Navigating Errors: The Tolerance of Reinforcement Learning Algorithms to Misleading Heuristics

Andy Edmondson[1, 2], **Ronald P. A. Petrick**[1]

[1]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
[2]School of Informatics, The University of Edinburgh, Edinburgh, UK
Andy.Edmondson@ed.ac.uk, R.Petrick@hw.ac.uk

Abstract

Applying guidance to *Reinforcement Learning (RL)* algorithms to improve learning sample efficiencies using techniques such as *reward-shaping* and *action heuristics* have been explored since the inception of the field. With the advent of *Large Language Models (LLMs)*, a new and potentially powerful source of heuristics with broad contextual understanding is being explored. However, LLMs are prone to hallucination and are likely to provide inconsistent heuristic values. This paper explores how tolerant *Tabular Q-Learning*, *Deep Q Networks (DQN)*, *Proximal Policy Optimisation (PPO)* and *Soft Actor-Critic (SAC)* are to misleading heuristics and how this affects training. The contribution of this study is to show the extent to which RL algorithms can learn good policies despite misleading guidance. These findings will assist researchers to understand the effects of misleading heuristics on RL algorithms' ability to reliably learn good policies, as well as the effect of heuristics on final policies, informing use of error-prone heuristics such as LLMs.

Introduction

Recent advances in *Deep Reinforcement Learning (DRL)* have seen automated agents beat human world champions at Chess and Go (Schrittwieser et al. 2020), Diplomacy (Meta Fundamental AI Research Diplomacy Team (FAIR)† et al. 2022), Poker (Brown et al. 2020) and even drone racing (Hanover et al. 2023). However, the challenges of *credit assignment* and *exploration* (Sutton and Barto 2018) persist, with agents struggling to associate specific actions with delayed rewards and failing to sufficiently explore the action space, a problem made more acute in environments with sparse rewards.

Researchers have tried various methods of enriching the reward landscape to address these issues. For instance, the *Intrinsic Curiosity Module (ICM)* (Pathak et al. 2017) and *Hindsight Experience Replay (HER)* (Andrychowicz et al. 2017) add rewards based on past experiences, encouraging the agent to explore further afield. *Hierarchical Reinforcement Learning (HRL)* methods such as *HIRO* (Nachum et al. 2018) and the *Options Framework* (Sutton, Precup, and Singh 1999) learn subgoals, dividing tasks into chunks with their own additional rewards. Alternatively, heuristics can provide domain specific knowledge about the environment and available actions through hand-crafted functions (Ng, Harada, and Russell 1999; Sutton and Barto 2018). More recently, researchers have used *Large Language Models (LLMs)* to dynamically provide context-aware rewards during training (Hu and Sadigh 2023). Such heuristics can be instrumental in improving generalisation and learning sample efficiencies by conferring domain-specific knowledge on agents and enabling them to make better decisions earlier in training.

Figure 1: Maps from Point Maze (left) and Frozen Lake (right). Red diagonal arrows denote the natural route, green arrows along the edges denote the heuristic-directed route.

Heuristics such as *Manhattan Distance* are, by their nature, useful but incomplete proxies for desired action sequences and may be noisy or even misleading. In particular, estimation by LLMs is subject to their training data, the trained task, and to hallucinations (Qiao et al. 2024; Huang et al. 2024), again leading to questions of heuristic quality. Further, domain-specific knowledge provided to agents, or guidance given during human-agent teaming, may not translate to analytically optimal solutions with the best solution instead being one which is interpretable. Examining the subject in the context of RL, this work investigates the effect of misleading heuristics on the training of common algorithms in a simple navigation task, answering the questions:

1. To what extent can common RL algorithms tolerate misleading heuristics?
2. How do misleading heuristics affect learning?

To ensure heuristic error remains the key factor in train-

ing, the well known and simple Frozen Lake[1] (Towers et al. 2024) and Point Maze[2] (Fu et al. 2020) environments are selected. The *Tabular Q-Learning* (Watkins 1989) and *Deep Q Network (DQN)* (Mnih et al. 2015) algorithms are used within the discrete Frozen Lake environment and provide a useful baseline for comparison. The *Soft Actor Critic (SAC)* (Haarnoja et al. 2018) and *Proximal Policy Optimization (PPO)* (Schulman et al. 2017) algorithms are used with the continuous Point Maze.

Heuristics are applied using *Potential-Based Reward Shaping* (Ng, Harada, and Russell 1999), a common and well understood method within RL. Heuristic errors are introduced as misleading rewards, making actions in a given state appear good or bad when reality is the opposite. As shown in Figure 1, the heuristic asks the agent to learn a longer "inefficient" route around the environment's edge to prove the agent is learning from the heuristic and not from unguided exploration which prefers the shorter red route. It is important to note that the misleading heuristic does not simply consist of random noise applied to reward values, rather it consists of values designed to make desirable actions appear undesirable and vice versa.

While the navigation tasks in this paper are simple, the purpose of this work is to understand the algorithms' underlying capacity to learn despite directly misleading heuristics rather than prove anything about their capabilities in solving some arbitrary class of tasks. Also, by forcing the agent to learn an "inefficient" route, it will be clear whether an agent is still following the heuristic to the goal, or has become confused by the misleading heuristic and, nonetheless, found the goal through standard RL exploration mechanisms.

