gillian M Hayes. "An architecture for behavior-based reinforcement learning". In: *Adaptive Behavior* 13.1 (2005), pp. 5–32.

[18] Varun Raj Kompella, Sohrob Kazerounian, and Jürgen Schmidhuber. "An anti-hebbian learning rule to represent drive motivations for reinforcement learning". In: *International Conference on Simulation of Adaptive Behavior*. Springer. 2014, pp. 176–187.

[19] George Konidaris and Andrew Barto. "An adaptive robot motivational system". In: *International Conference on Simulation of Adaptive Behavior*. Springer. 2006, pp. 346–356.

[20] Hugues Bersini. "Reinforcement learning for homeostatic endogenous variables". In: *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*. 1994, pp. 325–333.

[21] Mehdi Keramati and Boris S. Gutkin. "A Reinforcement Learning Theory for Homeostatic Regulation". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 82–90. URL: http://papers.nips.cc/paper/4437-a-reinforcement-learning-theory-for-homeostatic-regulation.pdf.

[22] Clark Leonard Hull. "Principles of behavior: An introduction to behavior theory." In: (1943).

[23] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *ICML*. Vol. 99. 1999, pp. 278–287.

[24] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[25] Nicolò Cesa-Bianchi et al. "Boltzmann exploration done right". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6284–6293.

[26] Sebastian B. Thrun. *Efficient Exploration In Reinforcement Learning*. Tech. rep. Pittsburgh, PA, USA, 1992.

[27] Haoran Tang et al. "# Exploration: A study of count-based exploration for deep reinforcement learning". In: *Advances in neural information processing systems*. 2017, pp. 2753–2762.

[28] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2-3 (2002), pp. 235–256.

[29] Claes Strannegård et al. "Generic animats". In: *International Conference on Artificial General Intelligence*. Springer. 2017, pp. 23–32.

[30] Marlos C Machado et al. “Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents”. In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 523–562.

[31] Will Dabney et al. “Implicit quantile networks for distributional reinforcement learning”. In: *arXiv preprint arXiv:1806.06923* (2018).

[32] Thanh Thi Nguyen. “A multi-objective deep reinforcement learning framework”. In: *arXiv preprint arXiv:1803.02965* (2018).

[33] Open AI. *Open AI Gym*. Nov. 2018. URL: `https://gym.openai.com/`.

[34] Arryon D Tijsma, Madalina M Drugan, and Marco A Wiering. “Comparing exploration strategies for Q-learning in random stochastic mazes”. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2016, pp. 1–8.

A

Environments

A total of 13 distinct environments were constructed, built on the Open AI Gym framework[1], for the tests described in the main text. These environments are divided into three different complexity categories depending on the dimensionality of the observable state spaces.

First, the simple environments have no external state observable by the agent, although the environments themselves are not limited to a single state. Second, the Gridworld environments present the agents with their coordinates in the world. Last, the Expanding Color Gridworlds allow the agents to observe their immediate vicinity in the form of colour values.

In all worlds, a terminal state is considered reached when any of the internal states are outside the interval $(-1, 1)$.

A.1 Simple environments

The simple environments share the fact that there is no observable external state, only the internal variables are observed by the agent. For these environments, the internal needs of the agent are decreased by 0.004 each time step.

A.1.1 Greedy

The Greedy environment has 4 actions as described in Table A.1 and is fully deterministic and static.

[1] `https://gym.openai.com/`

Table A.1: The change in internal needs for the different actions in the greedy environment.

Action	Δh_1	Δh_2
a_0	0.04	0
a_1	0	0.02
a_2	-0.02	0
a_3	0	0

A.1.2 Random

The Random environment has 4 actions. Initially a h_1, h_2 increase and a h_2 decrease are each randomly associated with one action. The change in internal variables for each of these outcomes are 0.02, 0.04 and -0.02 respectively, similarly to the greed environment. Once an action with a non-zero outcome is selected, that outcome is randomly associated with a new action. Outcomes may randomly be associated with the same action. Likewise, two outcomes can be associated with the same actions.

A.1.3 Dual

The dual environment has 4 actions. The environment switches between acting like the Greed and Random environment on a fixed time basis. Starting out as a Greed environment, each 100 time steps the environment switches its type. When the environment switches to being greedy, outcome associations from the Random period are kept fixed rather than reset to that of the greedy world.

