EL2805 Reinforcement Learning

Computer Lab 1

Luís Santos
lmpss@kth.se

Simon Mello
smello@kth.se

December 2020

1 The Maze and the Random Minotaur

1.1 Formulating the problem as an MDP.

We can define an MDP as a tuple: $M = (S, A, P, r)$.

- **State space:** $S = \{(i, j, k, l) : \text{such that } (i, j) \neq \text{obstacle}\}$.
- **Action space:** $A = \{Stay, Left, Right, Up, Down\}$.

In our state space S, the tuple (i, j) is the player position and (k, l) is the Minotaur position. The Minotaur "actions" are encoded in the transition probabilities.

The finite-time horizon objective is

$$\max_{\pi} \mathbb{E}^{\pi} \left[\sum_{t=0}^{T} r_t(s_t, a_t) \right], \tag{1}$$

where our rewards are stationary, given by

$$\begin{aligned} &r(s = S_{wo}, a = A_{wo}) = -\infty, \\ &r(s = S_{\{i=k, j=l\}}, a = \cdot) = -\infty, \\ &r(s = S_{\{i=G_x, j=G_y\}}, a = \cdot) = 0, \\ &r(s = S', a = \cdot) = -1, \end{aligned}$$

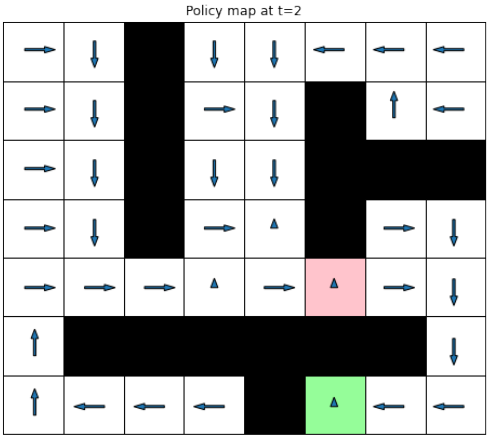
where S_{wo} is a state when the player is directly next to a wall or obstacle given any Minotaur position, and A_{wo} is the action that would move the player into the wall or obstacle, G_x and G_y are the goal coordinates, and S' is any state that is not a wall, an obstacle, the Minotaur or the goal. The player transitions are deterministic, however, due to the random nature of the Minotaur movement, the state we transition to is non-deterministic in some cases. The transition probabilities are then

$$
\begin{aligned}
&P(s' = S_{\{i\neq k, j\neq l\}} \mid s = S_{\{i=k,j=l\}}, a = \cdot) = 0,\\
&P(s' = s \mid s = S_{\{i=k,j=l\}}, a = \cdot) = 1,\\
&P(s' = S_{\{i\neq G_x, j\neq G_y\}} \mid s = S_{\{i=G_x,j=G_y\}}, a = \cdot) = 0,\\
&P(s' = s \mid s = S_{\{i=G_x,j=G_y\}}, a = \cdot) = 1,\\
&P(s' = S_{\{i'=i,j'=j\}} \mid s = S_{wo}, a = A_{wo}) = 1,\\
&P(s' = S'_{\{i',j',k',l'\}} \mid s = S'_{\{i,j,k,l\}}, a = \cdot) = \frac{1}{N_s},
\end{aligned}
$$

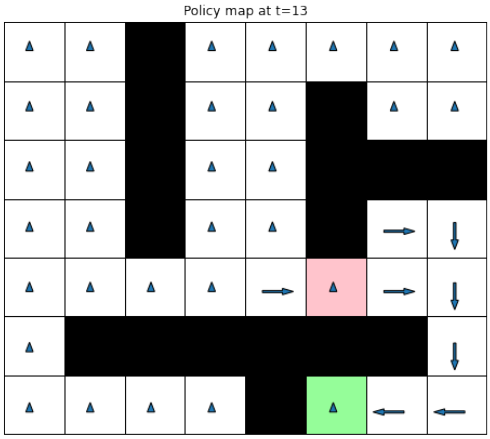
where N_s is the number of legal moves for the Minotaur and takes values 2, 3 or 4 depending on the position. For example, if the Minotaur is in a corner, it only has 2 legal moves. We are working under the assumption that the Minotaur can move through obstacles in such a way that it treats obstacles as normal squares and does not jump over the obstacle. However, both the player and the Minotaur are bounded by the maze walls.

