Lab1 Reinforcement Learning

Merlijn Sevenhuijsen — merlijns@kth.se — 200104073275
Hugo Westergård — hugwes@kth.se — 200011289659

November 2023

1 Problem 1

1.1 1a) formulation as MDP

State space $\mathcal{S}$:
Within this model there are two objects that are continuously moving. Namely the player and the Minotaur. The player can move in any direction (up, down, left, right) or stay in the same position. The Minotaur can move in any direction (up, down, left, right). The state space is the set of all possible positions of the player and the Minotaur in the maze. However, there is one Moreover if the player is caught then it is impossible to get any future rewards, or move out of the current position.

$$\mathcal{S} = \Big\{((i,j),(k,l)) : \text{such that the cell}(i,j) \text{ is not an obstacle or (k, l) and (k,l)}$$
$$\text{is not a boundary}\Big\}|\text{caught}.$$

Action space $\mathcal{A}$
We cannot control the Minotaur, but we can control the player. Therefore we have five options: move 'left', 'right', 'down', 'up' and 'stay'. Just like lab0 we allow the player to move in all directions, and move into a "wall", but the state will not change and the reward will be a large negative number to prevent this from happening.

$$\mathcal{A} = \{\text{up, down, left, right, stay}\}.$$

Transition probabilities $\mathcal{P}$
At a state (or position) s taking any action will always lead to a different state (position or caught) s', then $P(s'|s,a) = \frac{1}{m}$. This is due to the Minotaur always being able to move and it not going into the walls. Note that it depends on the amount of possible moves for the Minotaur, called m. If the Minotaur is in the bottom right corner, then it can only move left and up. This leads to two possibilities for each action.

An exception to this is that it is possible that the Minotaur catches the player. In that case the next state will be caught, thus $p(caught|s,a) = \frac{1}{m}$ if the action a would lead to the position of the player, (i,j) being the same as the Minotaur (k,l), thus $i = k, j = l$.

If the player is caught then it will always go to the caught state *caught*, then for any action a $P(caught|caught,a) = 1$.

Rewards $\mathcal{R}$
The objective of the player is to find the exit of the maze while avoiding the obstacles and the Minotaur.

- If at state s, taking action a, leads to a wall or an obstacle then $r(s,a) = -\infty$
- If at state s, taking action a, leads to the Minotaur then $r(s,a) = -\infty$
- If at state *caught*, taking any action a lead to $r(caught,a) = -\infty$
- If at state s, taking action a, leads to some other position in the maze that is not the exit nor a wall nor the Minotaur nor an obstacle, then $r(s,a) = -1$.
- If at state s, taking action a, leads to the exit then $r(s,a) = 0$.

1.2 1b) Difference when the player and the Minotaur move one at a time

We start with the MDP definition of the situation where the player and the Minotaur move by alternating.

State space $\mathcal{S}$
Within this model there is only one object that can be moved, namely the player. The Minotaur moves as well but when the player takes an action this does not move the Minotaur directly. The player can move in any direction (up, down, left, right) or stay in the same position. The state space is the set of all possible positions of the player. Here the assumption was made that the player only moves and that each round the Minotaur is seen as an obstacle. This is done as the transitions now are certain and only move the player instead of the Minotaur as well. The Minotaur can be seen as a type of wall, which occupies five spaces. Namely the direct position and all the direct neighbours (by moving up, down, left and right one time). The only difference with the difference with moving simultaneously is that now the player cannot swap positions with the Minotaur, as this will end up directly on the Minotaur and give a large negative reward. This will elaborated upon later.

$$\mathcal{S} = \big\{(i,j) : \text{such that the cell } (i,j) \text{ is not an obstacle nor the Minotaur } \big\} | caught.$$

Action space $\mathcal{A}$
The five options remain the same, move 'left', 'right', 'down', 'up' and 'stay'. we allow the player to move in all directions, and move into a "wall", but the state will not change and the reward will be a large negative number to prevent this from happening.

$$\mathcal{A} = \{\text{up, down, left, right, stay}\}.$$

Transition probabilities $\mathcal{P}$
At a state (or position) s taking any action will now lead to a guaranteed next always lead to a next state as the Minotaur does not move with the action of the player anymore. Thus s', then $P(s'|s,a) = 1$.

The exception to this is that when the player moves close Minotaur we have to take into account the possibility of the Minotaur moving towards the player. If the player moves on the Minotaur using action a then the next state is caught: $P(caught|s,a)$. If the player moves towards a place where the Minotaur is one step away by any possible action, then $P(caught|s,a) = \frac{1}{m}$ to still account for the randomness of the Minotaur catching the player. Here m depends on the possible next actions of the Minotaur.

If the player is caught then it will always go to the caught state *caught*, then for any action a $P(caught|caught,a) = 1$.

