LAB 1 - EL2805

Mustafa Al-Janabi
970101-5035
musaj@kth.se

Mikel Zhobro
950216-T590
zhobro@kth.se

Problem 1: The Maze and the Random Minotaur

a) MDP formulation

State space

We define a state as a tuple (i,j,k,l) where the two first parameters describe the position of the player in the grid and the last two that of the minotaur. While the position of the player is restricted only on free grid spaces, the position of the minotaur can be everywhere on the grid map. In addition to that we define two terminal states which correspond to the state where the player is eaten by the minotaur and the state where the player has exited the maze successfully. The formal definition of the state space is given below.

$$\mathcal{S} = \Big\{(i,j,k,l) \ : (i,j) \text{ is neither obstacle nor goal and } (i,j) \neq (k,l)\Big\} \cup \Big\{Eaten\Big\} \cup \Big\{Exit\Big\} \quad (1)$$

Action space $\mathcal{A}$

The action space is defined as

$$\mathcal{A} = \{stay, up, down, left, right\} \quad (2)$$

Transition probabilities $\mathcal{P}$

Two different models have been studied with respect to the transition probabilities.

Model 1

In the first model the following set up is used for the transition probabilities.

- Terminal state $s = Eaten : P(s|s,a) = 1, P(s'|s,a) = 0$ if $s' \neq s, \forall a$.
- Terminal state $s = Exit : P(s|s,a) = 1, P(s'|s,a) = 0$ if $s' \neq s, \forall a$.
- Player moves to empty cell $P(s'|s,a) = 1$
- Player moves to impossible cell $P(s'|s,a) = 0$

Model 2

Since we did consider the random movement of the monotaur in the transition probabilities for this model, there is no need to penalize the positions near the minotaur as before.

- Terminal state $s = Eaten : P(s|s,a) = 1, P(s'|s,a) = 0,$ if $s' \neq s, \forall a$
- Terminal state $s = Exit : P(s|s,a) = 1, P(s'|s,a) = 0,$ if $s' \neq s, \forall a$
- Player takes action a, where next state, s' is not an obstacle:
  $P(s'|s,a) = \frac{1}{\text{\#actions by minotaur}} \forall s'$
- Player cannot move to obstacle $P(s'|s,a) = 0$ and must stay in place $P(s|s,a) = 1$

- If next state s' is an obstacle then the player stays in the same cell, ie. $P(s'|s,a) = 0$ and $P(s|s,a) = 1$

This change takes into account the fact that the minotaur makes a uniformly random move for every action.

Rewards $\mathcal{R}$

Model 1

Since we didn't consider the random movements of the minotaur in the translation probabilities, this model solved that by giving a negative reward to the states where the player is located at a state which is immediately reachable by the minotaur. That is summarized by the following:

- 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 $r(s,a) = -80$
- If at state s, taking action a leads to a state where the player is at a state that is immediately reachable by the minotaur $r(s,a) = \frac{-80}{\text{\#actions by minotaur}}$
- If at state s and previous state was with the minotaur, $r(s,a) = -80$
- If at state s, taking action a leads to the exit then $r(s,a) = 0$
- If at state s and previous state was the minotaur, $r(s,a) = 0$
- If at state s, taking action a leads to neither of the aforementioned states $r(s,a) = -1$

Model 2

For this model we don't assume any different any reward for being in the moving range of the minotaur since the random movement of the minotaur is considered in the transition probabilities.

- 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 $r(s,a) = -80$
- If at state s and previous state was with the minotaur, $r(s,a) = -80$
- If at state s, taking action a leads to the exit then $r(s,a) = 0$
- If at state s, taking action a leads to neither of the aforementioned states $r(s,a) = -1$

b) Solving the problem with finite horizon

Firstly, we present an illustration of an optimal policy with time horizon $T = 20$ in Figure (1)

Secondly, we now study the probability of existing as a function of time, T. As discussed earlier, we denote the model where the minotaur is modeled in reward as **Model 1** and the model where the minotaur is described in the transition probabilities as **Model 2**. Furthermore we try two different scenarios, one where the minotaur only take the actions: $up, left, down, right$. And another scenario where the minotaur can also take the action $stay$ on top of $up, down, right, left$. The result is presented in Figure (2).

The result shows that, with respect to the probability of existing, the difference between allowing the minotaur to stay or not is small. However, Model 2, where the minotaur is modelled in the probabilities, performs better than Model 1 where the minotaur is modelled in the reward. This can be explained by the fact that modelling the minotaur in the reward (model 1) makes the player avoid getting near the minotaur as much as possible to avoid a negative reward, thus it takes the player a longer time to achieve the goal and thus it requires longer time horizon. In model 2 the player is more aggressive and takes the chance of getting near the minotaur.

