The Great Escape

Harald Stiff
Royal Institute of Technology
hstiff@kth.se

Christian Ryan
Royal Institute of Technology
cryan@kth.se

Abstract

In this report we consider the use of dynamic programming, value iteration algorithms, and Q-learning algorithms to optimize policies for various maze problems. In each section the modeling of each Markov decision problem is derived. Moreover the optimal policies are illustrated for different parameter choices. It is shown that the different algorithms are feasible to use on the proposed maze problems.

1 The Maze and the Random Minotaur

1.1 Method

Consider the maze in figure 1 where the goal is to move a character in position A to position B in less than T steps without ever being in the same position as a minotaur which spawns in position B.

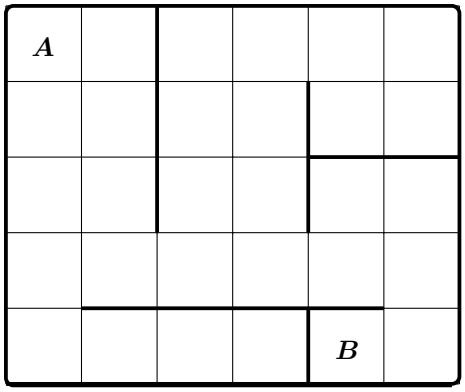


Figure 1: This figure shows the maze.

The character and the minotaur can only move one step at a time but only the minotaur can walk through the walls inside the maze. Moreover the minotaur follows a random walk and moves simultanuously with the character. To identify the optimal policy of the character the problem is modeled as a Markov Decision Process (S, A, P, R, λ). The state space is

$$S = \{(i,j)\} \times \{(i,j)\} \cup S_{\text{end}} \tag{1}$$

$$0 \leq i \leq 4 \tag{2}$$

$$0 \leq j \leq 5 \tag{3}$$

where i and j are integers. That is, the product of the character's postion and the minotaur's position plus the terminal states S_{end} of either reaching the end or hitting the Minotaur. Furthermore, the

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

character can move freely or stand still while following the limits of the walls giving an action space of

$$A = \{\text{Up, Down, Left, Right, Stay}\}. \tag{4}$$

Let each state $s_t = (s_t^C, s_t^M)$ where s_t^C is the position of the character and s_t^M is the position of the minotaur at time t. Moreover let $\mathcal{N}(s_t^M)$ be the set of neighbouring positions of the minotaur and $\mathcal{K}(s_t^C)$ the set of neighbouring positions of the character which are not separated by a wall. Lastly we define a boolean function $d(s_t^C, s_{t+1}^C, a)$ which is true if s_{t+1}^C is to the right of s_t^C and $a =$ right, s_{t+1}^C is to the left of s_t^C and $a =$ left etc. With these assumptions we define a move $(s_t^C, s_t^M) \mapsto (s_{t+1}^C, s_{t+1}^M)$ legal if and only if

$$s_{t+1}^M \in \mathcal{N}(s_t^M) \wedge s_{t+1}^C \in \mathcal{K}(s_t^C) \wedge d(s_t^C, s_{t+1}^C, a) \tag{5}$$

is true. At each time instant the transition probabilities are then given by

$$p(s_{t+1}|s_t, a) = \begin{cases} \frac{1}{|\mathcal{N}(s_t^M)|} & \text{if Legal move and } \mathrm{s}_t \notin S_{\text{end}} \\ 1 & \text{if } \mathrm{s}_{t+1} \in S_{\text{end}} \text{ and } s_t \in S_{\text{end}} \\ 0 & \text{else.} \end{cases}. \tag{6}$$

