EL2805 Computer Lab 1: *The Great Escape*

Deepika Anantha Padmanaban
19950303-4408
deap@kth.se

Styliani Katsarou
900331-7006
stykat@kth.se

Abstract

1 Problem 1: The Maze and the Random Minotaur

1.1 Formulation of the problem as an MDP

State Space: The state space comprises of feasible combinations of the players's position (X_P, Y_P) and the minotaur's position (X_M, Y_M) in the maze , that is, the corresponding row and column values of the maze block they occupy, such that they are both in the maze and the player is not on a wall inside the maze. So it includes all possible quadruples $\{S_i = (X_P, Y_P, X_M, Y_M)\}$ where

$$\{(X_P, Y_P, X_M, Y_M) \in \mathbb{N} : 0 \leq (Y_P, Y_M) \leq 7 \wedge 0 \leq (X_P, X_M) \leq 6 \wedge (X_P, Y_P) \neq (X_{wall}, Y_{Wall})\}.$$

Actions: At each time-step the player's available moves are {"stay","move left","move right","move up","move down"} and the for the minotaur they are {"move left","move right","move up","move down" }.

Time Horizon and Objective: Finite horizon $T = 20$, and the objective function is $\mathbb{E}\left\{\sum_{t=0}^{T-1} r_t\left(s_t, a_t\right) + r_N\left(s_N\right)\right\}$, which is to be maximised.

Rewards: For every action that leads to the player or minotaur surpassing the limits of the maze, a penalization reward is given so that this state never occurs:

$r\left(s_{(X_P,Y_P,X_M,Y_M)}, \cdot\right) = -100$ if $(X_M, Y_M) = (X_{limit}, Y_{limit})$ or $(X_P, Y_P) = (X_{limit}, Y_{limit})$

where:

$$(X_{limit}, Y_{limit}) = (7, 8)$$

Any action that results to the player exiting the maze is positively rewarded to denote the optimal goal:

$$r\left(s_{(X_P,Y_P,X_M,Y_M)}, \cdot\right) = 0 \text{ if } (X_M, Y_M) \neq (X_{exit}, Y_{exit}) \text{ and } (X_P, Y_P) = (X_{exit}, Y_{exit})$$

If the player lands in a block which is not the exit nor occupied by the minotaur, the corresponding action is rewarded by -1 so that the sub-goal of an optimal path is fulfilled:

$$r\left(s_{(X_P,Y_P,X_M,Y_M)}, \cdot\right) = -1 \text{ if } (X_M, Y_M) \neq (X_P, Y_P) \text{ and } (X_P, Y_P) = (X_{goal}, Y_{goal})$$

where:

$$(X_{goal}, Y_{goal}) = (6, 5)$$

Every action that causes the agent to be caught by the minotaur is penalized by a negative reward:

$$r\left(s_t, \cdot\right) = -5 \text{ if } (X_M, Y_M) = (X_P, Y_P)$$

Transition Probabilities: Player's actions are deterministic, while the minotaur's actions are random and hence,does not have a fixed transition probability matrix.

Depending on where the minotaur is standing on the maze, the number of possible moves n it can make varies between 3 and 5, since it can only make 3 moves when sitting in a corner, 4 when touching a maze limit that is not a corner, and 5 in the remaining situations: $p_t\left(s_{t+1}|s_t, a=\cdot\right)=1/n$

The transition probability of the player was coded as a 3 dimensional matrix of shape (SxSxA), where a value of 1 in the cell was considered as a possible transition.

1.2 Finding the optimal policy for a finite time horizon

A simulation of our solution of an optimal policy is provided in the following link: `https://youtu.be/uXKVlPcL5Oc`.

The optimal escape policy includes a path of 16 blocks. Thereby, as depicted in Fig. 1 the maximal probability of exiting the game prior to the 15th time step would be zero, indicating that it is practically infeasible to exit the game when not enough time is provided for the agent to manage to get to the exit block. After the 15th step maximal probability is always 1, meaning that the agent can always find the way to the exit without being eaten. When the minotaur can also stand still the agent can follow a path which prioritizes not being eaten over a fast exit, and might end up running out of time. When there is enough time provided, the agent can safely exit the maze and the maximal probability of winning is again equal to 1 as can be seen in Fig. 2.