1.2 Solving and illustrating for T=20.

With a finite-time horizon objective, we use dynamic programming (DP) to solve the problem. In Figures 1 and 2, we illustrate the optimal policy for each player position given specific Minotaur positions and time t, as the optimal policy is not stationary. These are shown both when the Minotaur can stay in the same position and when it is forced to move.

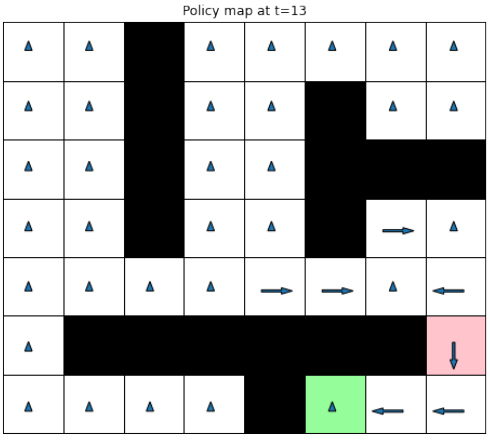


(a) Minotaur cannot stay, $t = 2$.

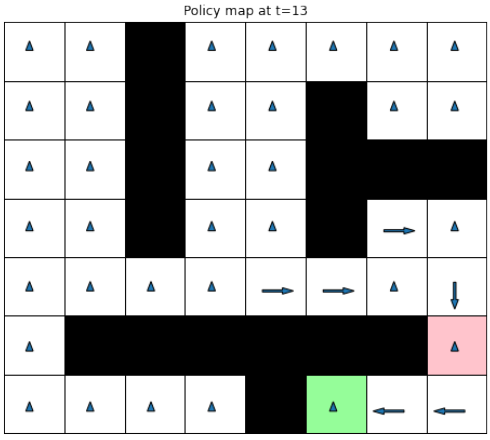


(b) Minotaur cannot stay, $t = 13$.

Figure 1: Comparison between optimal policies at different time steps.



(a) Minotaur can stay, $t = 13$.



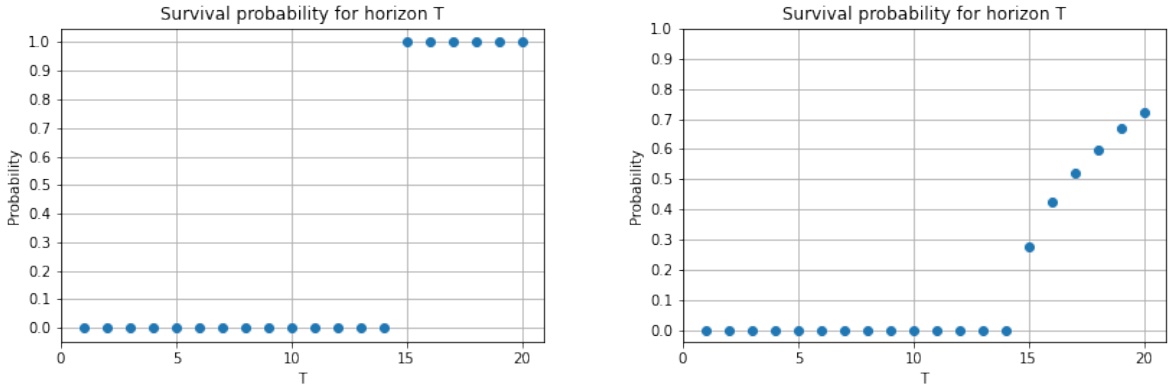
(b) Minotaur cannot stay, $t = 13$.

Figure 2: Comparison between optimal policies when Minotaur can and cannot stay.

As can be seen in Figure 1, as long as the player can reach the end the policy does not change. However, when the player is unable to reach the goal before the time runs out, the player simply does not try to; although the player still tries to avoid the Minotaur as that would incur an even greater cost.

In Figure 2, we observe that the change in optimal policy is such that the player no longer moves into the Minotaur position, as there is a chance that the Minotaur will occupy this position in the next time step now, which was not possible previously.

We can also look at the survival probabilities at each time step, comparing when the Minotaur is forced to move and when it can stay. This data is shown in Figure 3.



(a) Minotaur cannot stay.

(b) Minotaur can stay.

Figure 3: Survival probabilities at each time t, with different possible Minotaur moves.

When the Minotaur is forced to move, there exists an optimal policy that guarantees that the player reaches the goal, as the player is always able to walk onto the square where the Minotaur is currently. From there on out, the player can move freely towards the goal. This results in the player always choosing the shortest path, and as such it takes 15 moves to reach the goal.