A.2 Gridworlds

The environments that fall into the Griworld category consist of two main types, the small cube shaped Gridworld and the Corridor Gridworld, each of which is described in their respective section.

A.2.1 Gridworld

The Gridworld environments are 3×3 worlds where the observation given to the agents consist of the agents (x, y) coordinate. Each time step, the internal variables of the agent are decreased by 0.0005. The agent has five actions, to move in the cardinal directions and to stand still. Standing still lowers the need decrease by a factor of 5. Unless stated otherwise, the top left corner tile gives $\Delta h_1 = 0.002$ and the lower right corner $\Delta h_2 = 0.002$. Note that traversing from one corner to the

other is exactly sustainable, however if any suboptimal action is selected the agent would have to stand still at a resource to remain sustainable.

The agent may try to move off the grid. Doing so will result in the position not changing, but the full decrease of needs being applied.

A.2.1.1 Static Safe

The static safe variant of the Gridworld is the most basic and easy version of the Gridworld as described above. No additional changes are made.

A.2.1.2 Static Hostile

The hostile modifier in the Gridworld makes it contain an extra tile in the top right corner with $\Delta h_2 = -0.02$.

A.2.1.3 Dynamic Safe

The dynamic modifier in the Gridworld makes it non-stationary. Now, each time the agent enters the h_1 tile, its abundance decreases by 1%. This abundance parameter is applied to calculate the total Δh_1 gained such that consequent visits will give less. When the abundance reaches zero, the tile is randomly placed in any of the 9 tiles, possibly overlapping with the h_2 tile. The initial h_1 value is increased by a factor of 1.5 in order to keep the total amount of h_1 gained over time constant as compared to the non-dynamic versions.

A.2.1.4 Dynamic Hostile

The dynamic and hostile Gridworld is a combination of the dynamic and hostile modifier. The dynamic tile is not constrained and may overlap the hostile tile.

A.2.2 Corridor

The Corridor environments are similar to the previously mentioned Gridworlds in its underlying structure. There are two instances of this environment type, the difference being the length, either 10 or 20 tiles which are laid out lengthwise. In each end of the resulting corridor one of the resources are found, both resulting in a resource gain of 0.1. The resource decay for the short corridor is 0.02 and for the long corridor 0.01. The up and down action is not included.

A.3 Expanding Color Gridworld

The final category of environments are the Expanding Color Gridworlds. In these environments the external observable space is the RGB colour of the 9 tiles surrounding the agent, a total of 27 floats values in the range of (0,1). All ECG environments start out with a single h_2 resource in the top right and a h_1 source in the lower left corner of a 5×5 grid. The size of the grid increase every 10,000 steps with one tile in each cardinal direction.

The resource decay in all of the environments is 0.0005, and the gain from collecting any resource is 0.2.

A total of 4 different ECG environments were used, with the settings Static/Dynamic and Safe/Hostile.

A.3.1 Static Safe

The static safe environment only expanded in size, but did not provide any additional resources or dangers.

A.3.2 Static Hostile

The static hostile environment had an additional danger tile which reduced h_2 with 0.05.

A.3.2.1 Dynamic Safe

The dynamic safe environment created additional h_2 sources and trees as the environment expanded. In this environment the h_1 source would disappear upon consumption and respawn near one of the trees.

A.3.2.2 Dynamic Hostile

The dynamic hostile environment was similar to the dynamic safe environment, however this environment also spawned additional danger tiles with $\Delta h_2 = 0.05$.

B

Results

This appendix contains the full results of the evaluations.

Table B.1: The average number of deaths over 10 simulations.