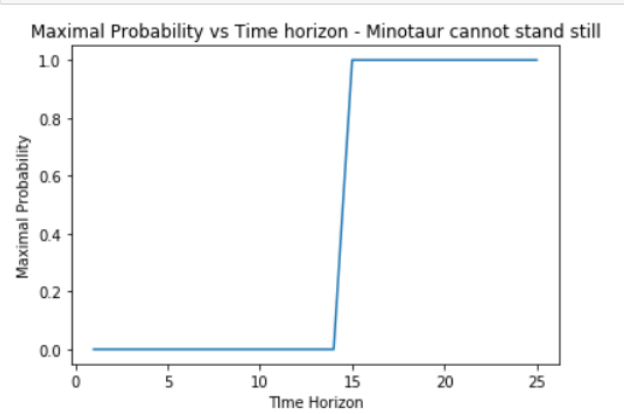

Figure 1: Maximal probability of exiting the maze over T, when the minotaur cannot stand still.

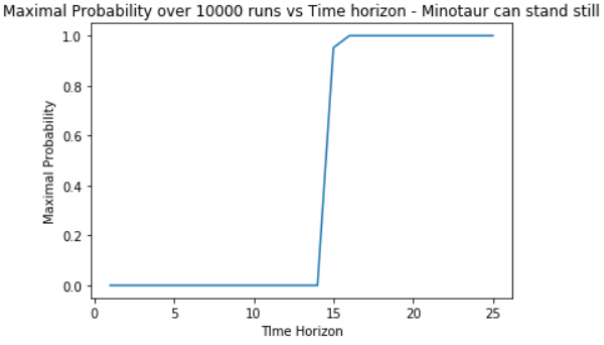

Figure 2: Maximal probability of exiting the maze over T, when the minotaur can stand still.

1.3 Finding the optimal policy when agent's life is geometrically distributed with mean 30

Changes in the formulation of the problem: The time horizon is now a discounted infinite one $\mathbb{E}\left\{\sum_{t=0}^{\infty}\lambda^t r_t\left(s_t, a_t\right)\right\}$ where λ is the discount factor which can be computed by $\frac{1}{1-\lambda}=30$ since

the agent's life is geometrically distributed with mean 30. We conduct 10 000 experiments and find the probability of getting out alive equal to 0.73.

2 Problem 3: Bank Robbing (Reloaded)

The possible states of the game were based on the robber's position (X_R, Y_R) and the police's position (X_P, Y_P) in the maze , that is, the corresponding row and column values of the maze block they occupy, such that they are both in the maze and the robber is not on a wall inside the maze. So the state space includes all possible quadruples $\{S_i = (X_R, Y_R, X_P, Y_P)\}$ where

$\{(X_R, Y_R, X_P, Y_P) \in \mathbb{N} : 0 \leq (Y_R, Y_P) \leq 3 \wedge 0 \leq (X_R, X_P) \leq 3\}$.

The state-action value for the robber going to the same cell as that of the bank is +1 ,i.e.,

$$\{r(S_i = (X_R, Y_R, X_P, Y_P), :) = 1\}$$

where $(X_R, Y_R) = (X_B, Y_B)$ and $(X_B, Y_B) = (1, 1)$ while that of going to the same cell as that of the police is -10, i.e.

$$\{r(S_i = (X_R, Y_R, X_P, Y_P), :) = -10\}$$

where $(X_R, Y_R) = (X_P, Y_P)$.

2.1 Q-Learning

In this part of the problem, Q learning is implemented. This algorithm learns the optimal value function by using some policy π_b which is a behavior design determined by the experimenter. In our case, the behavior design was to explore actions uniformly at random so that every state-action pair is explored and visited infinitely often. The update equation used was:

$$Q(s, a) = Q(s, a) + \alpha \left(R(s, a) + \gamma \max_{a'} Q\left(s', a'\right) - Q(s, a) \right)$$

The convergence of the algorithm can be seen in Fig. 3.

