Mischievous Sibling's Grid World

Aykut C. Satici[a]

[a]*Boise State University, Mechanical and Biomedical Engineering, Boise, 83725, Idaho, USA*

Abstract

This is a technical note that introduces a novel grid world environment and an optimal control problem that is meant to be solved by a reinforcement learning (RL) agent in one-shot. That is to say, the algorithm that the RL agent runs is supposed to be able to solve the environment by training only for a single episode. The environment is very similar to the well-known Grid World problem, but with a twist. We solve the problem in two different ways: first using a direct probability calculation and then using reinforcement learning on a judiciously designed Markov Decision Process. The results are compared and discussed. The numerical solution to the problem allows us to provide further statistics that sheds some more insight into the problem.

Keywords: Probability, Expectations, Reinforcement learning, Gymnasium, Policy iteration

1. Introduction

Grid World environments are frequently utilized to establish baselines for many novel reinforcement learning algorithms. These environments are simpler compared to other discrete environments such as chess and go; they have much smaller state and action spaces. Yet, they still require reasoning for a sequential, evaluative and sampled feedback problem. In this sense, these set of problem provide a good entry point to either test new algorithms or start learning basic concepts of reinforcement learning.

In this note, we consider the ordinary instance of a stochastic grid world environment, in which the problem is to find the shortest path from a start state to a goal state. However, in this case, the task is to do this in a one-shot manner. That is, the agent must learn to find the optimal policy in a single episode. This is a challenging problem, as the agent must learn to explore the environment, and exploit the information it has gathered to reach the goal state in the fewest number of steps possible. Furthermore, the reinforcement learning problem we pose to solve this instance of the grid world problem requires learning two Q-functions as opposed to one; one corresponding to a familiar Q-function for a classical grid world, and the other a Q-function for a related multi-armed bandit problem. The agent must learn to balance the exploration of the grid world environment with the exploitation of the multi-armed bandit problem to solve the task optimally. We coin the term "sibling's grid world" to describe this problem.

The algorithms we use to solve the sibling's grid world problem are very well studied and documented in the literature Sutton and Barto (2018). We use either value iteration or its linear programming equivalent Luenberger and Ye (2008) to solve the Bellman equation to find the optimal state- and action-value functions of the grid world, given the true world belief. The multi-armed bandit subproblem's Q-function is found iteratively by updating its value iteratively by the reward received by the robot each time it takes a step. The policy the agent uses is then just the greedy policy with respect to the value function under the current world belief the agent holds while updating its multi-armed bandit's Q-function.

The contribution of this paper is in introducing a baseline grid world environment in which one-shot RL algorithms may be evaluated, providing a theoretical solution to this problem, and comparing the performance of a baseline Q-learning RL solution to the theoretical solution.

We provide a thorough statement of this problem in Section 2, before providing a theoretical solution in the first subsection of Section 3. The subsequent subsection of the same section then provides our formulation of the reinforcement learning problem to solve the sibling's grid world in a one-shot manner. We then present the results of our RL formulation in Section 4, comparing and contrasting against the theoretical solution in Section 3.1. We conclude with a discussion of the results and potential future work in Section 5.

2. Problem Statement

Alice has a 5×5 chess board as shown in Figure 1, where each square is described by its coordinates $(i, j) \in \{0, 1, 2, 3, 4\}^2$. Her remote-controlled robot starts on the square $(2, 2)$. The remote-controller has 4 buttons that can move the robot east, north, west, and south by 1 square with each press. Bob, her mischievous sibling, has tampered with the wiring of these buttons. Assuming that the buttons are fixed after having shuffled once, what is the expected value of the minimum number of times Alice needs to press the buttons in order to move the robot to the goal square $(4, 4)$?

In this setup, we assume that the agent has knowledge of the environment it is in; i.e., the agent knows the map of the environment. This means, in particular, that the agent would

*Corresponding author
Email address: aykutsatici@boisestate.edu (Aykut C. Satici)

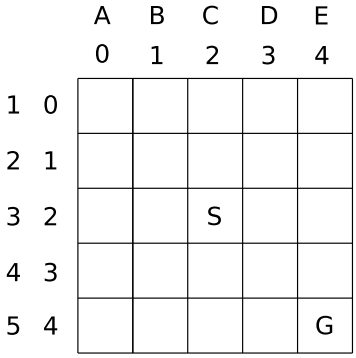

Figure 1: Schematic of the problem.

know that it needs to try a different key whenever it reaches the wall of the chessboard by hitting a particular key (since hitting this key obviously would try and move the robot into the wall, wasting a key press).