The following sections discuss background and related work, describe the heuristics in detail, and show how they are used with each RL algorithm. Results are then discussed, showing that RL algorithms are generally robust to quite high rates of misleading heuristics with some cost to training stability and sample efficiencies. It is also shown that the effect of heuristics on policy is not uniform across different algorithm types. As such, it is possible, with appropriate algorithm selection and implementation, to use heuristics and reward shaping to transfer domain knowledge to general RL algorithms, directing policy learning in a desired direction and improving sample efficiencies.

Background and Related Work

Reinforcement Learning

Reinforcement Learning (RL) algorithms are those which learn a mapping from situations to actions through experience. This mapping can be achieved in various ways, but they all seek to learn a sequence of actions which achieve a defined goal through *reward maximisation* (Sutton and Barto 2018). Formally, this type of learning can be described as a *Markov Decision Process (MDP)* with the tuple $M = (S, A, P, R, \gamma)$, where S is the set of states, A the set of available actions, P the transition function $P(s'|s, a)$, R the reward function $R(s, a, s')$ and $\gamma \in [0, 1]$ is a discount factor (Sutton and Barto 2018). That is, from a state $s \in S$, undertaking an action $a \in A$ has a probability $p \in [0, 1]$ of transitioning to state $s' \in S$ and receiving a reward $r \in R$ with $\gamma \in [0, 1]$. RL therefore learns an optimal policy of sequential actions (Sutton and Barto 2018) where

$$q_*(s, a) \doteq \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a],$$

which maximises the *discounted sum of rewards*:

$$G_t \doteq \sum_{k=0}^{\inf} \gamma^k R_{t+k+1}.$$

Several approaches to RL have been tried for various contexts. This paper's focus is online model-free algorithms operating in discrete and continual action spaces. Tabular Q-Learning (Watkins 1989) is used as a baseline since convergence properties are well studied and understood (Sutton and Barto 2018; Albrecht, Christianos, and Schäfer 2024). Policies are learned through repeated application of the Bellman update (Sutton and Barto 2018):

$$Q(S, A) \longleftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)],$$

where S is the current state, S' the state at the next timestep, A the action taken in S, and a an available action in S'. There are different exploration schemes, but in this work actions are selected using $\epsilon-greedy$, with actions taken greedily from the policy or chosen randomly with probability ϵ.

DQN (Mnih et al. 2015) is the first *Deep Reinforcement Learning (DRL)* model examined, providing an initial comparison with the tabular version before moving to the continuous environment. Since DQN uses a replay buffer to store experience tuples $e = (s, a, r, s')$ and DNNs to approximate target and current action values, updates use the loss function (Mnih et al. 2015):

$$L_i(\theta_i) = \mathbb{E}_{e\sim U(D)}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)^2]$$

Integrating dual Q-networks with an additional actor network to cope with continuous environments, *Twin Delayed DDPG (TD3)* (Fujimoto, van Hoof, and Meger 2018) provides a deterministic policy actor-critic method for continuous action spaces, bringing with it many of the advantages of DQL (Mnih et al. 2015), while solving some of the shortcomings of *Deep Deterministic Policy Gradient (DDPG* (Lillicrap et al. 2016) such as overestimation bias.

Taking a different approach to the off-policy actor-critic architecture, *Soft Actor Critic (SAC)* (Haarnoja et al. 2018) uses a stochastic policy and an entropy term in the loss function, aiming to maximise random behaviour while still achieving the desired goal. Here, Q-network optimisation is

$$J_Q(\theta) = \mathbb{E}_{(s,a)\sim D}$$
$$= [\frac{1}{2}(Q_\theta(s, a) - (r(s, a) + \gamma \mathbb{E}_{s'\sim p}[V_{\bar{\theta}}(s')]))^2],$$
where $V(s) = \mathbb{E}_{a\sim\pi}[Q(s, a) - \alpha \log \pi(a|s)]$,

and the policy is optimised with

$$J_\pi(\phi) = \mathbb{E}_{s\sim D}[\mathbb{E}_{a\sim\pi_\phi}[\alpha \log(\pi_\phi(a|s)) - Q_\theta(s, a)]]$$

[1] https://gymnasium.farama.org/environments/toy_text/frozen_lake/

[2] https://robotics.farama.org/envs/maze/point_maze/

Alternatively, the popular *Proximal Policy Optimization (PPO)* (Schulman et al. 2017) is an evolution of *Policy Gradient* algorithms and is designed to minimise variance in updates through *trust regions*. Requiring only two networks, compared to the six of TD3 and SAC, it is additionally the more computationally efficient option.

The following section discusses heuristics in general, and how this paper applies heuristics to the algorithms used.

Heuristics for Reinforcement Learning

Heuristics are applied to RL for a myriad of reasons including exploration (Pathak et al. 2017; Andrychowicz et al. 2017), improved sample efficiency (Schaul et al. 2016), and policy understandability for agent-human cooperation (Hu and Sadigh 2023). Most approaches involve reward shaping (Ng, Harada, and Russell 1999), seeking to enrich the reward surface and guide the policy indirectly (Pathak et al. 2017; Radford et al. 2021; Kwon et al. 2023). Alternatively, Hu and Sadigh (2023) apply the heuristic directly to the value function of Tabular Q-Learning and DQN, and modify the discrete PPO objective with a *Kullback-Leibler (KL)* penalty.