There is only one terminal reward

$$r_T(s_T) = \begin{cases} 1 & \text{if } \mathrm{s}_T \in S_B \\ 0 & \text{else} \end{cases} \tag{7}$$

where $S_B = \{\forall s \in S | s^C = (4,4),\ s^M \neq s^C\}$. The other rewards obtained during the remaining time steps are set to zero. The objective is to obtain a policy that maximizes the expected future reward

$$\mathbb{E}\left[\sum_{t=0}^{T-1} r_t^\pi(s_t) + r_T(s_T)\right]. \tag{8}$$

To obtain this policy the Bellman equations

$$u_T^*(s_T) = r_T(s_T) \tag{9}$$

$$u_t^*(s_t) = \max_a \left\{ r(s_t, a) + \sum_{j \in S} p(j|s_t, a) u_{t+1}^*(j) \right\} \tag{10}$$

are solved yielding optimal decision rules

$$\pi_t(s_t) = \arg\max_{a \in A_{s_t}} \left\{ r(s_t, a) + \sum_{j \in S} p(j|s_t, a) u_{t+1}^*(j) \right\}. \tag{11}$$

resulting in a policy of $\pi = (\pi_i)_{i=0}^{T-1}$. When the horizon is a geometrically distributed random variable with mean $\frac{\lambda}{1-\lambda}$ the objective is equivalent to maximizing the expected discounted future reward

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t r_t(s_t, a_t)\right] \tag{12}$$

which is solved by using the value iteration algorithm.

1.2 Policy Evaluation

The performances of the policies obtained from the Bellman equation and the value iteration algorithm are shown in figure 2. One can spot a significant performance difference between the case when the minotaur can stand still and when it can not stand still. The reason being is that the possibility of the minotaur standing still causes a danger of moving towards the minotaur when character is beside it. Moreover it introduces a possibility of the minotaur standing still and blocking the path to the goal for multiple time steps.

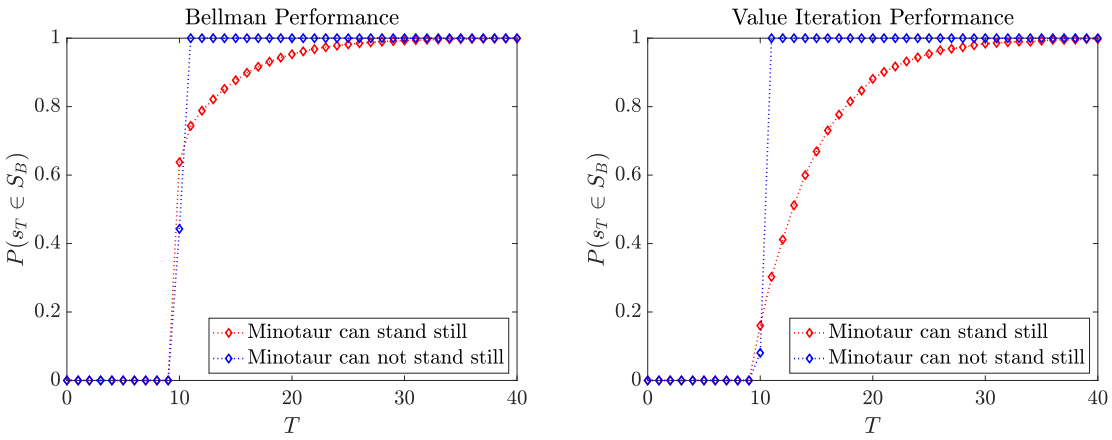


Figure 2: Performances of the policies. The probability of reaching the end of the maze is plotted against the time horizon T.

The value iteration performance plots were obtained by simulating the performance for each T 10000 times. The Bellman performance plots were obtained from the fact that

$$p(s_T \in S_B) = V_T^*(s_A) \tag{13}$$

holds in the way the rewards were defined in (7). Furthermore, when $T \sim \text{geo}(\lambda)$

$$p(s_T \in S_B | T \sim \text{geo}(\lambda)) = \sum_{t=0}^{\infty} \lambda^t (1-\lambda) V_t^*(s_A) \tag{14}$$

that is, a geometrically weighted sum of the probabibilities obtained from (13). In the case where the minotaur can not stand still and we get

$$\sum_{t=0}^{\infty} \lambda^t (1-\lambda) V_t^*(s_A) = \lambda^{10}(1-\lambda) V_{10}^*(s_A) + \sum_{t=11}^{\infty} \lambda^t (1-\lambda) \tag{15}$$

$$= \lambda^{10}(1-\lambda) V_{10}^*(s_A) + 1 - (1-\lambda)\frac{(1-\lambda^{11})}{(1-\lambda)} \tag{16}$$

$$= \lambda^{10} V_{10}^*(s_A) + \lambda^{11}(1 - V_{10}^*(s_A)) \tag{17}$$

$$\approx 0.70. \tag{18}$$

This result is also consistent with results obtained by sampling $T \sim \text{geo}(\lambda)$ and simulating 10000 games. In the case where the minotaur can stand still the probability is estimated to 0.63 from a simular experiment.