When studying probability of getting eaten, it is clear that whether the minotaur can stay or not does indeed make a difference. We still find that the more aggressive Model 2 does perform better in both cases. For the same reasons mentioned above. Furthermore, we find that the probability of being eaten is lower if we do not allow the minotaur to stay. This can be explained by the fact that the uncertainty in the minotaur's movement is lower since it ends up in a certain state with a

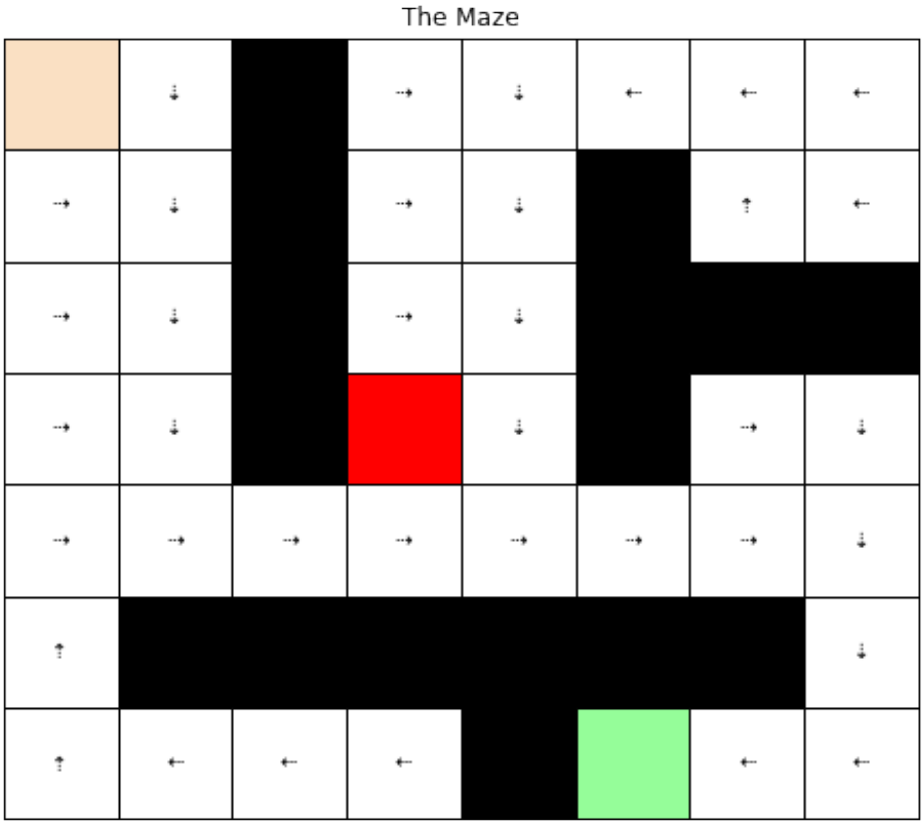


Figure 1: The optimal policy shown as arrows given the player is in state $s = (i, j) = (0, 0)$ and the minotaur at $s = (k, l) = (3, 3)$ at time step t=0 with T=50.

probability of $\frac{1}{4}$ as opposed to the lower certainty in position $\frac{1}{5}$, when we allow the minotaur to stay, that increases the chances of avoiding the minotaur successfully and, thus, have lower probability of getting eaten.

From Figure 2 we also find that the optimal time horizon is around 60, since all models converge to the same probability of exiting, if T=60.

c) Solving the problem using value iteration

Assuming that our life is geometrically distributed with mean 30 based on

$$\mathbb{E}(X) = \frac{1}{1-\lambda} \tag{3}$$

We come to the conclusion that for this case $\lambda = \frac{29}{30}$ The objective of this problem is to learn the policy that ensures the player reaches the goal in as few steps as possible. This is done by modifying the reward according to the following new formulation

- 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 $r(s, a) = -80$
- If at state s and s is goal $r(s, a) = 0$
- If at state s and is at the same cell as the minotaur $r(s, a) = -80$
- If at state s and previous state was with the minotaur, $r(s, a) = 0$
- If at state s, taking action a leads to the exit then $r(s, a) = 0$