Strategy	Corridor long	Corridor short	Dual	ECG DH	ECG DS	ECG SH	ECG SS	Greed	Gridworld DH	Gridworld DS	Gridworld SH	Gridworld SS	Random
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=0.0001)	696.00	1143.40	3122.30	244.30	720.40	435.10	588.40	440.70	257.40	240.90	96.30	18.80	3592.60
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=0.001)	767.20	1071.00	3038.50	299.20	809.00	310.10	657.00	707.90	268.80	222.60	52.40	13.30	3531.00
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=1e-05)	1089.50	1392.30	2986.60	287.70	556.90	395.30	677.40	275.00	299.70	253.50	201.50	105.90	3422.70
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=0.0001)	703.60	501.90	2359.20	303.40	877.00	643.00	1356.20	119.10	242.90	161.50	150.20	5.90	3131.40
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=0.001)	751.30	481.50	2315.20	356.00	894.30	822.30	1330.30	1134.30	219.40	173.30	55.50	2.40	3120.90
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=1e-05)	1135.20	849.30	2281.20	395.90	1047.40	762.90	1603.70	752.10	245.70	184.80	222.30	84.40	3020.10
AnnealedSoftmax (τ=0.001,τ_{Δ}=1e-05,$\tau_{\min}$=0.0001)	823.00	1161.10	2765.50	1031.00	1564.20	601.90	1044.10	574.70	219.80	196.70	41.60	9.20	3421.40
AnnealedSoftmax (τ=0.01,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	588.50	516.00	1761.00	292.40	1229.40	950.40	392.00	646.90	274.00	234.10	91.20	62.20	2558.00
AnnealedSoftmax (τ=0.1,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	541.30	461.50	1726.00	977.40	455.10	300.60	556.90	479.50	266.30	239.10	165.90	70.50	2605.80
AnnealedSoftmax (τ=1.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	577.80	738.90	2138.10	241.20	380.80	298.70	752.80	666.00	288.60	244.90	289.00	171.70	2900.90
AnnealedSoftmax (τ=10.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	1492.90	2467.00	1953.90	517.00	726.30	755.10	904.40	1520.30	299.60	243.80	302.00	182.60	2818.50
AnnealedSoftmax (τ=100.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	7787.80	14753.10	1276.90	4829.00	5428.90	7348.90	7977.00	1371.30	418.40	247.40	419.20	211.10	1270.50
ConstantEpsilon (ϵ=0.0)	843.70	1394.00	2733.10	934.80	639.60	1506.80	437.10	744.40	325.50	263.10	45.70	33.00	3438.80
ConstantEpsilon (ϵ=0.01)	781.40	879.70	2982.50	739.10	1249.50	1085.90	424.60	1122.20	263.70	227.30	20.90	2.50	3525.10
ConstantEpsilon (ϵ=0.1)	725.10	453.50	2540.10	354.80	897.30	661.00	1356.50	1038.50	237.70	161.40	0.30	2.10	3091.70
ConstantEpsilon (ϵ=0.2)	839.20	284.20	1945.90	584.10	1600.10	1055.70	2264.20	594.80	226.40	155.10	42.40	17.60	2504.50
ConstantEpsilon (ϵ=0.3)	1135.80	345.10	1261.00	924.60	2303.60	1551.00	3348.80	103.30	253.20	137.90	85.10	9.90	1969.20
ConstantEpsilon (ϵ=0.4)	1359.40	602.70	998.90	1244.90	2911.60	1921.40	4103.70	244.70	279.30	181.20	158.10	20.50	1341.50
EWB ϵ (θ=0)	3460.80	4387.20	658.50	1046.20	2049.70	1782.10	2527.20	291.30	318.70	198.10	294.70	91.50	275.00
EWB ϵ (θ=0.2)	2282.10	2616.60	646.30	479.60	669.50	680.10	744.90	216.80	288.60	205.20	181.90	52.90	361.40