2 Robbing Banks

2.1 Method

For this task, consider the maze in figure 3 where the goal is to rob a bank for as long as possible, without being caught by a chasing police officer. The robber spawns in position B1 (Bank 1), and the cop in position PS (police station). If the robber gets caught by the cop, by simultaneously standing on the same square, this is where they respawn to.

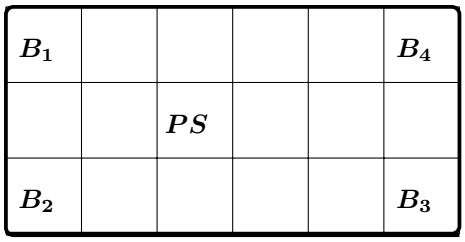


Figure 3: Illustration of cop & robber maze.

To maximize the robber's average discounted reward, a Markov Decision Process (S, A, P, R, λ) needs to be established. The state space is simply,

$$S = \{(i,j)\} \times \{(i,j)\} \tag{19}$$

$$0 \leq i \leq 2 \tag{20}$$

$$0 \leq j \leq 5 \tag{21}$$

analogous to eq. (1)-(3) in section 1. There is no terminal state as the robber wishes to rob until the end of time if that implies maximizing its reward. As in section 1, the robber shares the same freedom, or lack thereof, in movement as long as it does not violate the limits of the walls, i.e.

$$A = \{\text{Up, Down, Left, Right, Stay}\}. \tag{22}$$

The transition probabilities for this task will be an extension of section 1, i.e. eq. (6), except for the fact that no walls are present, just the limits of the maze. Furthermore, the police officer always moves randomly in the robber's direction, however, has only three valid neighbours to move to in the direction of the robber, if the robber and cop are either on the same row or column in maze; else the cop has only two legal neighbours to move to in the direction of the robber, as it is always chasing. Let this be denoted respectively as $S_a = \{\forall s \in S | s^C = (a,b), s^M = (a,c)\}$, $S_b = \{\forall s \in S | s^C = (a,b), s^M = (c,b)\}$, $S_c = \{\forall s \in S | s^C = (a,b), s^M = (c,d)\}\ \forall\ a,b,c,d \in (i,j)$. Let $\mathcal{G}$ be defined as, $\mathcal{G}(s | s \in S_a \cup S_b) = 3,\ \mathcal{G}(s | s \in S_c) = 2$. This yields,

$$p(s_{t+1}|s_t, a) = \begin{cases} \frac{1}{\mathcal{G}(s)} & \text{if Legal move,} \\ 0 & \text{else.} \end{cases}. \tag{23}$$

The rewards are as follows, $+10$ SEK for each time step the robber is standing at any bank (B1, B2, B3, B4) and the cop is not there. If the robber is caught standing on the same square as the cop, a penalty of -50 SEK will be imposed and both parties will respawn at their respective initialization positions; for any other case no reward (0 SEK) is given. The reward function is then,

$$r_t(s_t) = \begin{cases} 10 & \text{if } \mathrm{s}_t \in S_B \\ -50 & \text{if } \mathrm{s}_t \in S_F \\ 0 & \text{else} \end{cases} \tag{24}$$

where $S_B = \{\forall s \in S | s^C \in [(0,0),(2,0),(0,5),(2,5)],\ s^M \neq s^C\}$ and $S_F = \{\forall s \in S | s^M = s^C\}$, using the same notation as in section 1, however, the cop and robber are represented as the minotaur and character, respectively.

The objective is to obtain a policy that maximizes the expected discounted future reward, for $\lambda \in [0,1)$

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t r_t(s_t, a_t)\right] \tag{25}$$

which is solved by using the value iteration algorithm.

2.2 Results

The MDP was trained with the value iteration algorithm for 100 different values of $\lambda \in [0,1)$. In figure 4 the expected future reward is plotted against the discount rate from the initial state.