Applying a heuristic directly to the objective function works well in discrete action spaces, but is not easily applied in continuous spaces where smooth mechanical actions are required. If the heuristic function is 'expert' enough to provide a suitably smooth series of actions, the learning problem changes to one of *Imitation Learning (IL)* (Zare et al. 2024) instead of the standard RL problem whereby high-value actions are not discoverable before the agent acts in the environment (Sutton and Barto 2018). Rather, through reward shaping, the agent is given some level of domain-specific knowledge with no prior judgment made about the specific actions required to maximise the enriched rewards.

Potential-Based Reward Shaping (Ng, Harada, and Russell 1999), adjusts the reward with

$$r' = r + \gamma\Phi(s') - \Phi(s), \tag{1}$$

where $\Phi(s)$ returns the heuristic value for $s, s' \in S$ and $\gamma \in [0, 1]$ is a discount factor. This enriches the reward landscape and allows the agent to discover an optimal policy *within the region of the environment explored by the agent* without biasing the optimal policy (Ng, Harada, and Russell 1999). However, applying a heuristic as a reward must also be done in a numerically stable manner, especially when many heuristics (such as *Manhattan Distance*) reduce as the agent nears the goal. Gehring et al. (2022) propose

$$h_\gamma(s) = \frac{1 - \gamma^{h(s)}}{1 - \gamma}, \tag{2}$$

where $h(s)$ is the heuristic function. This provides a discounted heuristic which can be applied through Potential-Based Reward Shaping as

$$r' = r - (h_\gamma(s') - h_\gamma(s)), \tag{3}$$

where the discount is applied within the heuristic function.

Applying Misleading Heuristics to RL

As previously indicated, heuristics are naturally prone to some error rate, but it is not clear how much error can be tolerated by RL algorithms. To examine this, it is necessary to isolate the effect of misleading heuristics across different algorithms, so the following criteria were designed for:

- The RL algorithm can reliably learn the chosen task.
- There is an "efficient" path, naturally preferred by an RL agent seeking to maximise discounted rewards.
- A simple heuristic can guide the RL algorithms to reach the goal along an "inefficient" path.
- The error rate of the heuristic can be controlled precisely.

The following design decisions satisfy these criteria.

Design Decisions

To avoid simply re-benchmarking RL algorithms' relative ability to complete tasks, this project uses the well known *Frozen Lake* from *Farama Gymnasium* (Towers et al. 2024) as a discrete environment and *Point Maze* from *Gymnasium-Robotics* (Fu et al. 2020) as a continuous environment. These are shown in Figure 1 with Point-Maze on the left and Frozen-Lake on the right.

Frozen-Lake is a configurable grid-world, with each grid square either 'ice' or 'hole'. The agent must make its way across the ice from a start location to the goal. Available actions are up, down left or right. Moving into the edge of the map results in staying still and moving into a hole results in the episode ending. In this work, deterministic mode rather than stochastic mode is used.

Point-Maze is similarly configurable by a text grid which is translated by the engine into 3D blocks of either wall or open space. Start and goal locations are also defined. The agent is a ball, moved by applying a vector force in $[x, y]$ to the ball, a physics engine calculating movement.

These are both common trial environments for RL algorithms and, as such, are useful when exploring specific aspects of learning algorithms. The specified paths are shown in Figure 1, providing an efficient diagonal path which maximises discounted episodic returns and a less efficient path along the left-hand and bottom edges. By providing an environment which is simple to learn, this work focuses on the effects on learning of misleading heuristics and avoids comparative evaluation of base algorithm capability.

Four RL algorithms were selected for this experiment to cover a range of algorithm types available in popular RL libraries (Liang et al. 2018; Hoffman et al. 2020; Hill et al. 2018; Huang et al. 2022) and observe any differences.

- Tabular Q-learning is used with Frozen Lake to demonstrate the ideas in a simple manner and demonstrate a baseline effect on learning behaviour.
- Also with Frozen Lake, DQN is used as the initial deterministic off-policy DRL comparison.
- SAC, a stochastic off-policy actor-critic algorithm for continuous actions spaces, is used with Point Maze.
- PPO is a popular policy-gradient derived alternative and is selected for comparison within Point Maze.

Since these algorithms are well studied, with many additions and refinements tried by researchers, the well known *CleanRL*[3] (Huang et al. 2022) implementations, which in-

[3]https://github.com/vwxyzjn/cleanrl

clude the full implementation of each algorithm in a single file, are used as a basis. This ensures a focus on the effects of heuristics rather than the performance of chosen RL algorithm features. To assist learning in the Point Maze environment, a top-down 2D 'map' is supplied to the agents. The architecture for the *convolutional neural network (CNN)* (LeCun et al. 1989) is taken directly from the DrQ-v2[4] paper (Yarats et al. 2022) and adjusted to suit this environment.