A second observation to be made is that the agent should not make any more mistakes than 2 because, once 2 mistakes are made, the remaining actions would make progress towards the goal and the agent has time to register the fact that it is making progress. Hence, the worst-case scenario is that the agent takes 8 steps to solve the environment. Of course, it may get lucky and stumble upon the right combination of motions ("right" and "down") in the first two tries, reaching the goal position in 4 steps; however, this is not guaranteed.

If we linearly index the squares of the chessboard from 0 to 24 in a row-major fashion, then the mapping from the linear index to the (i, j) coordinates and back is given by the following bijections:

$$5j + i = k \leftrightarrow \left(i = k - 5\left\lfloor \frac{k}{5} \right\rfloor, j = \left\lfloor \frac{k}{5} \right\rfloor\right) \tag{1}$$

This mapping is useful and relevant for implementing the Q-learning algorithm for the RL solution to the problem.

3. Solutions

We present two solutions to the problem: a probabilistic solution and a reinforcement learning (RL) solution. The probabilistic solution will chase the robot's possible motions as it tries to reach the goal state, computing all the relevant expectations along the way. The RL solution will identify which MDP is to be solved rigorously, and will attack the problem using a Q-learning approach. It will need to keep track of two Q-functions as will be explained in the subsection that follows the next.

3.1. Probability Solution

We invoke a celebrated theorem from probability theory that will help us solve the problem. This theorem is called the *law of total expectation* or *tower rule of probability theory* in the literature Bertsekas and Tsitsiklis (2002). In this section, we use the correspondence $\{1, 2, 3, 4, 5\} \mapsto \{0, 1, 2, 3, 4\}$ and $\{A, B, C, D, E\} \mapsto \{0, 1, 2, 3, 4\}$ for ease of exposition.

Theorem 1 (Tower rule). *Let X be a random variable and $\{A_i\}_i^m$ be a finite partition of the sample space. Then,*

$$\mathbb{E}[X] = \sum_i^m \mathbb{E}\left[X \mid A_i\right] \mathbb{P}(A_i).$$

We denote the square on which the robot resides at the k^{th} step by s_k and the goal square by g. Consider the following family of random variables.

$$C_n := \sum_{k=n+1}^{\infty} r_k, \qquad r_k = \begin{cases} 1 & \text{if } s_k \neq g \\ 0 & \text{if } s_k = g \end{cases}. \tag{2}$$

The sum in this definition is well-defined because once the robot reaches g, all the remaining r_k's take the value zero. We are being asked the value of $\mathbb{E}[C_0]$. To that end, we will use the following repercussion of definition (2).

$$C_m = \sum_{i=m+1}^{n} r_i + C_n, \qquad m \leq n.$$

Using the properties of expectation, if $s_i \neq g$ for $m < i \leq n$, we can deduce the following result.

$$\mathbb{E}[C_m] = \sum_{i=m+1}^{n} \mathbb{E}[r_i] + \mathbb{E}[C_n] = n - m + \mathbb{E}[C_n].$$

Similar identities to the above expression hold for conditional expectations as well. Let us start to compute the expectation of the "value" of being in state $C3$ using the tower rule 1. Of course, by "value," we mean the expected minimum number of times a button needs to be pressed in order to reach the goal state.

$$\begin{aligned} \mathbb{E}[C_0] = \mathbb{E}\left[C_0 \mid s_1 \in \{C4, D3\}\right] \underbrace{\mathbb{P}(s_1 \in \{C4, D3\})}_{=\frac{1}{2}} \\ + \mathbb{E}\left[C_0 \mid s_1 \in \{B3, C2\}\right] \underbrace{\mathbb{P}(s_1 \in \{B3, C2\})}_{=\frac{1}{2}}. \end{aligned} \tag{3}$$

From now on, we will assume $s_1 = C4$ in the first term on the right-hand side and $s_1 = C2$ in the second. Notice that, because of the inherent symmetry of the problem, while the state $s_1 = D3$ is equivalent to $s_1 = C4$; $s_1 = B3$ is equivalent to $s_1 = C2$ in terms of their value.