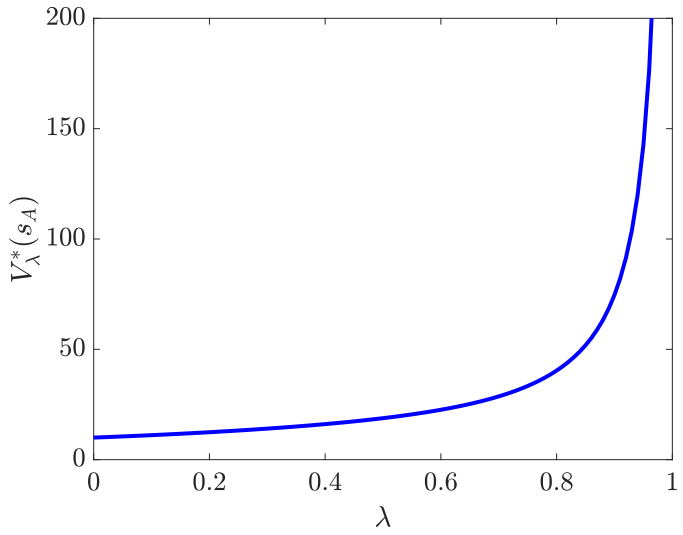


Figure 4: The expected future reward plotted against the discount rate.

The obtained plot is to be expected since

$$\lim_{\lambda \to 0} V^*_\lambda(s_A) = r(s_0|s_0 = s_A) = 10 \tag{26}$$

from (24) and

$$\lim_{\lambda \to 1} V^*_\lambda(s_A) = \infty \tag{27}$$

as there are nonzero rewards yielding a geometric sum that diverges when λ approaches 1. For illustrations of a random walk under optimal policy for $\lambda = 0.8$ and $\lambda = 0$, please see appendix. For lower values of λ, the future reward is more discounted, meaning the robber will value current reward higher. A $\lambda = 0$ thus implies that the robber values grabbing a reward of $+10$ SEK now higher than potentially gaining a -50 SEK penalty in the next time step. If each time step would be represented as days past, then an appropriate discount factor would be the annual inflation rate of the SEK averaged over each day of the annum.

3 Bank Robbing (Reloaded)

In the third problem the bank robbing scenario is revisited. This time there is one bank at position B which the character who's initially starting at position A is trying to rob without being caught by a police which moves randomly and spawns in the opposite corner of the character. The map is shown in figure 5.

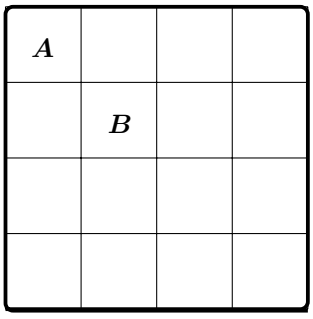


Figure 5: This figure shows the map.

3.1 Model

The state space of this problem is

$$S = \{(i,j)\} \times \{(i,j)\} \tag{28}$$
$$0 \leq i \leq 3 \tag{29}$$
$$0 \leq j \leq 3 \tag{30}$$
$$i, j \in \mathbb{N} \tag{31}$$

and the action space is

$$A = \{\text{Up, Down, Left, Right, Stay}\}. \tag{32}$$

The character is given a reward of 1 SEK each time it's staying inside the bank and -10 SEK each time it's caught by the police, and zero otherwise. More precisely, let each state $s_t = (s_t^C, s_t^P)$ where s_t^C be the position of the character at time t and the s_t^P is the position of the police at time t. The rewards are then given by

$$r_t = \begin{cases} 1 & \text{if } s_t^C = (1,1) \text{ and } s_t^C \neq s_t^P \\ -10 & \text{if } s_t^C = s_t^P \\ 0 & \text{else.} \end{cases} \tag{33}$$

The transition probabilities are nearly equal to those in problem 1. In this case the probabilities are given by

$$p(s_{t+1}|s_t, a) = \begin{cases} \frac{1}{|\mathcal{N}(s_t^P)|} & \text{if } s_{t+1}^C \in \mathcal{N}(s_t^C) \cup \{s_t^C\} \text{ and } s_{t+1}^P \in \mathcal{N}(s_t^P) \text{ and } d(s_t^C, s_{t+1}^C, a) \\ 0 & \text{else.} \end{cases}. \tag{34}$$

where $\mathcal{N}(s_t^i)$ again is the set of neighbouring positions of s_t^i and $d(s_t^C, s_{t+1}^C, a)$ is a boolean function which is true if the action corresponds to the same direction as the transition $s_t^C \mapsto s_{t+1}^C$. The optimal policy is obtained from two algorithms, the off policy Q-learning algorithm and the on policy SARSA algorithm.

