EL2805 Reinforcement Learning
Computer Lab 1

Neil Pradhan
npradhan@kth.se
Personal no.: 19960704-5359

Rohit Kini
rrkini@kth.se
Personal no.: 19960903-8519

1 Problem: Agent Vs. Minotaur

1.1 MDP

We define MDP as {T, S, $A_s, p_t(*|s,a), r(s,a), 1 \leq t \leq T, s\epsilon S, a\epsilon A_s$ }. Where T is the time-horizon, S is the sates set, A is the set of actions, $p_t(*|s,a)$ is the transition probabilities, $r(s,a)$ is the rewards for action state pair.

- **States[S]**: We define states as $[(a\_row, a\_col), (m\_row, m\_col)]$ where the first tuple corresponds to the position of agent and the second one is the position of Minotaur.For the agent, states where there are no obstacles are considered where as for the Minotaur we take the every block in the matrix as a valid state. Therefore the Entire state space will have = 2240 states in total

$$S = \{S_0, S_1, S_2 ... S_{2239}\} \bigcup \{Eaten\} \bigcup \{Goal\}$$

$$S_i = (P_x, P_Y, M_x, M_Y)$$
$$0 \leq P_x \leq 7, 0 \leq P_y \leq 8; 0 \leq M_x \leq 7, 0 \leq M_y \leq 8$$

- **Actions[A]**: The action space for Agent is {UP, Down, Left, Right, Stay} and the action space for Minotaur is {Up, Down, Left, Right}.
- **Rewards**: we define the rewards as following:
  Terminal Rewards:

Rewards	Values1
r(S=Eaten, a=*)	-1
r(S=Goal, a=*)	1
r(Step, a=*)	0
r(Wall, a=*)	0

- Time Horizon and Objective-Function:
  Since the objective of the problem is to maximize the probability of the agent reaching the goal alive,the objective function can be formulated as following:

$$\mathbb{E}\Big[\sum_{t=0}^{T=20} r(s_t, a_t)\Big] = P[Goal]$$

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

- **Transition Probabilities:** The possible transition probabilities are:

$$P_t(S = Eaten|S = Eaten, all\_actions) = 1$$

$$P_t(S = Goal|S = Goal, all\_actions) = 1$$

$$P_t(S' = Eaten|S = S_{Eaten}, all\_actions) = 1$$

$$P_t(S' = Win|S = S_{Goal}, all\_actions) = 1$$

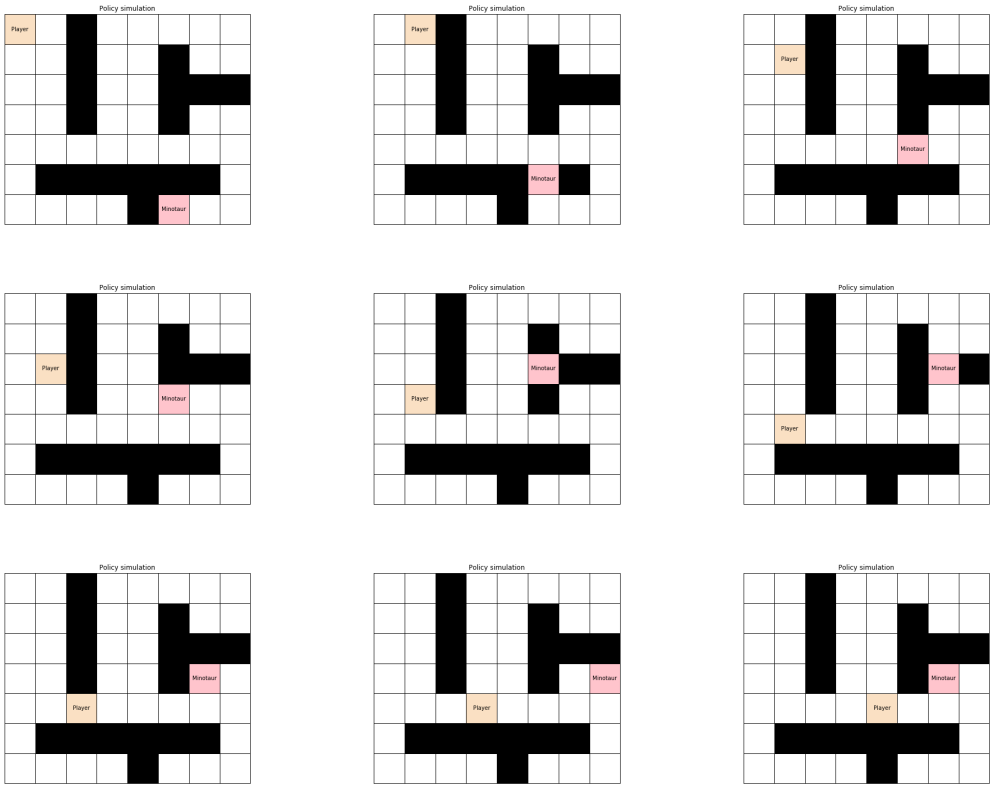
$$P_t(S' = S'_N|S = S_N, a = Minotaur\_actions) = 1/Minotaur\_actions$$