Computation of the first conditional expectation in (3). First of all, we consider the first expectation term on the right-hand side of equation (3). If at the first step, we found ourselves in square $C4$, that means we made progress towards the goal. We can keep this greedy motion for one more turn since it will get us closer to the goal. Hence, $s_2 = C5$, with a cost of $r_2 = 1$. Since at step 2, we hit the south wall, we need to change the button to press. We have 3 options of equal likelihood. Choosing one

will take us either to $D5$, back to $C4$, or to $B5$. Hence,

$$\begin{aligned}\mathbb{E}\left[C_0 \mid s_1 = C4\right] &= \overbrace{r_1 + r_2}^{=2} \\ &+ \overbrace{\mathbb{E}[C_2 \mid s_1 = C4, s_3 = D5]}^{=2} \overbrace{\mathbb{P}(s_3 = D5 \mid s_1 = C4)}^{=\frac{1}{3}} \\ &+ \mathbb{E}[C_2 \mid s_1 = s_3 = C4] \underbrace{\mathbb{P}(s_3 = C4 \mid s_1 = C4)}_{=\frac{1}{3}} \\ &+ \mathbb{E}[C_2 \mid s_1 = C4, s_3 = B5] \underbrace{\mathbb{P}(s_3 = B5 \mid s_1 = C4)}_{=\frac{1}{3}}.\end{aligned} \tag{4}$$

We delve further into the computation of the two expected values in equation (4), whose values are not immediately apparent. If at step 3 we found ourselves back on $C4$, we got farther from the goal, but we can repeat our previous move to get back to $C5$ on step 4. This maneuver costs us $r_3 + r_4 = 2$ units. Now, we know what directions two of the keys map to. The remaining expectations are computed below.

$$\begin{aligned}\mathbb{E}[C_2 \mid s_1 = s_3 = C4] &= \underbrace{r_3 + r_4}_{=2} \\ &+ \underbrace{\mathbb{E}[C_4 \mid s_1 = s_3 = C4, s_5 = D5]}_{=2} \underbrace{\mathbb{P}(s_5 = D5 \mid s_1 = s_3 = C4)}_{=\frac{1}{2}} \\ &+ \underbrace{\mathbb{E}[C_4 \mid s_1 = s_3 = C4, s_5 = B5]}_{=4} \underbrace{\mathbb{P}(s_5 = B5 \mid s_1 = s_3 = C4)}_{=\frac{1}{2}}.\end{aligned}$$

There is only one expected value that we have left to compute in equation (4). Again, if we found ourselves on the $B5$ square at step 3, we have moved farther away from the goal, but do now know how to get back to $C5$ in this case. Hence, we need to try out the final key to figure out the full mapping. The expectations are computed below.

$$\begin{aligned}\mathbb{E}[C_2 \mid\ & s_1 = C4, s_3 = B5] = \\ &\underbrace{\mathbb{E}[C_2 \mid s_1 = C4, s_3 = B5, s_4 = C5]}_{=4} \\ &\quad \times \underbrace{\mathbb{P}(s_4 = C5 \mid s_1 = C4, s_3 = B5)}_{=\frac{1}{2}} \\ &+ \underbrace{\mathbb{E}[C_2 \mid s_1 = C4, s_3 = B5, s_4 = B4]}_{=6} \\ &\quad \times \underbrace{\mathbb{P}(s_4 = B4 \mid s_1 = C4, s_3 = B5)}_{=\frac{1}{2}}.\end{aligned}$$

The computations above allows us to determine the first expected value in equation (3) using equation (4): $\boxed{\mathbb{E}[C_0 \mid s_1 = C4] = 6}$.