Hyperparameter Tuning

DRL is sensitive to hyperparamer choice so, for the continuous environment, automated tuning was completed using a *Genetic Algorithm (GA)* (Yang and Shami 2020) run for 15 generations using a population size of 15 with 4 elites, 50% multi-point crossover, 10% mutation and complements (Louis and Rawlins 1993). Encouraging fast learning and high test results,

$$\textit{fitness} = [\frac{3}{10N}\sum_{i=1}^{N} r_{e_i} + \frac{7}{10M}\sum_{j=1}^{M} r_{t_j}], \tag{4}$$

where r_{e_i} is the reward for evaluation episode i and r_{t_j} is the reward for test episode j. With a random seed for each run, a policy evaluation of 15 episodes is completed every 5,000 steps. Following hyperparameter search, the program executes 10 test runs of the fittest hyperparameters with unique seeds, all algorithms successfully learning to reach the goal with sparse rewards and no heuristic.

Heuristics

The choice to provide a heuristic which guides the agent along the edges of the environment instead of the efficient diagonal route is to ensure is to ensure clarity as to when the agent is no longer following the heuristic. This is because it is not possible to distinguish reaching the goal due to heuristic-following and reaching the goal through natural exploration of the environment since both solutions will share the same route. However, in the setup as designed, there are four options; follow the heuristic to the goal, follow the natural diagonal to the goal, fail to reach the goal while following the heuristic, and fail to reach the goal while following the natural diagonal path. This makes it easy to identify whether the agent is following the heuristic or not and the extent to which it has failed. Additionally, there are many reasons why it is desirable to follow an apparently inefficient route. For instance, we may need to follow some regulations such as traffic laws and property rights, or an area may be known to be unsafe or difficult for a given embodiment to traverse. Understanding an agent's ability to follow heuristics along an inefficient route and in the face of misleading errors is therefore valuable in itself.

For Frozen Lake the heuristic is simple. Each cell in the grid is given a value which increases from 1 in the start position to 28 in the goal location. This represents an increase in value of 1 for each cell advancing along a desired path down the left-hand side and along the bottom edge. Cells not along this path have the value 0.2. The reward is adjusted using Potential-Based Reward Shaping (Ng, Harada, and Russell 1999) defined in Equation 1. Advancing along the desired path returns a rich reward above 0, while retreating or moving off the desired path gives a negative reward.

During the experiments, agents were trained 15 times with each of the set of heuristic error rates $E \doteq \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The error is introduced by the following expression:

$$r' = r + \begin{cases} H(s,s') & X \sim U(0,1) < e \in E, \\ -H(s,s') & \text{otherwise}, \end{cases} \tag{5}$$

where $H(s,s')$ is the heuristic function in terms of the current and next states, e is the selected error rate and X is a random value from a uniform distribution in the range $[0,1)$.

Similarly, the continuous Point Maze environment uses distance from goal as a heuristic. Distance is not used directly since it naturally reduces as the agent approaches the goal. To manage this, the numerically stable approach proposed by Gehring et al. (2022) and set out in Equations 2 and 3 is applied as follows:

$$h_\gamma(s) = \begin{cases} \frac{1-\gamma^{h(s)}}{1-\gamma} & \text{moving in desired region}, \\ 0 & \text{stationary in desired region}, \\ -1.0 & \text{outwith desired region} \end{cases} \tag{6}$$

$$r' = r - (h_\gamma(s') - h_\gamma(s)), \tag{7}$$

where γ is a discount factor, s and s' are the current state and next states, and h is the heuristic function based on distance. Note, the discount is part of the heuristic function so the additional discount of Potential-Based Reward Shaping is not required. If naively applied the heuristic value reduces as the agent nears the goal, which is not desirable. This is most simply shown numerically, for instance if we assume the agent is 5.2 m from a goal and moves to 5.1 m away, and later from 0.2 m to 0.1 m away:

$$\begin{aligned} v' &\doteq -(\gamma\frac{1-\gamma^{h(s')}}{1-\gamma} - \frac{1-\gamma^{h(s)}}{1-\gamma}) \\ v'_A &= -(0.99\frac{1-0.99^{5.1}}{1-0.99} - \frac{1-0.99^{5.2}}{1-0.99}) \\ v'_B &= -((0.99\frac{1-0.99^{0.1}}{1-0.99} - \frac{1-0.99^{0.2}}{1-0.99}) \\ v'_A &\approx 0.15 > v'_B \approx 0.1 \end{aligned}$$

It should also be noted that a misleading heuristic is not the same as introducing random noise to the heuristic value. The practical effect of this kind of noise is shown in Figure 2, where adding noise to the heuristic at a rate of 1.0 does not cause learning to fail until $X \sim U(-n, n)$, where $n > \max_{s\sim S}(h_\gamma(s') - h_\gamma(s))$. At this point, while not failing, the agent no longer follows the heuristic but instead learns the efficient diagonal route. The decision of whether to follow the heuristic is made at each decision point (e.g. to move down or diagonally) with the heuristic followed when $n' < \max_{s\sim S}(h_\gamma(s') - h_\gamma(s))/2$ at that decision point. Therefore, to ensure the heuristic error is controllably seen as misleading by the algorithm, with poor actions seen as

[4]https://github.com/facebookresearch/drqv2

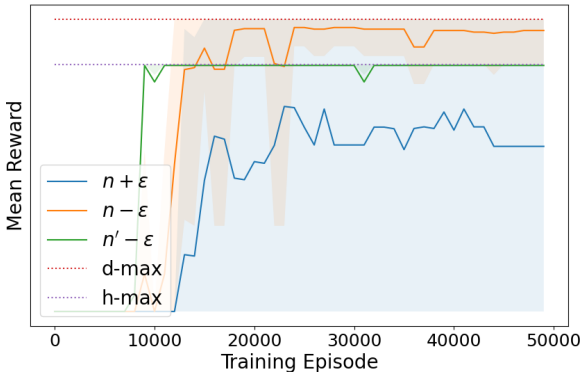

Figure 2: The results of adding a random value $X \sim U(-n, n)$ to the calculated heuristic. The d-max and h-max lines show the maximum rewards by following a diagonal policy and a heuristic policy respectively. Learning fails at $n + \epsilon$, and the heuristic is reliably followed at $n' - \epsilon$.

good and good action seen as poor, Equations 5 and 7 are used instead of a random noise function.