EWB ϵ (θ=0.4)	1700.10	2363.30	813.20	316.70	433.60	411.70	574.90	66.70	318.20	226.60	51.30	8.30	504.90
EWB ϵ (θ=0.6)	1300.50	2181.00	1161.30	260.20	299.00	602.00	648.80	468.90	319.60	219.30	28.90	0.70	958.60
EWB ϵ (θ=0.8)	912.30	1816.70	2104.80	378.20	431.20	534.80	800.10	373.30	309.30	243.30	10.40	20.20	2319.50
EWBSoftmax (τ_h=0.001)	716.30	797.70	2598.70	407.30	1003.60	522.20	406.90	571.00	252.10	209.20	111.10	76.10	3164.30
EWBSoftmax (τ_h=0.01)	300.20	169.70	810.50	223.30	525.10	281.30	554.40	6.90	347.70	239.90	357.50	193.10	351.00
EWBSoftmax (τ_h=0.1)	5819.00	3391.30	1214.10	209.50	496.60	170.20	769.30	81.70	405.80	246.60	408.70	208.70	1163.50
EWBSoftmax (τ_h=1.0)	7681.20	14001.60	1272.20	3038.90	2707.30	4047.60	3342.80	1085.40	416.50	246.90	416.30	211.10	1277.20
EWBSoftmax (τ_h=10.0)	7782.50	14719.80	1276.80	4607.00	4687.40	6733.00	6698.90	1345.00	416.80	247.40	416.90	211.70	1271.80
EWBSoftmax (τ_h=100.0)	7791.20	14805.30	1286.90	4896.40	5367.60	7310.00	7903.40	1368.80	417.80	246.80	418.70	211.90	1281.00
EWG ϵ (θ=0)	777.20	220.00	769.90	1092.00	2204.10	1542.10	3091.60	190.20	229.70	178.30	83.50	16.20	352.00
EWG ϵ (θ=0.2)	723.00	222.20	852.80	287.20	560.90	327.10	780.50	29.10	234.20	189.20	53.10	15.00	493.80
EWG ϵ (θ=0.4)	1201.70	241.30	1084.20	111.80	490.60	175.50	411.70	27.80	264.30	214.00	74.80	8.50	809.90
EWG ϵ (θ=0.6)	1027.00	1308.90	1461.90	935.20	301.40	170.20	424.00	86.60	283.30	224.90	26.30	0.00	1429.30
EWG ϵ (θ=0.8)	970.70	1396.10	2279.70	177.70	991.10	395.40	1356.20	299.40	287.00	237.70	29.30	1.00	2481.70
EWGSoftmax (τ_h=0.001)	801.80	878.40	2361.10	465.60	456.00	651.50	770.70	210.20	245.60	183.30	57.60	19.90	3104.80
EWGSoftmax (τ_h=0.01)	226.40	125.70	785.10	191.10	325.10	352.40	702.50	1.60	288.40	248.90	234.30	139.70	249.70
EWGSoftmax (τ_h=0.1)	2304.10	607.30	1185.20	115.80	317.30	156.40	1859.70	0.10	330.50	249.70	316.00	216.00	1165.20
EWGSoftmax (τ_h=1.0)	7354.90	11346.40	1266.90	1895.10	2497.80	2232.40	2776.70	743.00	420.50	246.80	421.50	211.00	1271.10
EWGSoftmax (τ_h=10.0)	7706.60	14403.00	1276.50	4241.60	4726.90	5918.10	6343.50	1313.80	419.60	247.20	419.00	211.20	1269.80
EWGSoftmax (τ_h=100.0)	7771.20	14721.30	1282.40	4813.10	5394.50	7238.20	7866.00	1365.10	418.00	247.60	419.40	211.40	1277.10
Random	7794.60	14778.30	1281.60	4893.30	5487.40	7431.50	8080.70	1382.80	417.60	247.30	418.40	211.30	1274.50
Softmax (τ=0.001)	572.50	535.50	1899.40	489.10	927.10	505.00	457.30	316.30	274.00	233.90	70.80	65.70	2746.40
Softmax (τ=0.01)	411.30	129.10	1139.20	167.90	723.30	122.00	570.70	39.50	396.80	244.80	412.30	211.50	1050.20
Softmax (τ=0.1)	7121.30	8025.30	1277.30	739.50	1261.10	719.70	1453.90	549.50	418.00	247.50	418.60	210.50	1270.90
Softmax (τ=1.0)	7736.10	14420.10	1281.50	4148.60	4655.30	5971.80	6655.80	1264.30	418.00	246.80	418.90	211.60	1274.90
Softmax (τ=10.0)	7790.40	14777.00	1277.00	4830.60	5382.50	7303.90	7905.50	1360.60	417.60	246.90	418.40	211.10	1276.70
Softmax (τ=100.0)	7791.20	14799.90	1285.10	4901.00	5460.00	7439.80	8071.00	1374.10	417.20	247.10	418.50	210.90	1276.30