Rewards $\mathcal{R}$
The objective of the player is to find the exit of the maze while avoiding the obstacles and the Minotaur.

- If at state s, taking action a, leads to a wall or an obstacle then $r(s, a) = -\infty$
- If at state s, taking action a, leads to the Minotaur then $r(s, a) = -\infty$
- If at state s, taking action a, leads to a place where the Minotaur can move in one time-step then $r(s, a) = -\infty$
- If the player is in state caught then any action a will lead to $r(caught, a) = -\infty$
- If at state s, taking action a, leads to some other position in the maze that is not a wall, obstacle, or Minotaur position, then $r(s, a) = -1$
- If at state s, taking action a, leads to the exit then $r(s, a) = 0$

The difference between moving simultaneously and non-simultaneously is that the player cannot be in the exact position of the Minotaur. When they move simultaneously the player can swap places with the Minotaur without a punishment of getting caught. However, when the player and the Minotaur move by alternating then this is not possible anymore. The move of the player will always end up with the Minotaur on the same position.

When the Minotaur moves after the player one position more is occupied, due to swapping positions with the Minotaur not being possible. This means that the Minotaur has a higher probability probability of catching you.

1.3 1c) find a policy that maximizes the probability of leaving the maze alive for T = 20 and Illustrate this

It is assumed that the MDP where both the player and the Minotaur move simultaneously is used. This was done as this exercise was given first and the assignment only asked to compare both models and not which one to use. For the code please look at the notebook *problem*$_1$*.py* and *problem*$_1$*.ipynb*. In the jupyter notebook the code is put that generated the following plot: The plot

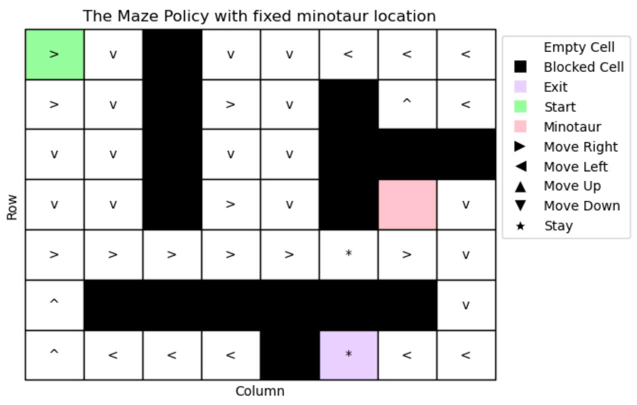

Figure 1: Maze policy with a fixed Minotaur location at (3, 6)

shows that the player walks towards the Minotaur until the point where it can kill the player, which is if the player takes one step right then there is a probability of getting caught. If the Minotaur was on another position then the following would have happened: Here the player is free to move towards the

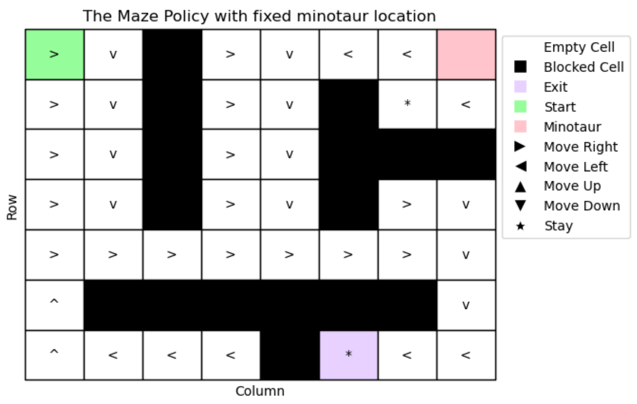

Figure 2: Maze policy with a fixed Minotaur location at (0, 7)

goal, a simulation can be seen in the Jupyter Notebook.

1.4 1d) For T = 1,...,30 compute a policy that maximizes the probability of exiting the maze alive (in the shortest time possible) and plot the probability. Is there a difference if the Minotaur is allowed to stand still? If so, why

There are two ways of interpreting the question. We answer both the ways, which leads to different results. The first way is that at each time-step (t) of an optimal policy for $T = 30$ we plot the probability of exiting at that time. This results to the following two graphs:

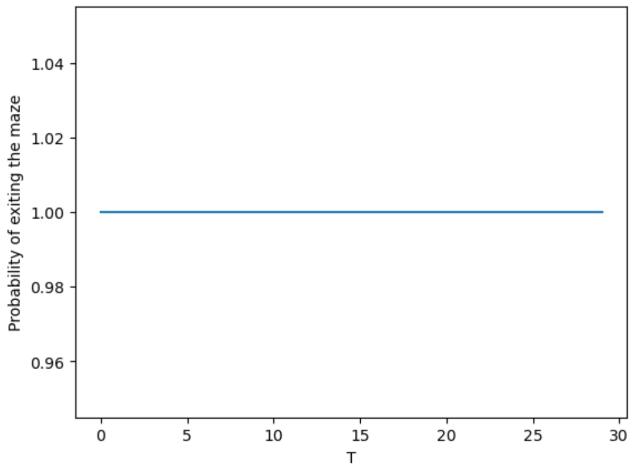

Figure 3: The probability plotted for each t from 1 to $T = 30$ using the optimal policy at $T = 30$ where the Minotaur cannot stand still