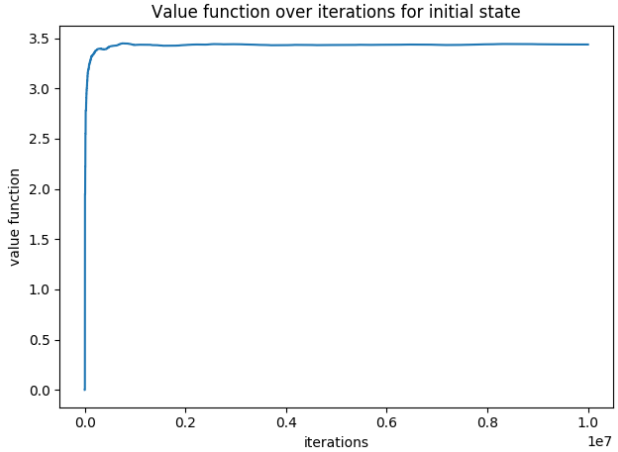

Figure 3: Value function of the initial state over time showing the convergence of Q learning algorithm

2.2 SARSA

Implementing Sarsa algorithm for ϵ values varying between 0.1 and 0.5 is shown in Fig.4. Sarsa learns Q function corresponding to a chosen policy π. In our case ϵ-greedy wrt Q_t updates was used, which means that the greedy action was chosen with probability $(1 - \epsilon)\%$. This practically means that the probability that SARSA = Q learning, is $\epsilon\%$. Consequently, by increasing the value of ϵ, the plot better approximates the optimal value function, and Fig.4 is an illustration of this conclusion. Implementation of SARSA for $\epsilon = 0.1$ may need more iterations to converge due to the

lower probability of random exploration in the policy, but better approximates the value function after convergence. The value function learnt by SARSA is not the optimal one, but a perturbation of the optimal Q function. The update equation used was:

$$Q(s,a) = Q(s,a) + \alpha\left(R(s,a) + \gamma Q\left(s',a'\right) - Q(s,a)\right)$$

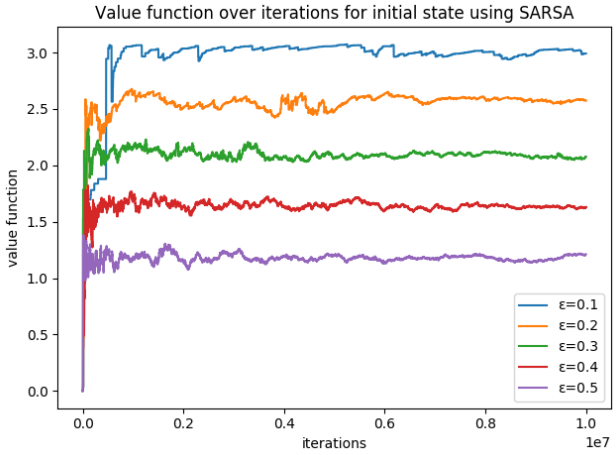

Figure 4: Value function of the initial state over time showing the convergence of SARSA algorithm

For ϵ values that are less than 0.1, SARSA algorithm always converges, as shown in Fig.5. The value functions are all converging towards a value of approximately 3, except for when ϵ gets really close to zero, which practically leads to very few random actions, so the algorithm is almost mimicking Q-learning, where the next action is greedy and does not follow the policy chosen.

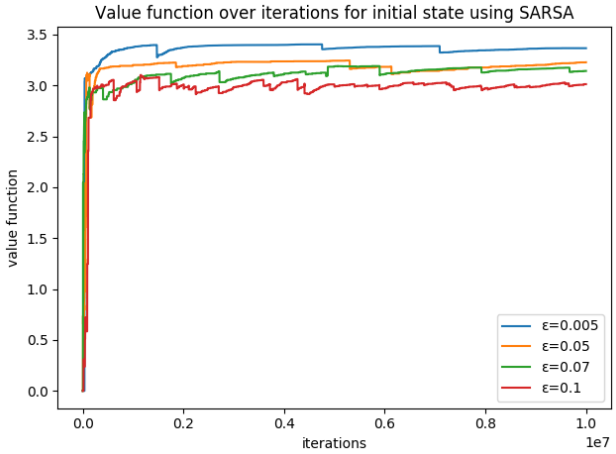

Figure 5: Value function of the initial state over time showing the convergence of SARSA algorithm for values of $\epsilon \leq 0.1$