Table B.2: The normalised score of the average number of deaths.

Strategy	Corridor long	Corridor short	Dual	ECG DH	ECG DS	ECG SH	ECG SS	Greed	Gridworld DH	Gridworld DS	Gridworld SH	Gridworld SS	Random	Average normalized score	Rank
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=0.0001)	0.91	0.92	-1.44	0.95	0.87	0.94	0.93	0.68	0.38	0.03	0.77	0.91	-1.82	0.39	30
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=0.001)	0.90	0.93	-1.37	0.94	0.85	0.96	0.92	0.49	0.36	0.10	0.87	0.94	-1.77	0.39	29
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=1e-05)	0.86	0.91	-1.33	0.94	0.90	0.95	0.92	0.80	0.28	-0.03	0.52	0.50	-1.69	0.35	34
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=0.0001)	0.91	0.97	-0.84	0.94	0.84	0.91	0.83	0.91	0.42	0.35	0.64	0.97	-1.46	0.49	22
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=0.001)	0.90	0.97	-0.81	0.93	0.84	0.89	0.84	0.18	0.47	0.30	0.87	0.99	-1.45	0.45	25
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=1e-05)	0.85	0.94	-0.78	0.92	0.81	0.90	0.80	0.46	0.41	0.25	0.47	0.60	-1.37	0.40	28
AnnealedSoftmax (τ=0.001,τ_{Δ}=1e-05,$\tau_{\min}$=0.0001)	0.89	0.92	-1.16	0.79	0.71	0.92	0.87	0.58	0.47	0.20	0.90	0.96	-1.68	0.41	27
AnnealedSoftmax (τ=0.01,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.92	0.97	-0.37	0.94	0.78	0.87	0.95	0.53	0.34	0.05	0.78	0.71	-1.01	0.50	20
AnnealedSoftmax (τ=0.1,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.93	0.97	-0.35	0.80	0.92	0.96	0.93	0.65	0.36	0.03	0.60	0.67	-1.04	0.49	21
AnnealedSoftmax (τ=1.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.93	0.95	-0.67	0.95	0.93	0.96	0.91	0.52	0.31	0.01	0.31	0.19	-1.28	0.39	31
AnnealedSoftmax (τ=10.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.81	0.83	-0.52	0.89	0.87	0.90	0.89	-0.10	0.28	0.01	0.28	0.14	-1.21	0.31	36
AnnealedSoftmax (τ=100.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	-0.00	-0.00	-0.00	0.00	0.00	0.00	44
ConstantEpsilon (ϵ=0.0)	0.89	0.91	-1.13	0.81	0.88	0.80	0.95	0.46	0.22	-0.06	0.89	0.84	-1.70	0.37	32
ConstantEpsilon (ϵ=0.01)	0.90	0.94	-1.33	0.85	0.77	0.85	0.95	0.19	0.37	0.08	0.95	0.99	-1.77	0.37	33
ConstantEpsilon (ϵ=0.1)	0.91	0.97	-0.98	0.93	0.84	0.91	0.83	0.25	0.43	0.35	1.00	0.99	-1.43	0.46	23
ConstantEpsilon (ϵ=0.2)	0.89	0.98	-0.52	0.88	0.71	0.86	0.72	0.57	0.46	0.37	0.90	0.92	-0.97	0.52	16
ConstantEpsilon (ϵ=0.3)	0.85	0.98	0.02	0.81	0.58	0.79	0.59	0.93	0.39	0.44	0.80	0.95	-0.55	0.58	10
ConstantEpsilon (ϵ=0.4)	0.83	0.96	0.22	0.75	0.47	0.74	0.49	0.82	0.33	0.27	0.62	0.90	-0.05	0.57	12
EWB ϵ (θ=0)	0.56	0.70	0.49	0.79	0.63	0.76	0.69	0.79	0.24	0.20	0.30	0.57	0.78	0.58	11