$$P_t(S(agent) = obstacles|S, all\_actions) = 0$$

The number of valid Minotaur actions are inconsistent at the corners and at the boundaries. Which are 2 and 1 respectively. At rest of the blocks, the Minotaur can perform 4 valid actions or 5 depending on if it is allowed the action STAY.

1.2 Policy Evaluation

The optimal policy is obtained by Dynamic Programming method and the simulation is as follows:



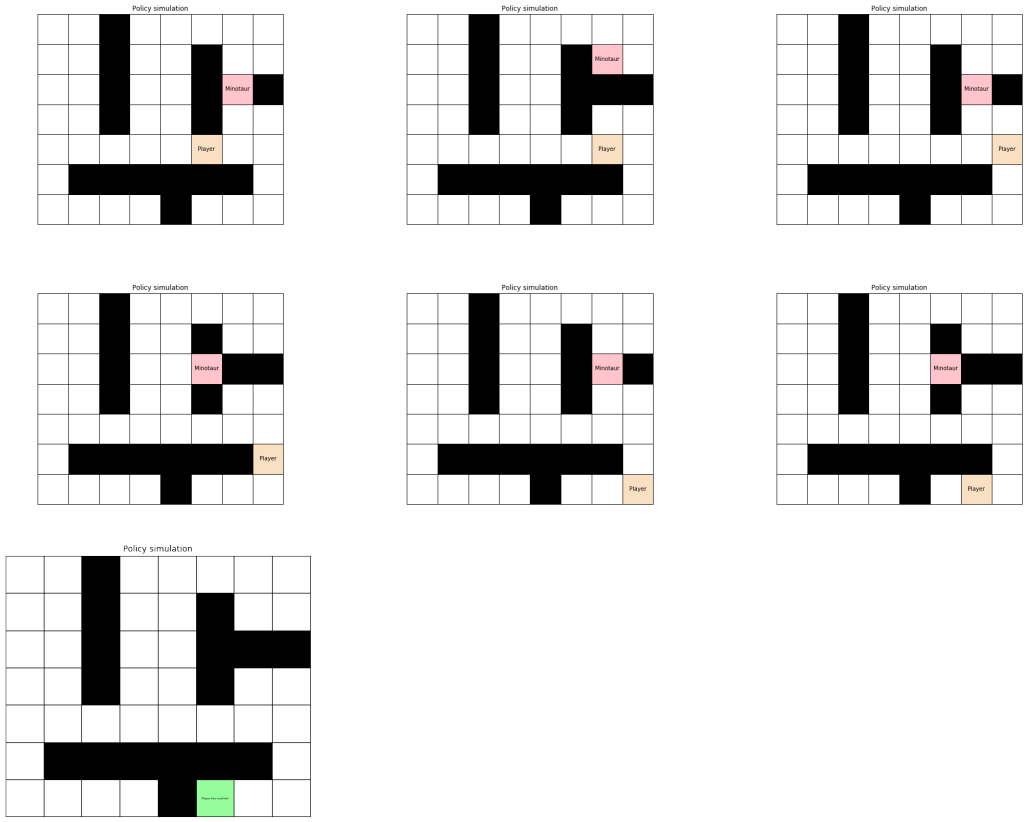
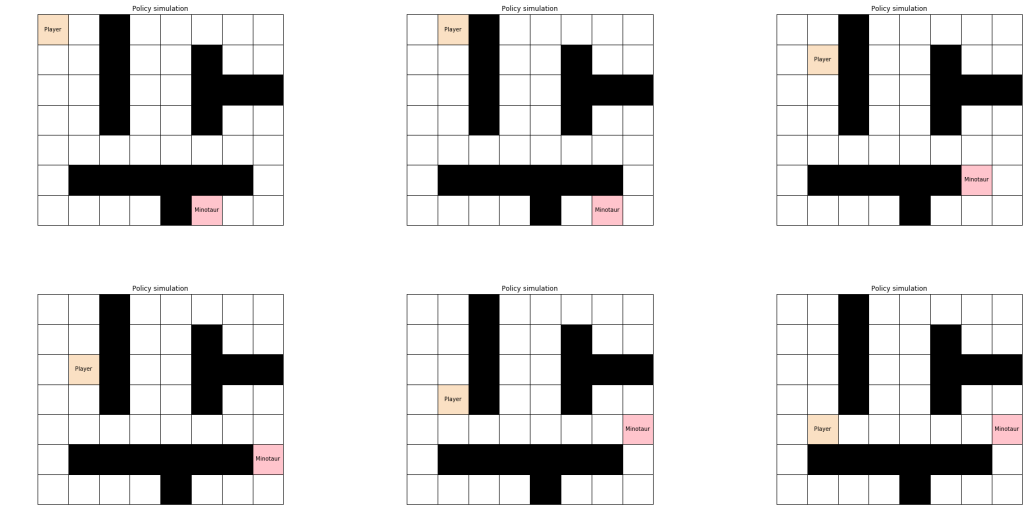