When the Minotaur is able to stay, the player cannot act in a brutish manner anymore. The first possible time step that the player can win remains the

same, as there is a non-zero probability that the Minotaur behaves exactly the same and never stays. However, it is no longer guaranteed because if the Minotaur chooses to stay in choke points, it blocks the player.

1.3 Geometrically distributed life.

If we now assume that the player's life is geometrically distributed with mean 30, we can modify the problem slightly to solve it. To minimize the expected time to exit the maze, we model the problem as an infinite-time horizon MDP with

$$\mathbb{E}[T] = \frac{1}{1-\lambda} = 30, \tag{2}$$

where λ is the discount factor with value 29/30. Then the objective becomes

$$\max_{\pi} \mathbb{E}^{\pi} \left[\sum_{t=0}^{T} \lambda^{t-1} r_t(s_t, a_t) \right]. \tag{3}$$

The rewards and transition probabilities remain the same as before. To find the optimal policy we run the Value Iteration (VI) algorithm instead, as the DP algorithm does not work with infinite time. We run 10000 simulations with the VI algorithm, where $\epsilon = 0.0001$, to estimate the probability of getting out alive. Based on the simulations, we estimate that when the Minotaur is forced to move, the probability of exiting is 62.52%; and when the Minotaur can stay, the probability of exiting is 54.21%.

With an expected life of only 30, we are often dying before reaching the minimum amount of moves necessary to exit the maze, as we require 15 moves. That means that in roughly 42% of games, as $T \sim \text{Geo}(1/30)$, we die before even having a chance. Therefore, we cannot expect very high success rates.

2 Robbing Banks

2.1 Formulating the problem as an MDP.

We define the MDP for the bank robber as a tuple: $M = (S, A, P, r)$.

- **State space:** $S = \{(i, j, k, l)\}$.
- **Action space:** $A = \{Stay, Left, Right, Up, Down\}$.

In our state space S, the tuple (i, j) is the robber position and (k, l) is the police position. The police "actions" are encoded in the transition probabilities.

The infinite-time horizon objective is

$$\max_{\pi} \mathbb{E}^{\pi} \left[\sum_{t=0}^{T} \lambda^{t-1} r_t(s_t, a_t) \right], \tag{4}$$

where our rewards are stationary, given by

$$\begin{aligned}
&r(s = S_w, a = A_w) = -\infty, \\
&r(s = S_{\{i=k, j=l\}}, a = \cdot) = -50, \\
&r(s = S_{\{i=B_x, j=B_y\}}, a = \cdot) = 10, \\
&r(s = S', a = \cdot) = 0,
\end{aligned}$$

where S_w is a state when the robber is directly next to a wall given any police position, and A_w is the action that would move the robber into the wall, B_x and B_y are the bank coordinates, and S' is any state that is not a wall, the police or a bank.

The robber transitions are deterministic, however, due to the random nature of the police movement, the state we transition to is non-deterministic in some cases. The transition probabilities are then

$$\begin{aligned}
&P(s' = S_I \mid s = S_{i=k, j=l}, a = \cdot) = 1, \\
&P(s' = S_{\{i'=i, j'=j\}} \mid s = S_w, a = A_w) = 1, \\
&P(s' = S'_{\{i', j', k', l'\}} \mid s = S'_{\{i, j, k, l\}}, a = \cdot) = \frac{1}{N_s},
\end{aligned}$$

where S_I is the initial state, and N_s is the number of legal moves for the police which takes values 2 or 3 depending on the position. For example, if the police is in a corner, it only has 2 legal moves. Both the player and the police are bounded by the city walls.

2.2 Value function as a function of λ

In Figure 4 we show the value function as a function of λ at the initial state.

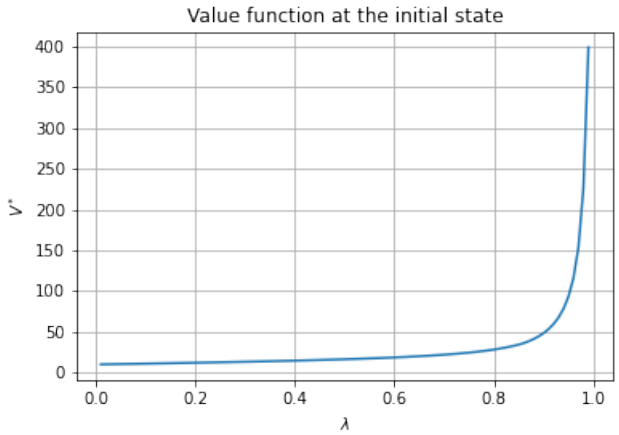


Figure 4: Value function at the initial state as a function of λ.