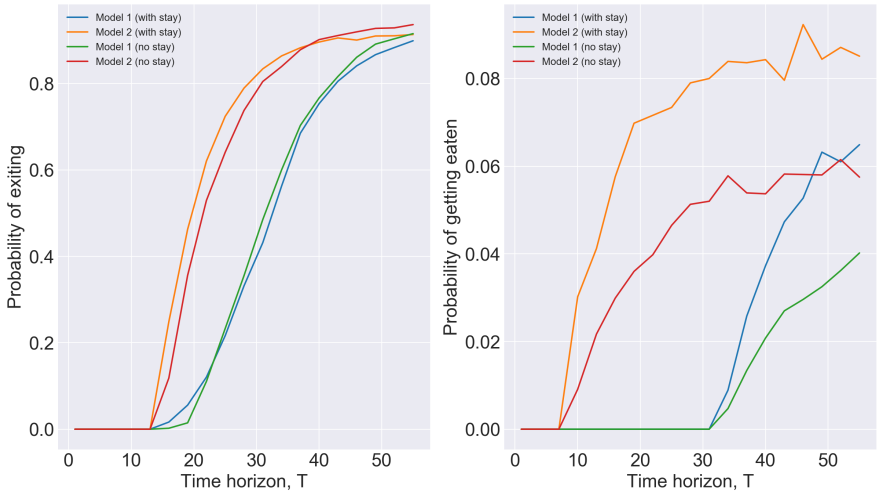


Figure 2: The probability of exiting the maze on the left and the probability of getting eaten on the right. For the two different models and scenarios for minotaur movememnt.

- If at state s, taking action a leads to neither of the aforementioned states $r(s, a) = -1$

Table 1: The expected time to reach the goal given the two different reward set ups

Reward	Expected time
Old reward model	27.6
New reward model	14.7

With this model since reaching the minotaur is only penalized for the first time it happens the player is less risk averse and, thus gets to the goal quicker.

Problem 2: Robbing Banks

a) MDP formulation

State space

We define a state as a tuple (i,j,k,l) where the first two parameters describe the position of the player in the grid and the last two that of the police. Any position outside the grid is called an $impossible\ state$. The formal definition of the state space is given below.

$$\mathcal{S} = \Big\{(i, j, k, l) \ : (i, j, k, l) \text{ is not an impossible state}\Big\} \tag{4}$$

Action space $\mathcal{A}$

The action space is defined as

$$\mathcal{A} = \{stay, up, down, left, right\} \tag{5}$$

where each action corresponds to the movement that the player takes.

Transition probabilities $\mathcal{P}$

The transition probabilities for the model are formulated as the following

- $\mathbb{P}(0,0,1,2|i,j,k,l,a) = 1$, if $i = k$ and $j = l, \forall a$. When the player is caught by the police, the game restarts regardless of what action the player takes.
- $\mathbb{P}(s'|s,a) = 0$ if s' is an impossible state . When at state s, the player cannot take a given action a and end up in an impossible state, instead it stays in the same place.
- $\mathbb{P}(s'|s,a) = \frac{1}{\#\text{actions by police}}$, for s' all possible states that the game can end up in after player takes action a and the police move according to the description in the exercise which depends on the relation between the position of the police and the player. Where *# actions by police* is 2, if after taking action a the player ends up in a state where $i \neq k$ and $j \neq l$. Otherwise if the player ends up in a state where $i = k$ or $j = l$, then *actions by police*$= 3$.
- $\mathbb{P}(s'|s,a) = 0$, otherwise.

Rewards $\mathcal{R}$

The rewards for the model are defined as follows

- If at state s, taking action a leads to an impossible state, then $r(s,a) = -\infty$
- If at state s, taking action a leads to a bank, then $r(s,a) = 10$
- If at state s, and taking action a leads to getting caught (moving into the same cell as police), then $r(s,a) = -80$
- If at state s and $i = k$ and $j = l$, i.e. the police has moved into our cell, then $r(s,a) = -80$
- If at state s, taking action a leads to neither of the aforementioned states $r(s,a) = 0$

b) The value function

The problem was solved using value iteration. In this part we study the value function as a function of the discount factor $\lambda \in (0,1)$, at the initial state $s = (0,0,1,2)$, as presented in Figure 3. Furthermore, we study the policy for different values for the discount values at states $s = (i,j,0,0)$.