The following sections summarise key results, present analysis of these, and discuss their implications for Reinforcement Learning with Heuristic guidance.

Results and Discussion

Each of the 4 algorithms, Q-Learning, DQN, SAC and PPO, were trained 15 times at each error rate using random seeds. Periodically through training each agent completed 15 evaluation episodes. Average scores of these evaluation episodes are used to evaluate algorithm performance. Overall, results show that RL algorithms can be tolerant to misleading heuristics. This is summarised in Table 1, showing a high level of misleading heuristic is required before meaningful degradation to learning occurs and before learning becomes untenable. Compute for experiments was a HP ZBook with 128 GB RAM, Intel Core i9-13950HX CPU and Nvidia RTX 5000 Mobile Ada GPU. Code is written in Python using PyTorch for DRL models and runs are multithreaded where possible. These results are discussed in detail below.

Frozen Lake

The Heuristics section, along with Figure 2, introduces the fact that random noise applied to the heuristic reward values is not enough to prevent the agent finding a good policy, even when the noise rate reaches 0.8. This is good news for the use of heuristic methods where the ordinality of a heuristic may be correctly identified by a heuristic function but the magnitude of the value inconsistently stated. For instance, while using LLMs to generate heuristics, Hu and Sadigh (2023) used a log function to squash heuristic range, the LLM able to correctly identify good and poor actions but returning values across a larger range than is desirable for RL rewards, a useful idea. The remainder of the results relate to the far more problematic misleading errors.

As shown in Figure 4, without heuristics the Tabular Q-Learning agent reliably learns an efficient diagonal route. The heatmap shows that after some initial evaluation runs where the agent tries to navigate the perimeter route, the policy settles on the expected diagonal trajectory. With a baseline heuristic, the agent quickly learns the inefficient route down the left edge and along the bottom, never trying the diagonal route during evaluation due to the additional heuristic guidance. This indicates the agent learns from heuristic reward shaping as opposed to general exploration, demonstrating its effectiveness and suggesting the approach is suitable to evaluate RL algorithms' tolerance to errors.

Once misleading errors are introduced, even at a rate of 0.1, initial learning takes approximately twice as long and becomes less reliable, seen as an increasing spread of results in Figure 5. However, it is not until error rates approach 0.4 that learning becomes problematically unreliable whereby agents become unlikely to find a policy.

DQN learns in fewer episodes than Tabular Q-Learning and learning is initially stable, as shown in Figure 3. Once errors are introduced at an error rate of 0.1 it continues to show similar behaviour to Tabular Q-Learning in that it takes longer to learn and shows some instability. However, increasing the error rate further significantly increases learning instability. Some instability is expected as an effect of DQN's DNN function estimation, but policy stability is lower than expected for such a simple environment. This may be a result of the replay buffer repeatedly applying incorrect heuristics through recorded heuristically-adjusted rewards, a theme identified through these results and discussed further when reviewing SAC and PPO.

While instability is observed during training, only one agent failed to learn a good policy after 10,000 episodes through all training to an error rate of 0.3 (60 runs to this point), the main cost therefore being sample efficiency in achieving the policy. However, the instability suggests that in more complex environments the agents may struggle to learn effectively and so DQN may not be a good option if the heuristic may produce misleading indicators. Since Frozen Lake is a discrete environment the trajectories with and without heuristic are the same for DQN as they are for Tabular Q-Learning and are not repeated.

Point Maze

As shown in Figure 3, the SAC algorithm learns quickly with the accurate heuristic, reliably learning a good policy. With an error rate of 0.1, training is a little more unstable at first but quickly stabilises to a reliable policy. At an error rate of 0.3 the instability is clear, and while it does learn a policy the sample efficiency is significantly reduced and the learned policy is less stable. At an error rate of 0.4, the agent is unable to reliably learn. Using SAC with heuristics, unless the error rate is known to be below 0.2, could be problematic since results indicate both sample efficiency and stability of learning may be meaningfully affected. In a complex environment which already needs millions of steps, training times may become prohibitive.