Figure 1: Policy simulation for optimal policy when Minotaur is allowed 4 actions namely Right,Left,Up,Down

When we allow the Minotaur to perform Stay action, we get the following simulation :



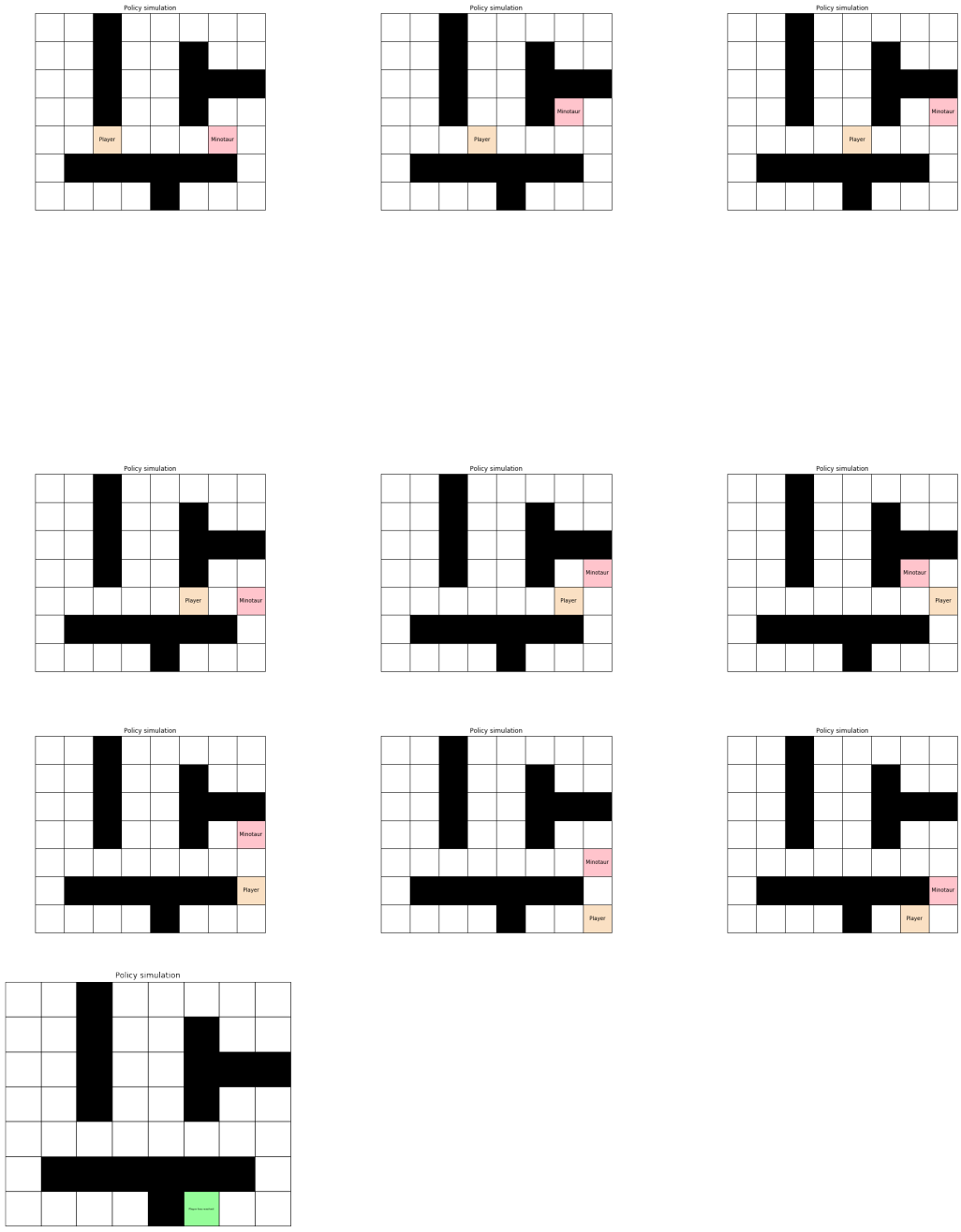


Figure 2: Policy simulation for optimal policy when Minotaur is allowed 5 actions namely Right,Left,Up,Down and Stay

We observe that the probability of exiting the maze increases over the time horizon, which also indicates that the chances of exiting is less at the beginning and increases gradually as the agent moves towards the goal .

The probability of exiting the maze is as follows :

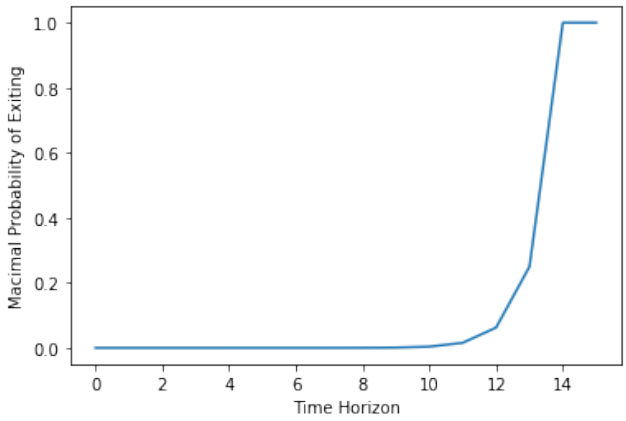


Figure 3: Maximal Probability(Minotaur not allowed to stay) Vs Time Horizon

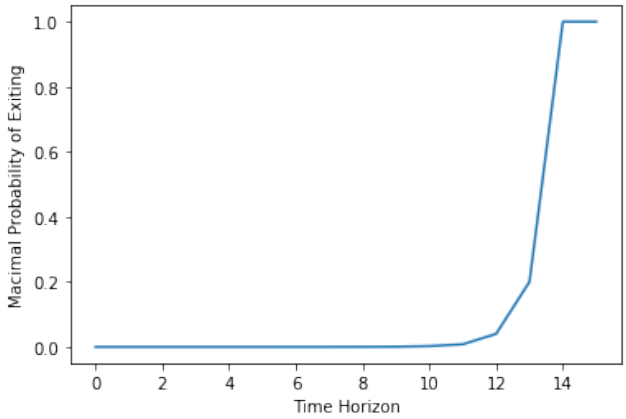


Figure 4: Maximal Probability (Minotaur not allowed to stay) Vs Time Horizon

The maximal Probability is asymptotically 0 for initial horizons and 1 when the goal is reached. We observe the difference between the slope in case of Minotaur Stay as the rewards and the transition value becomes different. When the Minotaur is allowed to stand still, we observe that there is a higher chance of being eaten. This is due to the expansion of the number of possible transitions when compared to the previous action set. As the transition matrix is of the dimension (S,S,A), with the action set increased the dimension becomes (2240)*(2240)*5. We ran the simulation for 10000 games when the Minotaur can perform a stay action, we obtained a win ratio of 7200/10,000, which is approximately 0.7 (70 %) where as when the Minotaur was only allowed to perform four actions the win probability obtained was 1.(winning all the games)

1.3 Case for Geometrically Distributed Life

Since the objective in this part was to minimize the expected time to exit the maze, the rewards in this case were chosen in order to give higher preference to reach the goal in short span, which was done by giving a penalty for Stay action. It was mentioned that the agent's life is geometrically distributed and the expected value for the geometrical distribution for agent's life is given by:

$$\mathbb{E}[T] = \frac{1}{(1-\lambda)}$$

Where λ is found to be λ=29/30. In this case we used value iteration to obtain an optimal policy by giving λ as the discount factor with epsilon = 0.001 which is making the agent to reach the goal at T=15. After getting the policy we simulated the game for 10000 times as asked. It was observed that

the probability of winning was 100%. when the Minotaur has 4 actions and 70% when it can also perform STAY action.