As we can expect, the value function increases with λ since we are factoring in future rewards more and more, resulting in infinite reward as λ goes to 1. Furthermore, when λ passes 0.8, we start observing a rapid increase in the value function. The robber then employs more long-term strategies, omitting risky moves and never getting caught.

2.3 Optimal policy for different λ

As hinted at above, the policy changes with λ. However, as seen in Figure 5, the changes are very small. We can see that a short-sighted robber prefers going back to bank 1 and gathering quick rewards; while the far-sighted robber understands that the police will take much longer to reach the right side since the police will not move with probability one half as often, but rather one third. For example, if the player is at bank 1 and the police is in grid cell (1,1), the police always moves towards you. On the other hand, if you are at bank 4, the police will find themselves in the middle row a lot and potentially move down. This means that while the robber is not collecting rewards as fast, in the long-term the robber can stay at those banks for longer, ultimately acquiring more reward.

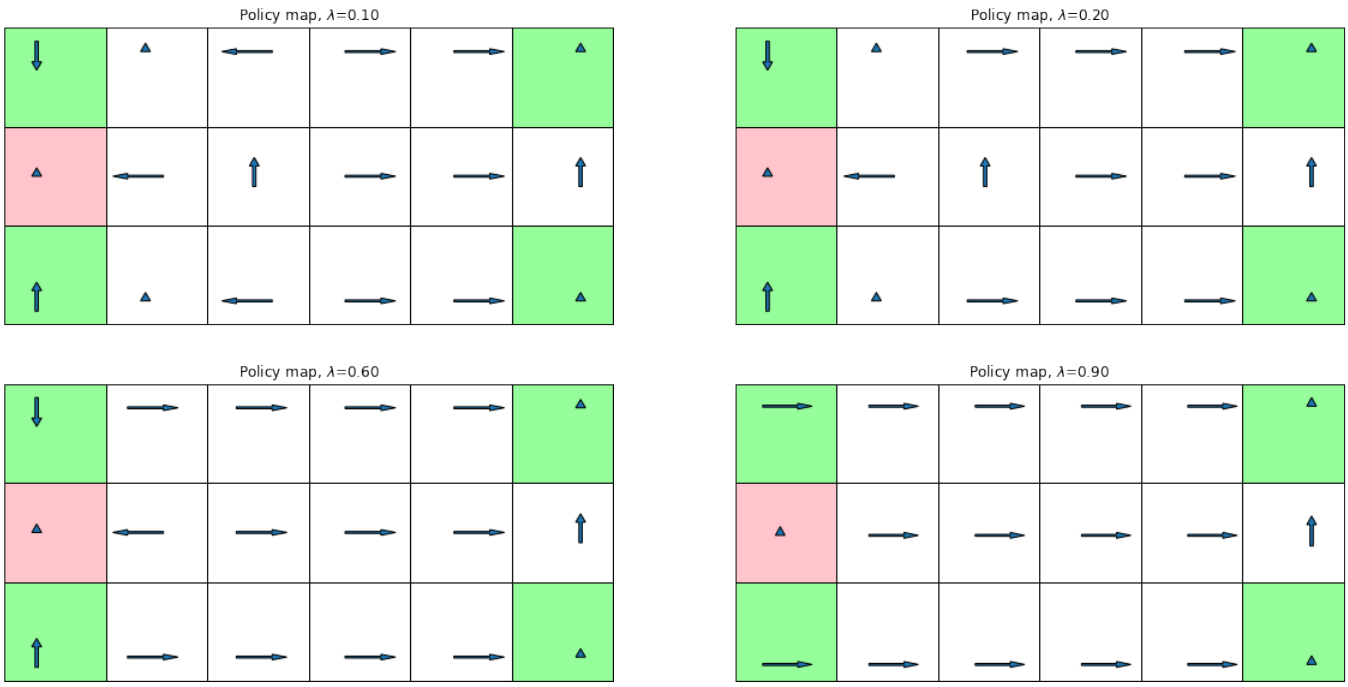


Figure 5: Optimal policies for different values of λ with a fixed police position.

The λ values shown are where we observe the policy changing, with the final policy change seen at $\lambda = 0.90$.