It is likely that learning methods which sample from a replay buffer, such as DQN and SAC, are more affected by misleading heuristics because errors are stored and then repeatedly applied during updates. This occurs because it

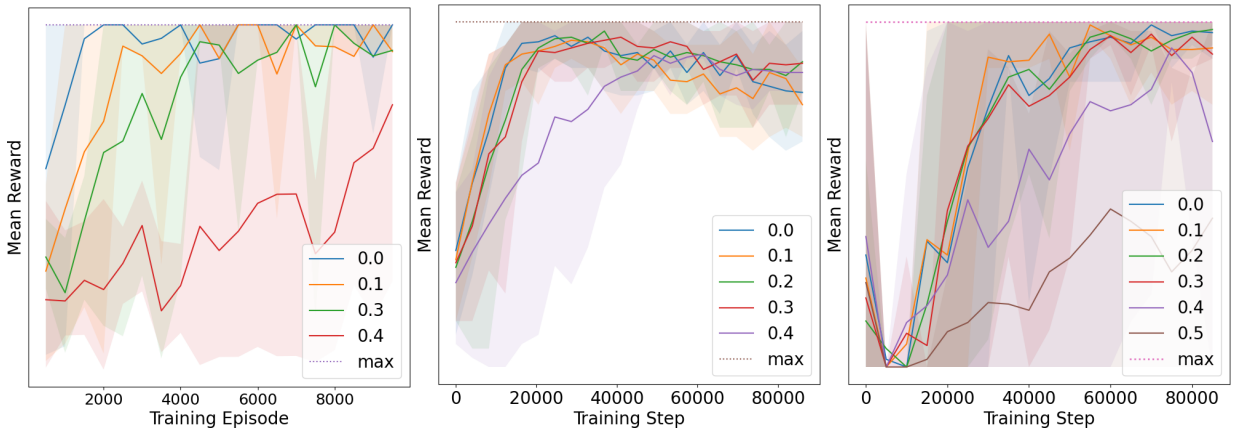

Figure 3: DQN (left) in Frozen Lake shows that it learns quite quickly and reliably with no errors. As error rates increase, stability quickly declines. PPO (center) in Point Maze shows good stability and is not significantly affected by the misleading heuristic until error rates approach 0.4. SAC (right) similarly becomes untenable at an error rate of 0.4, but lower error rates of 0.2 and 0.3 also show clear increases in instability compared to baseline. Max line is max achievable heuristic-guided rewards.

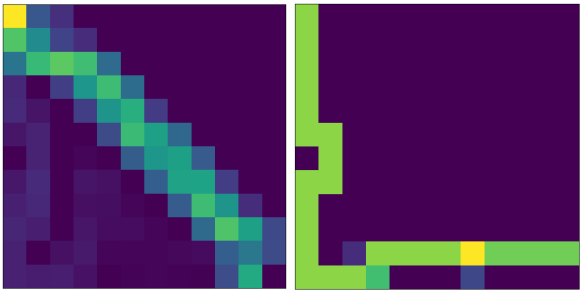

Figure 4: Locations traversed during evaluation with no heuristic (left) and with error-free heuristic (right).

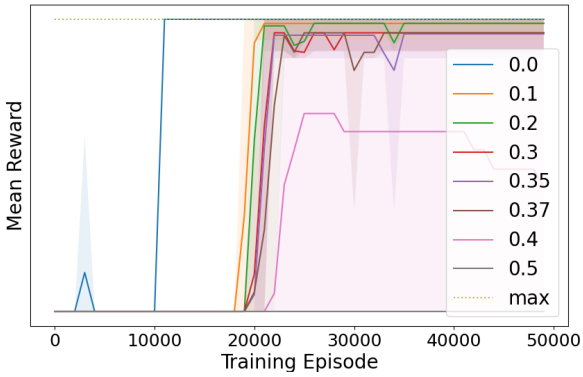

Figure 5: Q-Learning evaluation episode results during training at different heuristic errors showing the mean, 1st and 9th deciles of the 15 evaluation episodes run every 1000 episodes (left). Agent learning becomes less stable until, with error approaching 0.4, it no longer learns reliably.

is more computationally efficient to calculate the heuristic once during exploration and store the resulting r' in the replay buffer (the approach chosen for this project). To mitigate this, an alternative approach could be to recalculate the heuristic during gradient updates. This requires more compute but would prevent errors being repeatedly applied. Exploration of mitigations is left to future research.

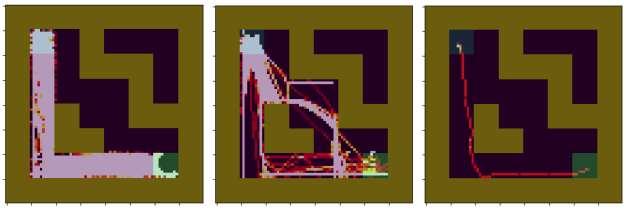

Figure 6: Locations traversed by SAC agents with 0.1 error rate during training (left), evaluation (middle) and 15 test episodes (right). Start position is top left and goal is bottom right. Agents try the diagonal route during evaluation, but the final policy follows the heuristic.

As can be seen from the trajectory plots for SAC in Figure 6, with error rate 0.2 the agent must follow the heuristic-preferred path during training and successfully explores the heauristic-augmented rich rewards to the goal. During evaluation the agent attempts many different trajectories, some of which involve the diagonal route. However, once the policy has been trained, it remains conditioned on the heuristic-preferred route and in testing shows reliable adherence to the perimeter path. This shows that SAC is not only able to learn from a heuristic, but is also able to learn a sequence of actions which follow defined characteristics. This can be useful if a certain approach to the goal is required, but may not be desirable if the general direction of a goal can be defined heuristically while the most appropriate path can not.