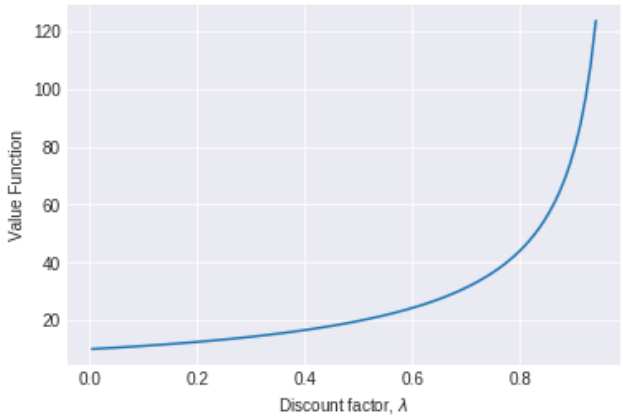
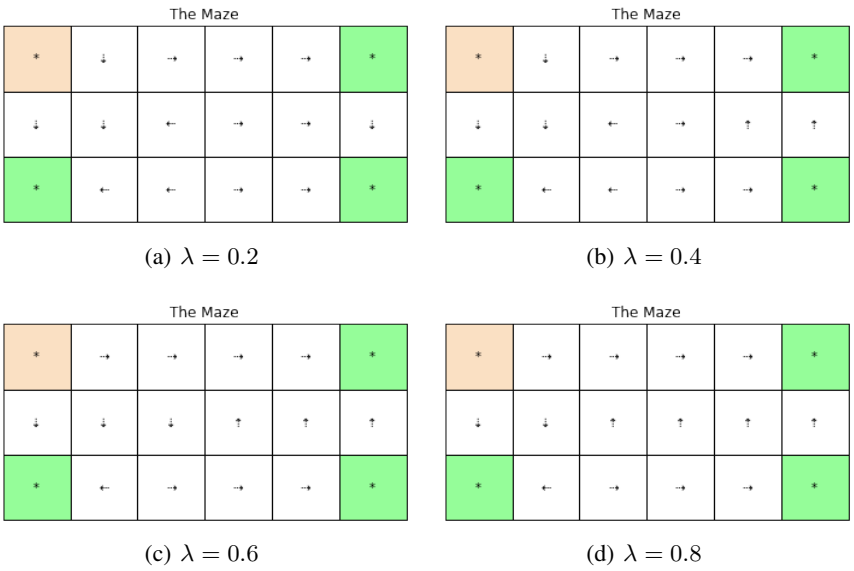


Figure 3: The value function at the initial state $s = (0,0,1,2)$ as a function of λ

As can clearly be seen in 3 the value function for the initial state increases with lambda. This can be explained by the fact that the lower λ is the less we consider the rewards of future steps, and since the player never gets caught and most of the time stays at a different bank, it corresponds to a high positive value. On the other hand, if $\lambda \approx 0$ the value function is approximately as big as the reward gained for staying at the inital state which coincides with one of the a bank, $V^*(s) \approx r(s,a)$. Thus, the rewards is as big as the one gained for staying at the bank and no more, since the value function, then, doesn't consider any future states due to the small discount factor.

In Figure 4 we observe that for lower discount factors the optimal policy tells the player to move towards the bank that is closest to it. However, increasing the discount factor allows the player to look further into the future which explains why in the Figures the player tends towards banks that



(a) $\lambda = 0.2$ (b) $\lambda = 0.4$

(c) $\lambda = 0.6$ (d) $\lambda = 0.8$

Figure 4: The policy for state state$s = (i, j, 0, 0)$ given different values for the discount factor λ

aren't necessarily closer. On the other hand, given a larger long term reward, since the player than is also aware of the fact that the police will take longer to catch up to him/her, leads to the player preferring banks further away from the police.

Problem 3: Bank Robbing (Reloaded)

a) Q-learning

In this part we study the convergence of the Q-learning algorithm, given the problem in the exercise, at the initial state $s = (0, 0, 3, 3)$ We use the same action space as in Problem 2, and the sampled rewards for every action are sampled in a similar manner too, but with slightly different values.

Reward $\mathcal{R}$

- If at state s, taking action a leads to an impossible state, then $r(s, a) = -\infty$
- If at state s, taking action a leads to a bank, then $r(s, a) = 1$
- If at state s, taking action a and letting the police take a random move leads to getting caught then $r(s, a) = -10$
- If at state s and $i = k$ and $j = l$, i.e. the police has moved into our cell, then $r(s, a) = -10$
- If at state s, after taking action a and letting police move randomly leads to neither of the aforementioned states $r(s, a) = 0$

We find in figure 5 that the Q-learning algorithm converges to a value of $V(s) \approx 37$ after 1.5 million iterations.