2 Problem 3: Bank Robbing(Reloaded)

2.1 Q-Learning

- **States[S]**: We define states as $[(r\_row, r\_col), (p\_row, p\_col)]$ where the first tuple corresponds to the position of robber and the second one is the position of the Police.

$$S = \{S_0, S_1, S_2 ... S_{255}\} \bigcup \{Caught\}$$

$$0 \leq r\_row \leq 3; 0 \leq r\_col \leq 3; 0 \leq p\_row \leq 3; 0 \leq p\_col \leq 3$$

- **Actions[A]**: The action space is a_{robber} ={UP, Down, Left, Right, Stay} for the robber and the action space for Police is a_{police}= {Up, Down, Left, Right}.
- **Rewards**: The following rewards are possible:

Rewards	Values
r(S=robber at Bank)	1
r(S=robber caught by Police)	-10

- **Objective-Function:**
  The objective is to find the optimum policy π^* such that:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{T=1}^{T} \lambda^{T-1} r(s_t^{\pi}, a_t^{\pi}) | s_1^{\pi} = s\right]$$

- **Transition Probabilities:** The non-zero transition probabilities $P_t(S'|S, a) = 1$

We get the following plot by implementing learning with epsilon-greedy by exploring action uniformly at random. We observe that the Q values after iteration become consistent as we can see in the plot. we have considered a random state and observed the change in Q values over 1e7 iterations.

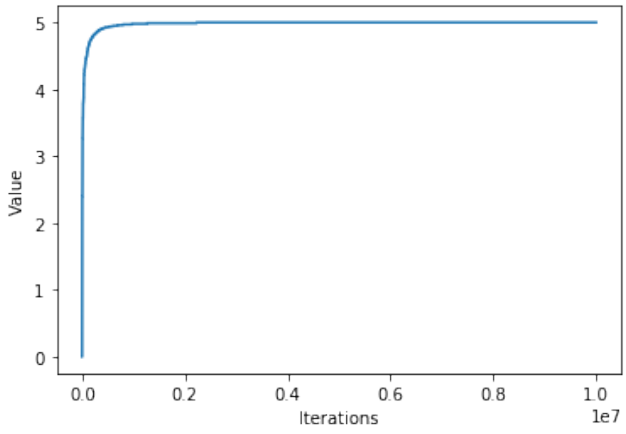


Figure 5: Q learning: Q-value versus Iterations

2.2 SARSA

The following plots for different values of ϵ were obtained:

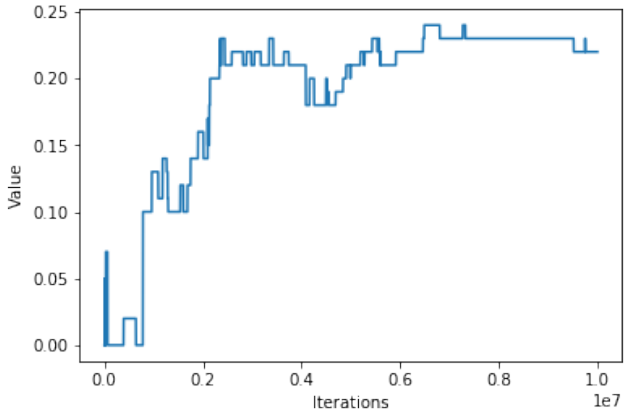


Figure 6: Value vs Iteration for $\epsilon = 0.1$

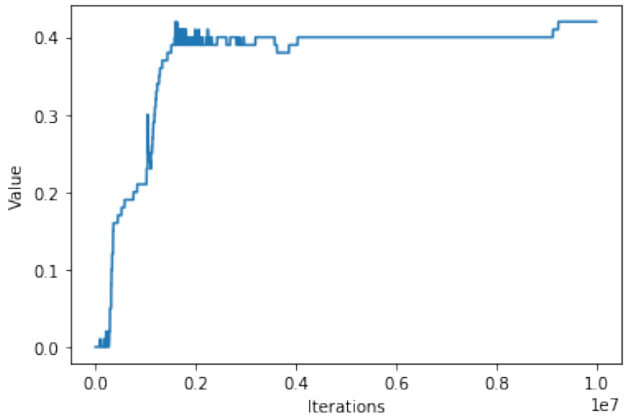


Figure 7: Value vs Iteration for $\epsilon = 0.009$

We observe that SARSA doesnot perform as better as Q learning and only converges at lower value of ϵ