EWB ϵ (θ=0.2)	0.71	0.82	0.50	0.90	0.88	0.91	0.91	0.84	0.31	0.17	0.57	0.75	0.72	0.69	6
EWB ϵ (θ=0.4)	0.78	0.84	0.37	0.94	0.92	0.94	0.93	0.95	0.24	0.08	0.88	0.96	0.60	0.73	2
EWB ϵ (θ=0.6)	0.83	0.85	0.09	0.95	0.95	0.92	0.92	0.66	0.23	0.11	0.93	1.00	0.25	0.67	7
EWB ϵ (θ=0.8)	0.88	0.88	-0.64	0.92	0.92	0.93	0.90	0.73	0.26	0.02	0.98	0.90	-0.82	0.53	15
EWBSoftmax (τ_h=0.001)	0.91	0.95	-1.03	0.92	0.82	0.93	0.95	0.59	0.40	0.15	0.73	0.64	-1.48	0.42	26
EWBSoftmax (τ_h=0.01)	0.96	0.99	0.37	0.95	0.90	0.96	0.93	1.00	0.17	0.03	0.15	0.09	0.72	0.63	9
EWBSoftmax (τ_h=0.1)	0.25	0.77	0.05	0.96	0.91	0.98	0.90	0.94	0.03	0.00	0.02	0.01	0.09	0.46	24
EWBSoftmax (τ_h=1.0)	0.01	0.05	0.01	0.38	0.51	0.46	0.59	0.22	0.00	0.00	0.01	0.00	-0.00	0.17	38
EWBSoftmax (τ_h=10.0)	0.00	0.00	0.00	0.06	0.15	0.09	0.17	0.03	0.00	-0.00	0.00	-0.00	0.00	0.04	41
EWBSoftmax (τ_h=100.0)	0.00	-0.00	-0.00	-0.00	0.02	0.02	0.02	0.01	-0.00	0.00	-0.00	-0.00	-0.01	0.00	45
EWG ϵ (θ=0)	0.90	0.99	0.40	0.78	0.60	0.79	0.62	0.86	0.45	0.28	0.80	0.92	0.72	0.70	4
EWG ϵ (θ=0.2)	0.91	0.98	0.33	0.94	0.90	0.96	0.90	0.98	0.44	0.23	0.87	0.93	0.61	0.77	1
EWG ϵ (θ=0.4)	0.85	0.98	0.15	0.98	0.91	0.98	0.95	0.98	0.37	0.13	0.82	0.96	0.36	0.72	3
EWG ϵ (θ=0.6)	0.87	0.91	-0.14	0.81	0.95	0.98	0.95	0.94	0.32	0.09	0.94	1.00	-0.12	0.65	8
EWG ϵ (θ=0.8)	0.88	0.91	-0.78	0.96	0.82	0.95	0.83	0.78	0.31	0.04	0.93	1.00	-0.95	0.51	17
EWGSoftmax (τ_h=0.001)	0.90	0.94	-0.84	0.90	0.92	0.91	0.90	0.85	0.41	0.26	0.86	0.91	-1.44	0.50	19
EWGSoftmax (τ_h=0.01)	0.97	0.99	0.39	0.96	0.94	0.95	0.91	1.00	0.31	-0.01	0.44	0.34	0.80	0.69	5
EWGSoftmax (τ_h=0.1)	0.70	0.96	0.08	0.98	0.94	0.98	0.77	1.00	0.21	-0.01	0.24	-0.02	0.09	0.53	14
EWGSoftmax (τ_h=1.0)	0.06	0.23	0.01	0.61	0.54	0.70	0.66	0.46	-0.01	0.00	-0.01	0.00	0.00	0.25	37
EWGSoftmax (τ_h=10.0)	0.01	0.03	0.00	0.13	0.14	0.20	0.21	0.05	-0.00	0.00	-0.00	0.00	0.00	0.06	40
EWGSoftmax (τ_h=100.0)	0.00	0.00	-0.00	0.02	0.02	0.03	0.03	0.01	-0.00	-0.00	-0.00	-0.00	-0.00	0.01	42
Random	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	0.00	47
Softmax (τ=0.001)	0.93	0.96	-0.48	0.90	0.83	0.93	0.94	0.77	0.34	0.05	0.83	0.69	-1.15	0.50	18
Softmax (τ=0.01)	0.95	0.99	0.11	0.97	0.87	0.98	0.93	0.97	0.05	0.01	0.01	-0.00	0.18	0.54	13
Softmax (τ=0.1)	0.09	0.46	0.00	0.85	0.77	0.90	0.82	0.60	-0.00	-0.00	-0.00	0.00	0.00	0.35	35
Softmax (τ=1.0)	0.01	0.02	0.00	0.15	0.15	0.20	0.18	0.09	-0.00	0.00	-0.00	-0.00	-0.00	0.06	39
Softmax (τ=10.0)	0.00	0.00	0.00	0.01	0.02	0.02	0.02	0.02	-0.00	0.00	-0.00	0.00	-0.00	0.01	43
Softmax (τ=100.0)	0.00	-0.00	-0.00	-0.00	0.00	-0.00	0.00	0.01	0.00	0.00	-0.00	0.00	-0.00	0.00	46