3.2 Results

In figure 6 the Q-function is plotted from the initial state using the Q-learning algorithm with a random behavior policy π_b. It can be seen that the action down yields a higher return than standing still from the initial state.

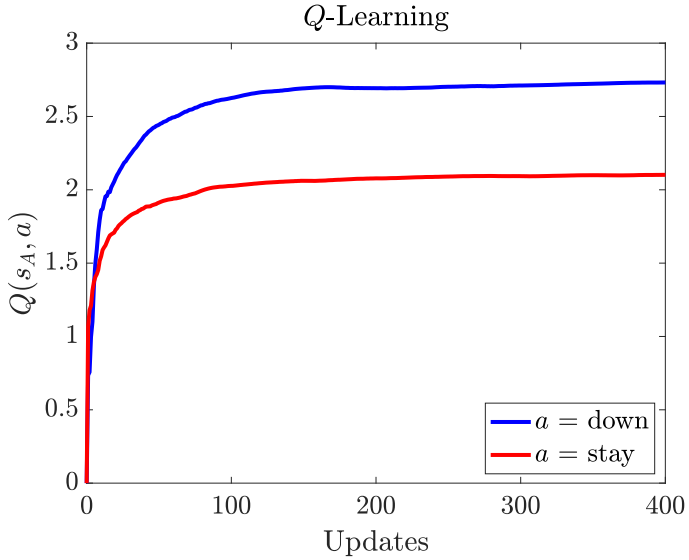


Figure 6: The Q-function for the initial state and two different actions is plotted against the amount of times the specific element has been updated.

In figure 7 the convergence of the SARSA-algorithm is plotted for different ϵ-greedy policies. Here we define and ϵ-greedy action in the following manner. With probability $1 - \epsilon$ an optimal action is chosen and with probability ϵ a nonoptimal action is chosen. It can be seen that the algorithm converges to smaller values with a higher value of ϵ. This is of no surprise since an increase in ϵ decreases the optimality of the policy.