One can see that the probability of exiting for one is 1, this is due to the Minotaur always being on a tile that is not reachable for the user. Think of it as a chess board, where one starts at white and the other starts at black. If both players move one step at a time then they can never end up on the same tile as black goes to white and white goes to black. By allowing the Minotaur to stand still it is possible to end up on the same tile, therefore that probability is lower. We can see that if the Minotaur cannot move we get a probability of 1 of exiting, which is only ≈ 0.695.

The second way of interpreting is that we train an optimal policy for $T = 1$, $T = 2$, ..., $T = 30$. This results in the following graphs: We see that if the Minotaur cannot stand still that starting from $T = 16$ it is possible to reach the exit. If the Minotaur is able to catch the player (think again of the chess board where both start on black, which is for odd T), then it increases with more T. When the Minotaur is able to stand still it is not possible for the player to have a guaranteed exit. Here the probability of exiting increases with T. However, if

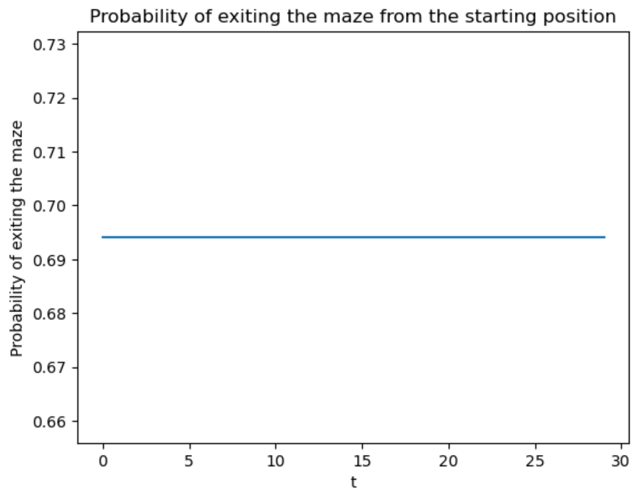

Figure 4: The probability plotted for each t from 1 to $T = 30$ using the optimal policy at $T = 30$ where the Minotaur can stand still

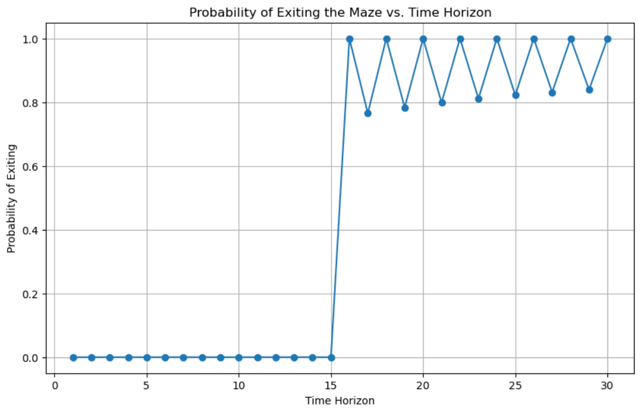

Figure 5: The probability plotted for an optimal policy trained from $T = 1$ to $T = 30$ where the Minotaur cannot stand still

the Minotaur is able to stand still the general probability of exiting without being killed by the Minotaur increases. This is due the chess board idea explained earlier. In short, there is a difference in probability.

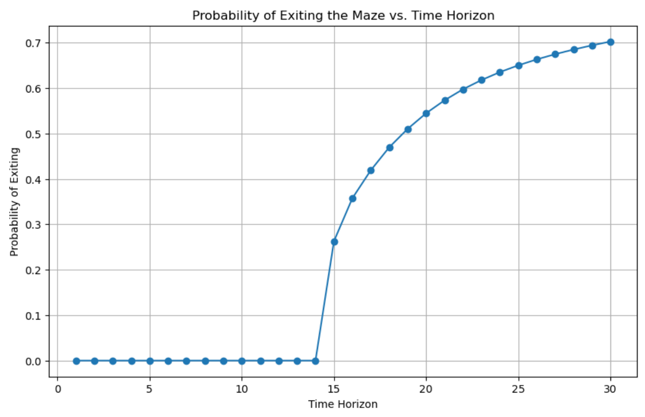

Figure 6: The probability plotted for an optimal policy trained from $T = 1$ to $T = 30$ where the Minotaur can stand still

1.5 1e) Geometrically distributed lifespan due to poison, motivate the changes.