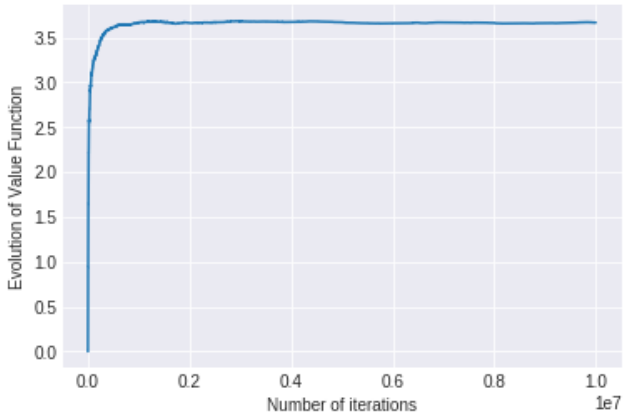
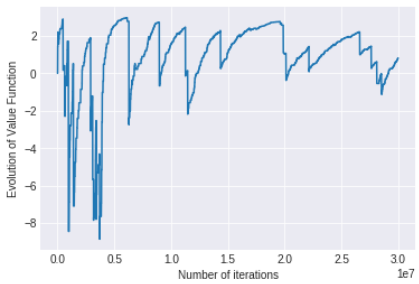


Figure 5: The evolution of the value function at the initial state $s = (0, 0, 3, 3)$ for Q-learning

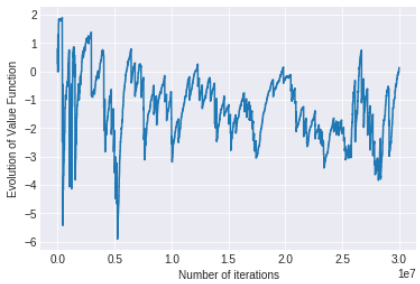
b) SARSA

For SARSA, the reward function is exactly the same as Q-learning. The only difference now is that we do SARSA in an ϵ-greedy manner. We take an exploring action with probability ϵ and with probability $1 - \epsilon$ we perform the action suggested by the most recent policy. In Figure 6 we can see the obtained result, when studying different values for ϵ.

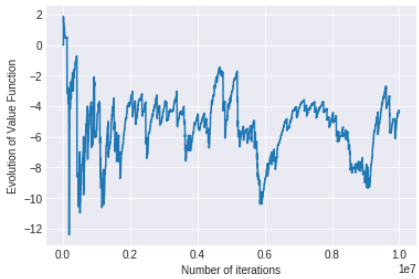
As seen in Figure 6 the Value Function converges to lower values with increasing ϵ. While it is lower than the values, the Q-learning converged to, under no means does it mean that the agent has not learned. For example for the case where $\epsilon = 0.1$ both algorithms converge towards the same policy. The Value Function presented in Figure 6 depends heavily on the random actions that are taken once in a while through the ϵ-greedy policy, independent of the base policy. This can be seen in the fluctuations of the Value Function which are of higher frequency the bigger ϵ gets. The often we explore the often we are prone to mistakes and the smaller the value we tend to converge to is. In addition to that we concluded that one cannot derive convergence by only observing the Value Function evolution of the ϵ-greedy policy. For that we would need to decay $epsilon$ or use some other action choice mechanism that reduces exploration over time.



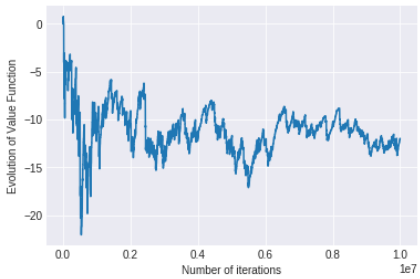
(a) $\epsilon = 0.1$



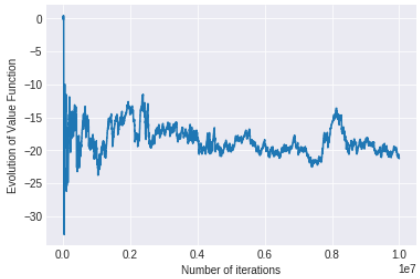
(b) $\epsilon = 0.2$



(c) $\epsilon = 0.3$



(d) $\epsilon = 0.4$



(e) $\epsilon = 0.5$

Figure 6: The evolution of the value function at the initial state for different values of ϵ using SARSA with 20 000 000 iterations.

In conclusion we can say that Q-learning's big strength is directly learning the optimal policy, but it does so at the expense of being more likely to make certain kinds of error, whereas SARSA's strength is that it will better optimise the values seen whilst learning, but it does this at the expense of not learning the optimal policy directly.