Table B.3: The average fraction of exploratory actions taken over 10 simulations.

Strategy	Corridor long	Corridor short	Dual	ECG DH	ECG DS	ECG SH	ECG SS	Greed	Gridworld DH	Gridworld DS	Gridworld SH	Gridworld SS	Random
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=0.0001)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=0.001)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.01,ϵ_{Δ}=1e-05)	0.04	0.04	0.04	0.05	0.05	0.05	0.05	0.04	0.05	0.05	0.05	0.05	0.04
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=0.0001)	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=0.001)	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
AnnealedEpsilon (ϵ=1.0,$\epsilon_{\min}$=0.1,ϵ_{Δ}=1e-05)	0.09	0.09	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
AnnealedSoftmax (τ=0.001,τ_{Δ}=1e-05,$\tau_{\min}$=0.0001)	0.01	0.01	0.03	0.05	0.06	0.05	0.06	0.10	0.15	0.18	0.24	0.28	0.03
AnnealedSoftmax (τ=0.01,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.06	0.04	0.26	0.33	0.35	0.30	0.40	0.44	0.53	0.57	0.63	0.66	0.27
AnnealedSoftmax (τ=0.1,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.06	0.04	0.26	0.31	0.40	0.33	0.39	0.46	0.55	0.55	0.60	0.66	0.26
AnnealedSoftmax (τ=1.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.07	0.04	0.25	0.32	0.43	0.35	0.37	0.39	0.52	0.54	0.53	0.59	0.24
AnnealedSoftmax (τ=10.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.12	0.09	0.29	0.34	0.42	0.34	0.41	0.33	0.54	0.57	0.54	0.60	0.27
AnnealedSoftmax (τ=100.0,τ_{Δ}=0.0001,$\tau_{\min}$=0.001)	0.67	0.67	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75
ConstantEpsilon (ϵ=0.0)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ConstantEpsilon (ϵ=0.01)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
ConstantEpsilon (ϵ=0.1)	0.07	0.07	0.07	0.08	0.08	0.08	0.08	0.07	0.08	0.08	0.08	0.08	0.07
ConstantEpsilon (ϵ=0.2)	0.13	0.13	0.15	0.16	0.16	0.16	0.16	0.15	0.16	0.16	0.16	0.16	0.15
ConstantEpsilon (ϵ=0.3)	0.20	0.20	0.23	0.24	0.24	0.24	0.24	0.23	0.24	0.24	0.24	0.24	0.22
ConstantEpsilon (ϵ=0.4)	0.27	0.27	0.30	0.32	0.32	0.32	0.32	0.30	0.32	0.32	0.32	0.32	0.30
EWB ϵ (θ=0)	0.26	0.24	0.38	0.18	0.18	0.16	0.16	0.12	0.38	0.37	0.35	0.24	0.45
EWB ϵ (θ=0.2)	0.15	0.12	0.33	0.05	0.05	0.04	0.04	0.06	0.30	0.30	0.18	0.10	0.44
EWB ϵ (θ=0.4)	0.08	0.06	0.29	0.02	0.02	0.02	0.02	0.02	0.23	0.24	0.04	0.01	0.42
EWB ϵ (θ=0.6)	0.03	0.03	0.23	0.01	0.01	0.02	0.02	0.06	0.16	0.16	0.02	0.00	0.36
EWB ϵ (θ=0.8)	0.01	0.01	0.12	0.01	0.01	0.01	0.01	0.02	0.08	0.09	0.00	0.01	0.19
EWBSoftmax (τ_h=0.001)	0.02	0.01	0.12	0.12	0.14	0.11	0.14	0.04	0.38	0.39	0.23	0.32	0.13
EWBSoftmax (τ_h=0.01)	0.17	0.08	0.55	0.44	0.50	0.42	0.46	0.18	0.64	0.67	0.64	0.69	0.58