We have a random time horizon, geometrically distributed with a mean of 30. From the slides we have: $E[T] = \frac{1}{1-\lambda}$. We have $E[T] = 30$, leading to $30 = \frac{1}{1-\lambda}$.

$$30 = \frac{1}{1-\lambda} \longrightarrow 30(1-\lambda) = 1 \longrightarrow 30-30\lambda = 1 \longrightarrow 30\lambda = 29 \longrightarrow \lambda = \frac{29}{30} \approx 0.967$$

Now we use a discounted reward:

$$max_{\pi} E[\sum_{t=1}^{\infty} \lambda^{t-1} r_t(s_t^{\pi}, a_t^{\pi})]$$

The remainder of the formulation stays the same, this way the short lifespan of the player is also taken into account.

1.6 1f) Simulate 10000 games and get the probability of exiting

10000 Simulations are ran to get the probability of exiting. We look at if the Minotaur is allowed to stand still and the case that the Minotaur cannot stand still. If the Minotaur cannot stand still the probability of exiting is yet again 1, therefore it is more interesting to look at the case where the Minotaur is allowed to move. Here the probability of exiting alive is $\frac{4917}{10000} \approx 0.492$. The more games we simulate the closer we get to the real probability of exiting.

1.7 1g) difference of on-policy and off-policy learning

1) What does it mean that a learning method is on-policy or off-policy?

On-Policy Learning:
On-policy learning methods learn the value of the policy being followed, meaning they evaluate or improve the policy that is used to make decisions. The key characteristic of on-policy methods is that they attempt to evaluate or improve the policy that dictates the agent's actions. An example of an on-policy algorithm is SARSA (State-Action-Reward-State-Action), where the learning process follows the policy the agent is actually executing, and updates are made based on the actions taken by the current policy.

Off-Policy Learning:
Off-policy methods learn the value of an optimal policy independently of the agent's actions. They evaluate or improve a policy different from the one used to generate the data. This means that the learning algorithm can learn from actions that are outside its current policy, such as from a fixed dataset of experiences, or from the actions chosen by a different policy. An example of an off-policy method is Q-learning, where the learning process updates its estimates based on the maximum reward of the next state, regardless of the action taken by the current policy.

2) State the convergence conditions for Q-learning and SARSA.
Q-learning is guaranteed to converge to the optimal action-value function Q*(s,a), provided the following conditions are met:

- The policy has a chance to visit all state-action pairs infinitely often. This is often ensured through adequate exploration, such as using an ϵ-greedy policy that has a non-zero probability of selecting any action.
- The learning rate α follows certain conditions, typically it must decrease over time and satisfy the criteria

$$\sum_{t=1}^{\infty} \alpha_t = \infty \text{AND} \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

  These conditions ensure that the learning rate is high enough to continue learning indefinitely but decays in such a way that the value function estimates eventually converge.
- The environment and the reward function must be stationary, meaning they do not change over time.

SARSA also converges to the optimal action-value function under similar conditions as Q-learning:

- All state-action pairs must be visited infinitely often, which is usually achieved through sufficient exploration (e.g., ϵ-greedy approach).
- The learning rate must satisfy the same criteria as for Q-learning, ensuring it decays appropriately over time.
- The environment and reward function should be stationary.
- Additionally, the policy being followed must also converge to a policy that is greedy with respect to the Q-value estimates. This is because SARSA's updates depend on the policy followed (as it's an on-policy method), and for convergence, this policy must stabilize.

1.8 1h)

Altering the State Space: Key Collection: The state must reflect whether the keys have been collected. This can be a binary part of the state (e.g., keys collected or not). Now the state space will consist of the location of the user, the location of the Minotaur and the key.

Adjusting the Transition Probabilities: **Player's Movement**: The transition probabilities for the player's movement remain largely the same, except for the key and the transition probabilities of the Minotaur. All the transitions to the tile C will transition to a state where the user also has a key.
Minotaur's Movement: The Minotaur's movement probabilities are modified so that there's a 35% chance it moves toward the player and a 65% chance it moves randomly in any direction. This means that there is a 51.25% chance to move towards the player and a 16.25% chance to move in every other direction in the case that all four directions are possible for the Minotaur. This is due to the random action also having a possibility of going towards the player.

Modifying the Reward Structure:
Key Collection Reward: Introduce a reward for collecting the keys, encouraging the policy to direct the player towards them. This reward will be 0, thus $R(s, a) = 0$ if action a in state s leads to the state $(0, 7)$.
The reward for reaching the exit now only gives a reward if the key is obtained. Otherwise it will just give a -1 as usual.