Computation of the second conditional expectation in (3). We use similar techniques to compute the second expected value on the right-hand side of equation (3). If on step 2, we find ourselves on square $B2$, then, we have identified all actions that take us farther from the goal. The two remaining actions both get us closer to the goal, from which we are 8 steps away.

$$\begin{aligned}\mathbb{E}\left[C_0 \mid s_1 = C2\right] &= \\ &\underbrace{\mathbb{E}[C_0 \mid s_1 = C2, s_2 = B2]}_{=8} \underbrace{\mathbb{P}(s_2 = B2 \mid s_1 = C2)}_{=\frac{1}{3}} \\ &+ \mathbb{E}[C_0 \mid s_1 = C2, s_2 = C3] \underbrace{\mathbb{P}(s_2 = C3 \mid s_1 = C2)}_{=\frac{1}{3}} \\ &+ \mathbb{E}[C_0 \mid s_1 = C2, s_2 = D2] \underbrace{\mathbb{P}(s_2 = D2 \mid s_1 = C2)}_{=\frac{1}{3}}.\end{aligned} \tag{5}$$

Once more, we utilize the tower rule to compute the expected values in equation (5) whose values are not immediately apparent. If at step 3, we end up on square $C3$, we can keep repeating this move until we reach $C5$ at step 4, incurring a cost of $r_1 + r_2 + r_3 + r_4 = 4$ units. From here, the only two possibilities is we get to square $D5$ or $B5$ on step 5. If we reach square $D5$, then the same motion will get us to the goal, hence the corresponding expected value is 2. If, on the other hand, we end up on square $B5$ on move 5, we have moved farther away from the goal, but now we know which button to press to get to the goal, resulting in an expected value of 4.

$$\begin{aligned}\mathbb{E}[C_0 \mid s_1 = C2, s_2 = C3] &= \overbrace{r_1 + r_2 + r_3 + r_4}^{=4} \\ &+ \underbrace{\mathbb{E}[C_4 \mid s_1 = C2, s_2 = C3, s_5 = D5]}_{=2} \\ &\quad \times \underbrace{\mathbb{P}(s_5 = D5 \mid s_1 = C2, s_2 = C3)}_{=\frac{1}{2}} \\ &+ \underbrace{\mathbb{E}[C_4 \mid s_1 = C2, s_2 = C3, s_5 = B5]}_{=4} \\ &\quad \times \underbrace{\mathbb{P}(s_5 = B5 \mid s_1 = C2, s_2 = C3)}_{=\frac{1}{2}}.\end{aligned}$$

There is only one expected value that we have left to compute in equation (5). If we find ourselves on square $D2$ at step 2, we can repeat this direction to reach $E2$ at step 3, incurring a cost of $r_1 + r_2 + r_3 = 3$ units. From here, we can either go to $E3$ or $D2$ with the remaining keys. If we find ourselves on $E3$, we can repeat the same motion to reach the goal in 3 more steps. If we find ourselves on $D2$ at step 4, we need to get back to $E2$ and use the remaining key to reach the goal, incurring a cost of 5 units.

$$\begin{aligned}\mathbb{E}[C_0 \mid s_1 = C2, s_2 = D2] &= \overbrace{r_1 + r_2 + r_3}^{=3} \\ &+ \underbrace{\mathbb{E}[C_3 \mid s_1 = C2, s_2 = D2, s_4 = E3]}_{=3} \\ &\quad \times \underbrace{\mathbb{P}(s_4 = E3 \mid s_1 = C2, s_2 = D2)}_{=\frac{1}{2}} \\ &+ \underbrace{\mathbb{E}[C_3 \mid s_1 = C2, s_2 = s_4 = D2]}_{=5} \\ &\quad \times \underbrace{\mathbb{P}(s_4 = D2 \mid s_1 = C2, s_2 = D2)}_{=\frac{1}{2}}.\end{aligned}$$

We can now use equation (5) in order to compute the second expected value on the right-hand side of equation (3): $\boxed{\mathbb{E}[C_0 \mid s_1 = C2] = \frac{22}{3}}$. We have computed every quantity that we are interested in. Now, we go back to equation (3), and plug in the values we have computed to find the expected value of the minimum number of times Alice needs to press the buttons.

$$\begin{aligned}\mathbb{E}[C_0] = &\underbrace{\mathbb{E}\left[C_0 \mid s_1 \in \{C4, D3\}\right]}_{=6}\underbrace{\mathbb{P}(s_1 \in \{C4, D3\})}_{=\frac{1}{2}} \\ &+ \underbrace{\mathbb{E}\left[C_0 \mid s_1 \in \{B3, C2\}\right]}_{=\frac{22}{3}}\underbrace{\mathbb{P}(s_1 \in \{B3, C2\})}_{=\frac{1}{2}} \\ &= \frac{20}{3} = 6.\bar{6}.\end{aligned}$$

3.2. RL solution

We want to be able to programmatically solve this toy problem so that we can find the solution for any given initial state. We can use reinforcement learning (RL) to solve this problem.