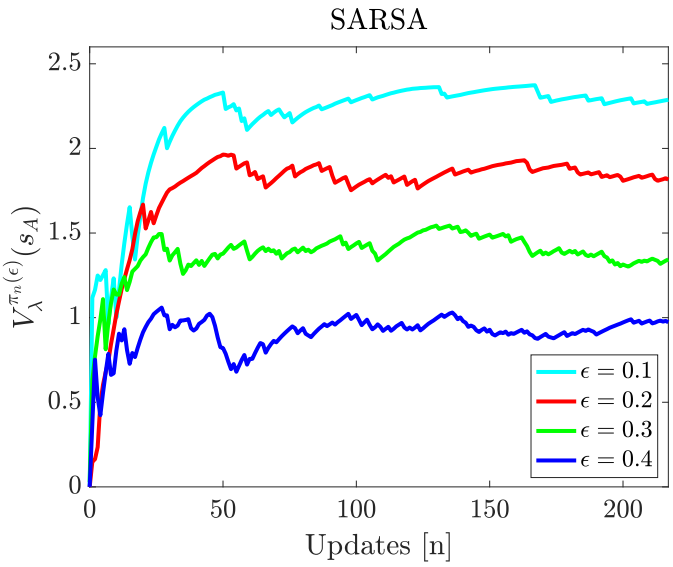
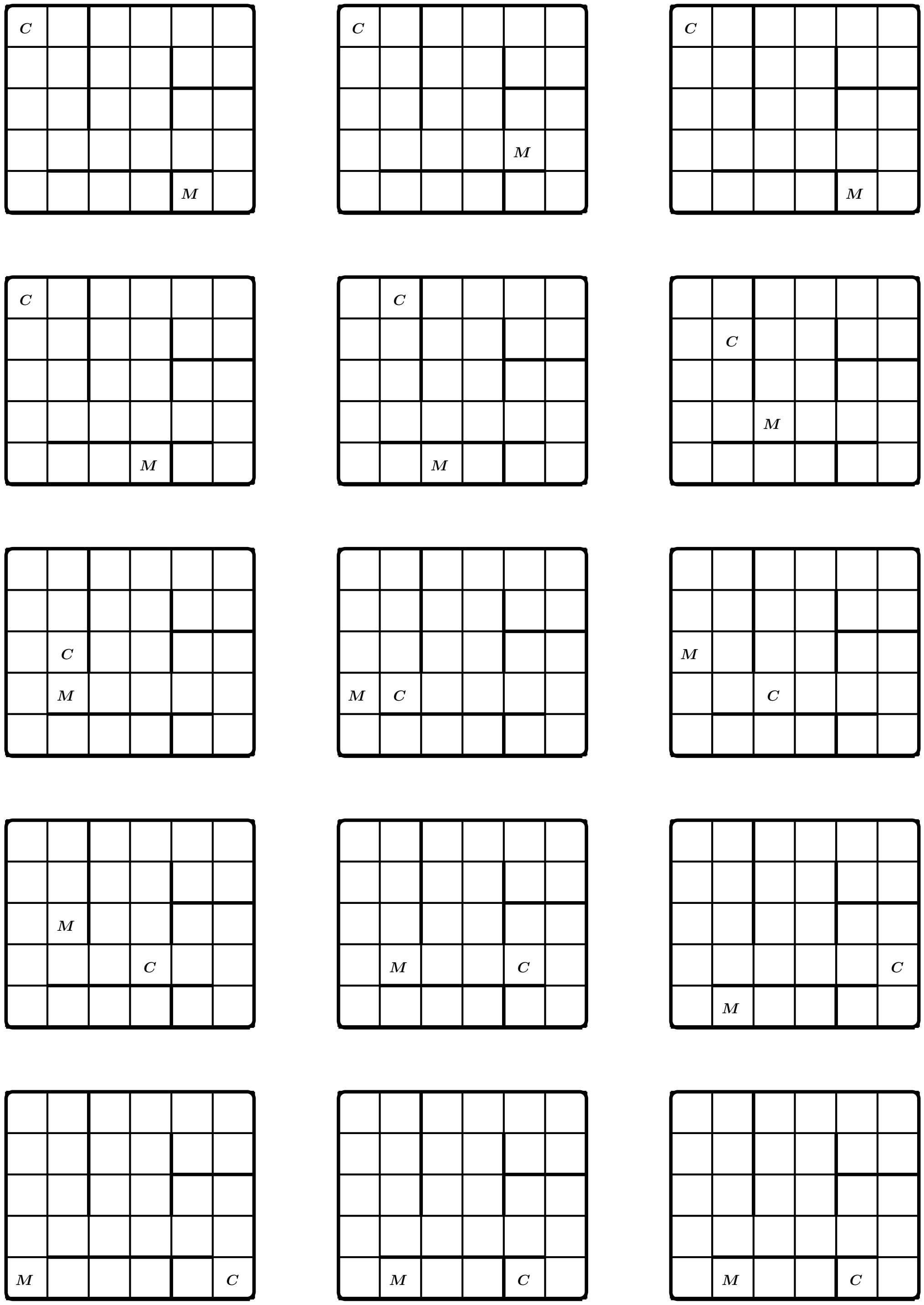