PPO quickly and reliably learns a good policy when there is no heuristic error. The need for *early stopping* or *checkpoint saving* to ensure a good policy is saved can be seen in the middle graph of Figure 3 since the stability of learning starts to degrade as training continues. The effect of errors

on PPO agents is minimal, with error rates to 0.3 learning at approximately the same rate. It is not until the error rate reaches 0.4 that misleading heuristics start to cause meaningful issues with learning stability at which point the agent eventually learns, but policy stability is already degrading. Once the error rate reaches 0.5 learning collapses completely (not shown for graph clarity).

This aspect of PPO learning is similar to Tabular Q-Learning; in both cases the misleading heuristics do not significantly affect learning until an error rate of 0.4. In contrast to DQN and SAC, Tabular Q-Learning and PPO do not require a persistent replay buffer. PPO processes then discards each minibatch, meaning any individual heuristic error is not later re-applied to the policy. Similarly, Tabular Q-Learning updates the policy on each step with the most recent experience, also not storing it. This supports the mitigation idea outlined above and could be interesting for future research. In general, the stability of PPO in the face of misleading errors to an error rate of over 0.3 makes the algorithm a good option for learning with heuristics which may contain errors.

Another meaningful difference between PPO and the other algorithms used is in test behaviour. If a stochastic policy is used during test, the agent will successfully use either route to reach the goal even though it hasn't used the efficient diagonal route during training, see Figure 7. Here, the PPO agent uses a stochastic policy (left) to successfully find the goal during all 15 test episodes even though it doesn't always traverse the heuristic-preferred, or other rational, path. This is despite never seeing the diagonal path during training (middle), meaning it sees the diagonal path as valuable for some reason other than association with collected rewards. Forcing use of the action distribution mean (right) interestingly leads the agent to only use the diagonal route which, to reiterate, it has never traversed during training. This raises some fundamental questions about the nature of learning since, despite the agent directly experiencing that most rewards are to be found around the edge, it has learned that some other sequence of actions is preferable. This fundamental question is left for future research.

This behaviour by PPO agents makes it useful in the opposite situation to SAC. Specifically, where a heuristic for an approximate action sequence can be implemented but the most appropriate action sequence is unknown, the PPO agent may be able to find the more effective policy. However, if a specific action sequence is desirable, it may not be possible to guide PPO to achieve a goal in a specific heuristically-indicated manner leading to unexpected behaviours.

Algorithm	Unstable	Untenable
Q-Learning	> 0.35	< 0.4
DQN	> 0.1	< 0.3
SAC	> 0.2	< 0.3
PPO	> 0.3	< 0.4

Table 1: Rates of misleading heuristic error by RL algorithm showing when training becomes unstable and untenable.

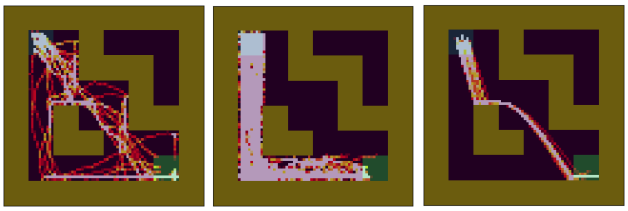

Figure 7: Successful PPO stochastic policy test (left) with 15 episodes uses both routes even though it has never seen the diagonal route in training (middle). Testing with only mean value actions (right) unexpectedly prefers the diagonal route.

Conclusion and Future Work

Heuristics are an effective way to improve learning efficiency of general RL algorithms by conferring domain-specific knowledge to the agent. However, heuristics are estimates by nature, and in some cases may be misleading. Reinforcement Learning is shown in this work to cope well with misleading heuristics, able to reliably learn effective policies with misleading error rates to 0.3. This is good news for RL researchers since it means the common RL algorithm types can make use of heuristics to provide domain knowledge and increase sample efficiency, even where misleading errors are expected, such as where LLMs are used.

As discussed, Tabular Q-Learning is the most clear cut and learning is minimally affected by a misleading heuristic until it approaches an error rate of 0.4, at which point it becomes unstable. Interestingly, PPO appears closest to Tabular Q-Learning in tolerance to misleading heuristics, perhaps because both algorithms discard experiences having updated the policy. DQN and SAC, however, store these experiences and therefore repeatedly apply the error to the policy, perhaps making them more susceptible to misleading heuristics and causing the higher learning instability seen in Figure 3. Identifying mitigations for this is left to future research; the author's initial suggestion is to recalculate the heuristic during policy updates, although this will increase compute.

A difference between how SAC and PPO execute policies is also noted in results, with SAC following the heuristically-defined path and PPO preferring a more efficient route. Both can be useful, with SAC a good option where following the heuristic-indicated path is desirable and PPO where the desired policy can't be defined heuristically. What this implies about the nature of agent learning is left to future work.

DQN is most affected by misleading heuristics and the difference in performance to Tabular Q-Learning is significant, increasing error rates making a clear difference to early learning and overall stability. Despite this, DQN does eventually learn a good policy until error rates approach 0.4.

This work has focused on a simple navigation task to set a baseline of RL algorithm tolerance to misleading heuristics, showing RL is capable of learning in the face of fairly significant error rates. Further experiments by the authors will explore other tasks, such as grasping, to determine whether they show similar results.

References

Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; and Zaremba, W. 2017. Hindsight Experience Replay. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NeurIPS)*.