The reasoning goes as follows: we want to apply RL techniques, such as double Q-learning Morales (2020) to find the action-value function and from it the state-value function. However, in order to do this, we need to define the state, action, transition, and reward functions. The key to solving this problem efficiently with RL methods is to observe that the state does not only comprise the position of the robot on the chessboard, but also the "world belief". Here, we define world belief as our current belief of mapping of the arrow keys to the actual directions.

Let $\underline{N} = \{0, 1, \ldots, N-1\}$ denote the set of nonnegative integers ranging from 0 to $N-1$. We model this problem as consisting of two RL subproblems: the first one is the sequential problem of finding the shortest route to the goal position, given the world belief, and the second is to find an appropriate world belief, which can be modeled as a 24-armed bandit problem.

We define the Markov decision processes (MDP) of the subproblems as follows:

3.2.1. State Spaces

The first subproblem has the state space $\mathcal{S}_1 = \underline{4}\times\underline{4}$, where the first factor stands for the column in which the robot is located, and the second stands for the row. The second subproblem is a bandit problem, with a singleton state, the world belief. The full problem may be considered to have the state space $\mathcal{S} = \mathcal{S}_1 \times \{1\} \cong \underline{4} \times \underline{4}$.

3.2.2. Action Spaces

The problem has the action space $\mathcal{A} = \underline{4} \times \underline{24}$. The first factor corresponds to the which key to be pressed given the world belief, which is an element of the second factor of the action space.

3.2.3. Transition function

The transition function is a deterministic function of the true world. The true world has a certain mapping between the keys and the directions, which is hidden from the agent. This key mapping is deterministic albeit hidden and the agent will move in the corresponding directions given the correct key mapping one hundred percent of the time.

3.2.4. Reward function

This function needs to be designed judiciously in order to give the agent the best chance to learn the optimal strategy. What worked well in our experiments is to assign a reward of -1 each time the agent moves into a nonterminal state (a state that is not G) for the grid world subproblem.

For the bandit subproblem, we define the reward as a sum of two terms. The first term is the difference between the value functions of the gridworld problem, given that the current world belief is correct. To explain, suppose that $w \in \underline{24}$ is the current world belief. We perform value iteration to find the value function of the grid world under this assumption and we assign the first term of the reward for the bandit as the value function evaluated at the grid world position at which we land minus the value function evaluated at the current grid world position.

It turns out that it is important to be able to distinguish between whether the desired movement direction is equal to the true direction of motion. For that reason, the second term that we add to the bandit reward signal -2 whenever the executed direction is different from the direction in which the agent expected the robot to move.

Our implementation Satici (2024) of this MDP is carried out using Gymnasium Towers et al. (2024), a Python library for modeling and RL.

3.2.5. Q-Learning

Once the MDP is defined, we apply Q-learning to find the optimal policy. For the grid world subproblem, this is the straightforward Q-learning algorithm that can be found in any RL textbook. For the bandit subproblem, the Q-function is learned by the iteration

$$\texttt{Q}_\texttt{b}\texttt{[a]} = \texttt{Q}_\texttt{b}\texttt{[a]} + (\texttt{reward} - \texttt{Q}_\texttt{b}\texttt{[a]})/\texttt{N[a]},$$

where `N[a]` is the number of times action a has been taken for the bandit problem.

The way we set this problem up, allows us to use RL to learn to get to the goal position in one episode in the last number of moves. Of course, this is a stochastic process, depending on what the initial world belief is selected and what the true world is. We perform Monte Carlo simulations of this setup to determine the statistics, including the expectation of the minimum number of moves (which was solved analytically in Section 3.1).

3.2.6. Grid World State-Value Function given World Belief

The state-value function V of the grid world environment is used to define the reward signal, as described in Section 3.2.4.