EWBSoftmax (τ_h=0.1)	0.55	0.48	0.72	0.71	0.71	0.70	0.69	0.62	0.77	0.78	0.77	0.78	0.72
EWBSoftmax (τ_h=1.0)	0.65	0.64	0.75	0.78	0.77	0.77	0.77	0.74	0.80	0.80	0.80	0.80	0.75
EWBSoftmax (τ_h=10.0)	0.67	0.66	0.75	0.80	0.80	0.79	0.79	0.75	0.80	0.80	0.80	0.80	0.75
EWBSoftmax (τ_h=100.0)	0.67	0.67	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75
EWG ϵ (θ=0)	0.39	0.40	0.42	0.37	0.32	0.42	0.32	0.60	0.36	0.38	0.49	0.51	0.45
EWG ϵ (θ=0.2)	0.33	0.37	0.38	0.16	0.11	0.15	0.09	0.60	0.29	0.32	0.43	0.49	0.43
EWG ϵ (θ=0.4)	0.22	0.32	0.32	0.14	0.08	0.13	0.07	0.59	0.23	0.26	0.37	0.47	0.39
EWG ϵ (θ=0.6)	0.12	0.19	0.25	0.14	0.06	0.12	0.06	0.57	0.16	0.19	0.36	0.50	0.32
EWG ϵ (θ=0.8)	0.04	0.05	0.12	0.07	0.07	0.05	0.07	0.47	0.08	0.10	0.30	0.41	0.16
EWGSoftmax (τ_h=0.001)	0.04	0.03	0.14	0.25	0.31	0.28	0.28	0.45	0.40	0.44	0.59	0.63	0.16
EWGSoftmax (τ_h=0.01)	0.36	0.31	0.57	0.67	0.68	0.66	0.66	0.68	0.66	0.69	0.67	0.70	0.58
EWGSoftmax (τ_h=0.1)	0.55	0.52	0.72	0.77	0.77	0.77	0.71	0.71	0.76	0.77	0.76	0.78	0.72
EWGSoftmax (τ_h=1.0)	0.63	0.61	0.75	0.79	0.79	0.78	0.79	0.73	0.79	0.79	0.79	0.80	0.75
EWGSoftmax (τ_h=10.0)	0.66	0.65	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75
EWGSoftmax (τ_h=100.0)	0.66	0.66	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75
Random	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Softmax (τ=0.001)	0.06	0.04	0.24	0.34	0.37	0.33	0.37	0.47	0.54	0.57	0.65	0.66	0.25
Softmax (τ=0.01)	0.43	0.36	0.68	0.71	0.74	0.72	0.73	0.68	0.75	0.76	0.75	0.76	0.68
Softmax (τ=0.1)	0.63	0.59	0.74	0.78	0.79	0.78	0.79	0.73	0.80	0.80	0.79	0.80	0.74
Softmax (τ=1.0)	0.66	0.66	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75
Softmax (τ=10.0)	0.67	0.67	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75
Softmax (τ=100.0)	0.67	0.67	0.75	0.80	0.80	0.80	0.80	0.75	0.80	0.80	0.80	0.80	0.75

C

Agents

Depending on the environment, two different sizes of the hidden layers were used. The small environments include the Simple environments and the 3×3 gridworlds. The large environments include the corridor environments and the Expanding Color Gridworlds. All other hyperparameters were kept constant between simulations, in order to compare the effects of the strategies alone. A full list of the hyperparameters of the agent can be found in C.1.

Table C.1: A large and a small DQN were used for different environments, the hyperparameters of which were kept identical the same in all evaluations.

	Small	Large
Hidden layers	[4,4]	[64,64,64]
Optimizer	Adam	Adam
Learning rate	1e-3	1e-3
γ	0.9	0.9
Batch size	32	32
Switch after steps	200	200
Train after steps	500	500
Drive n	4	4
Drive m	3	3