Brown, N.; Bakhtin, A.; Lerer, A.; and Gong, Q. 2020. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. In *International Conference on Neural Information Processing Systems (NeurIPS)*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In Dy, J.; and Krause, A., eds., *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, 1587–1596.

Gehring, C.; Asai, M.; Chitnis, R.; Silver, T.; Kaelbling, L.; Sohrabi, S.; and Katz, M. 2022. Reinforcement Learning for Classical Planning: Viewing Heuristics as Dense Reward Generators. *International Conference on Automated Planning and Scheduling (ICAPS)*, 32: 588–596.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Dy, J.; and Krause, A., eds., *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, 1861–1870.

Hanover, D.; Loquercio, A.; Bauersfeld, L.; Romero, A.; Penicka, R.; Song, Y.; Cioffi, G.; Kaufmann, E.; and Scaramuzza, D. 2023. Autonomous Drone Racing: A Survey. *arXiv:2301.01755*.

Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2018. Stable Baselines. https://github.com/hill-a/stable-baselines.

Hoffman, M. W.; Shahriari, B.; Aslanides, J.; Barth-Maron, G.; Momchev, N.; Sinopalnikov, D.; Stańczyk, P.; Ramos, S.; Raichuk, A.; Vincent, D.; Hussenot, L.; Dadashi, R.; Dulac-Arnold, G.; Orsini, M.; Jacq, A.; Ferret, J.; Vieillard, N.; Ghasemipour, S. K. S.; Girgin, S.; Pietquin, O.; Behbahani, F.; Norman, T.; Abdolmaleki, A.; Cassirer, A.; Yang, F.; Baumli, K.; Henderson, S.; Friesen, A.; Haroun, R.; Novikov, A.; Colmenarejo, S. G.; Cabi, S.; Gulcehre, C.; Paine, T. L.; Srinivasan, S.; Cowie, A.; Wang, Z.; Piot, B.; and de Freitas, N. 2020. Acme: A Research Framework for Distributed Reinforcement Learning. *arXiv preprint arXiv:2006.00979*.

Hu, H.; and Sadigh, D. 2023. Language Instructed Reinforcement Learning for Human-AI Coordination. In *International Conference on Machine Learning (ICML)*.

Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; and Liu, T. 2024. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*.

Huang, S.; Dossa, R. F. J.; Ye, C.; Braga, J.; Chakraborty, D.; Mehta, K.; and Araújo, J. G. M. 2022. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 23(274): 1–18.

Kwon, M.; Xie, S. M.; Bullard, K.; and Sadigh, D. 2023. Reward Design with Language Models. In *International Conference on Learning Representations (ICLR)*.

LeCun, Y.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W.; and Jackel, L. 1989. Handwritten Digit Recognition with a Back-Propagation Network. In Touretzky, D., ed., *Advances in Neural Information Processing Systems (NeurIPS)*.

Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J. E.; Jordan, M. I.; and Stoica, I. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In Bengio, Y.; and LeCun, Y., eds., *International Conference on Learning Representations (ICLR)*.

Louis, S. J.; and Rawlins, G. J. 1993. Syntactic analysis of convergence in genetic algorithms. In *Foundations of Genetic Algorithms*, volume 2, 141–151. Elsevier.

Meta Fundamental AI Research Diplomacy Team (FAIR)†; Bakhtin, A.; Brown, N.; Dinan, E.; Farina, G.; Flaherty, C.; Fried, D.; Goff, A.; Gray, J.; Hu, H.; Jacob, A. P.; Komeili, M.; Konath, K.; Kwon, M.; Lerer, A.; Lewis, M.; Miller, A. H.; Mitts, S.; Renduchintala, A.; Roller, S.; Rowe, D.; Shi, W.; Spisak, J.; Wei, A.; Wu, D.; Zhang, H.; and Zijlstra, M. 2022. Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning. *Science*, 378(6624): 1067–1074.

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

Nachum, O.; Gu, S. S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NeurIPS)*.

Ng, A. Y.; Harada, D.; and Russell, S. J. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *International Conference on Machine Learning (ICML)*, 278–287.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven Exploration by Self-supervised Prediction.

In Precup, D.; and Teh, Y. W., eds., *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, 2778–2787.

Qiao, S.; Fang, R.; Zhang, N.; Zhu, Y.; Chen, X.; Deng, S.; Jiang, Y.; Xie, P.; Huang, F.; and Chen, H. 2024. Agent Planning with World Knowledge Model. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In Meila, M.; and Zhang, T., eds., *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, 8748–8763.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; Lillicrap, T.; and Silver, D. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, second edition. ISBN 978-0-262-03924-6.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112(1–2): 181–211.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; Cola, G. D.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; Perez-Vicente, R.; Pierré, A.; Schulhoff, S.; Tai, J. J.; Tan, H.; and Younis, O. G. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv:2407.17032.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards*. Ph.D. thesis, King's College, Cambridge, UK.

Yang, L.; and Shami, A. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415: 295–316.

Yarats, D.; Fergus, R.; Lazaric, A.; and Pinto, L. 2022. Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

Zare, M.; Kebria, P. M.; Khosravi, A.; and Nahavandi, S. 2024. A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges. *IEEE Transactions on Cybernetics*, 1–14.