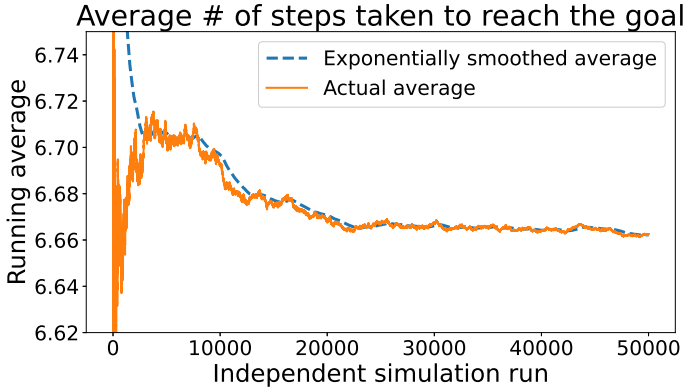

Figure 2: Running average number of steps taken to reach the goal over 50,000 independent Monte Carlo simulation runs.

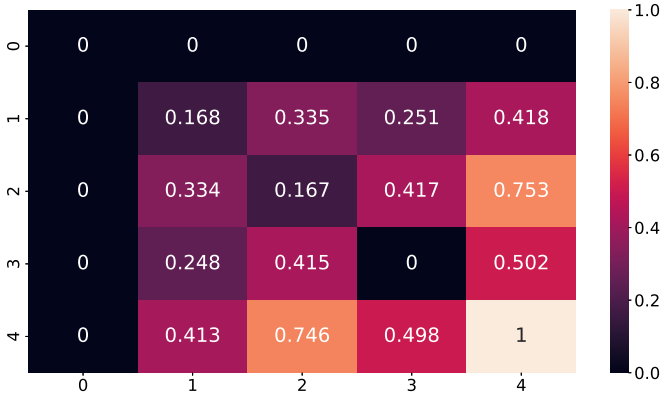

Figure 3: Ratio of times a given state is visited under the optimal policy to the number 50,000 of independent Monte Carlo simulation runs.

This function can be found easily by solving a linear programming problem or by running value iteration. This value function is given by

$$V_{\text{gw}} = \begin{bmatrix} -8 & -7 & -6 & -5 & -4 \\ -7 & -6 & -5 & -4 & -3 \\ -6 & -5 & -4 & -3 & -2 \\ -5 & -4 & -3 & -2 & -1 \\ -4 & -3 & -2 & -1 & 0 \end{bmatrix},$$

where the entry index of the matrix corresponds to the position of the robot. This function is computed assuming that the world belief is correct, hence, it can be computed once and used over and over again while solving the multi-armed bandit problem. When the agent is about to choose an action from the set $\{0, 1, 2, 3\}$, corresponding to the key to press, it acts greedily against this function, given the world belief. Hence, in this sibling's grid world problem, the second Q-learning problem is solved implicitly and only once before the iteration for the multi-armed bandit problem begins.

4. Experiment

The simulation experiments we run in Gymnasium supports the theoretical computation performed in Section 3.1. The Monte Carlo simulations that we perform can yield even more results, such as what is the expected number of "bad" moves that the robot makes before it reaches the goal. Here, a bad move means that the agent moves away from the goal. We can further compute a histogram of the states visited, and the rewards collected on average.

The first result in Figure 2 shows the running average number of steps taken to reach the goal over 50,000 independent Monte Carlo simulation runs. Figure 2 shows that the running average converges to $6.\bar{6}$ just as the theoretical computation indicated in Section 3.1.

Table 1 contains the probabilities of number of steps it took to solve the environment over 50,000 independent Monte Carlo simulation runs. These probabilities are calculated empirically by counting the number of times the agent takes 4, 6, and 8 steps to reach the goal and dividing by the total number of runs. We observe that it takes 4 steps to solve the environment $16.\bar{6}\%$ of the time, 6 steps $33.\bar{3}\%$ of the time, and 8 steps 50% of the time, yielding an expected value of

$$\frac{1}{6} \times 4 + \frac{1}{3} \times 6 + \frac{1}{2} \times 8 = \frac{20}{3} = 6.\bar{6}.$$

The shorter the path the agent takes to reach the goal, the more uncertain it tends to be about the identity of the true world. For instance, if it takes only 4 steps to reach the goal, the agent may only know that the world might be 1 of the 12 possible worlds, having eliminated the remaining 12. However, if it takes 8 steps to reach the goal, it will have eliminated 20 of the possible 24 worlds that it starts with and is certain that the true world is one of the remaining 4. If the agent takes 6 steps to solve the environment, then it may only have eliminated 14 of 24, still remaining with 10 possible worlds. Notice that this doesn't mean that the agent cannot solve the environment; indeed, it still can and has.

Figure 3 shows the frequency of visits of each state under the optimal policy deduced by Q-learning over 50,000 independent Monte Carlo runs. Notice that the ratio of the goal state being visited to the number of runs is 1, meaning that all runs end at the goal state. Further, we see that the states the robot passes through either $(4, 2)$ or $(2, 4)$ to reach the goal, and never through $(3, 3)$. The reason that this happens is that the Q- learning algorithm acts greedily with respect to its value estimates and when the agent reaches one of the states $(3, 2)$ or $(2, 3)$, it will always value the action it took to get there more than the others; hence continuing on to $(4, 2)$ or $(2, 4)$, respectively, instead of trying out the state $(3, 3)$.

Table 1: Histogram of the number of steps taken to solve the environment over 50,000 Monte Carlo simulation runs.

Probability	1/6	1/3	1/2
Steps	4	6	8

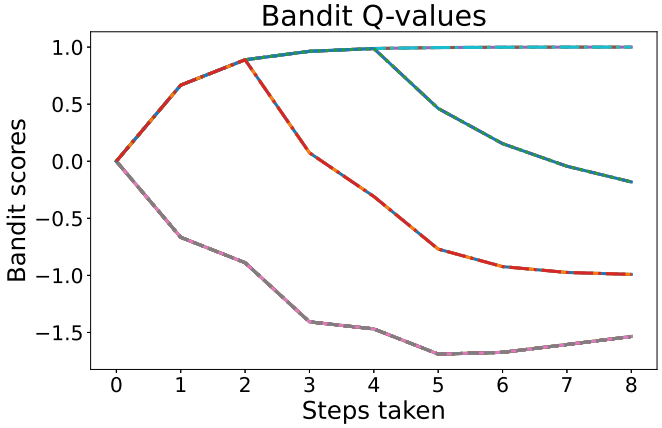

Figure 4: Smoothed scores each world belief achieves over a single run sample.

Figure 4 shows the scores each world belief achieves over simulation steps during a single run that lasts 8 steps. At the end there are four bandit arms (world beliefs) that have the same high score of 1, however, two of these bandits arms have been identified to swap the "down" and "right" actions. Since this swapping does not affect the performance of the agent, their score matches the other two top bandit arms which have not been differentiated.

5. Conclusions

We have defined a novel grid world type optimal decision-making problem that is meant to be solved by an RL agent in a one-shot manner. After thoroughly defining the problem, we solved it theoretically with classical probability theory and then provided its RL formulation before running Q-learning to solve it numerically.

The results of the RL solution is presented in Section 4. First observation was that the RL algorithm figures out the optimal policy in a single episode, which gives same expected number of steps to reach the goal as we find theoretically. This expected number is found by performing $50,000$ independent Monte Carlo simulation runs. Secondly, we have extracted additional statistics using these independent Monte Carlo runs. We have computed the distribution of the number steps it takes to reach the goal under the optimal policy, from which the expectation may be computed directly. We provided a distribution of the number of times all the states of the sibling grid world is visited. We also provided a plot of the scores of the bandit arms over the learning process (one-shot).

It is seen that one-shot RL may be performed in this simple, yet interesting environment by learning two Q-functions in parallel. In this note, we used the classical Q-learning algorithm. Other classical approaches such as double Q-learning, etc. may be attempted and compared against our results here.

References

Bertsekas, D.P., Tsitsiklis, J.N., 2002. Introduction to probability. volume 1. Athena Scientific Belmont, MA.

Luenberger, D., Ye, Y., 2008. Linear and Nonlinear Programming. International Series in Operations Research & Management Science, Springer US.

Morales, M., 2020. Grokking Deep Reinforcement Learning. Manning Publications.

Satici, A.C., 2024. Convoluted questions. URL: https://github.com/Symplectomorphism/miscellaneous/tree/master/dolambac.

Sutton, R., Barto, A., 2018. Reinforcement Learning, second edition: An Introduction. Adaptive Computation and Machine Learning series, MIT Press.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al., 2024. Gymnasium: A standard interface for reinforcement learning environments. arXiv preprint arXiv:2407.17032 .