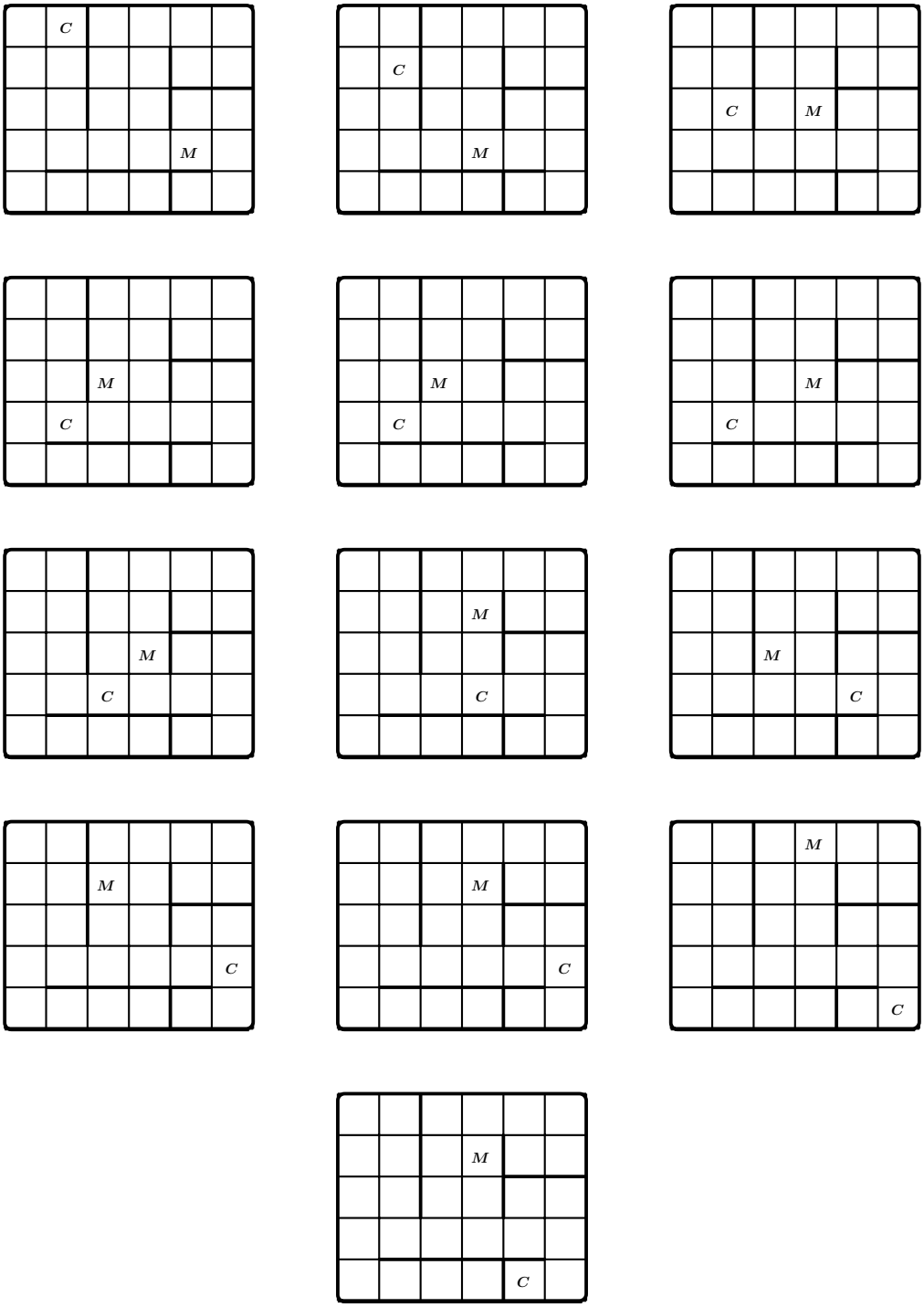
Figure 7: Convergence of the SARSA-algorithm for different ϵ-greedy policies.

A Illustration of Optimal Policies for Problem 1

A.1 Bellman Policy Without Still Minotaur $T = 15$

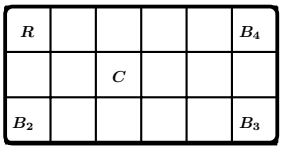


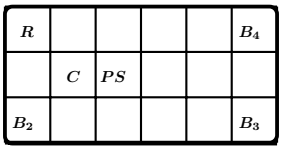
A.2 Bellman Policy With Still Minotaur

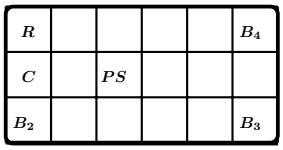


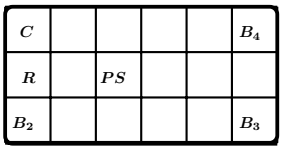
B Illustration of Optimal Policies for Problem 2

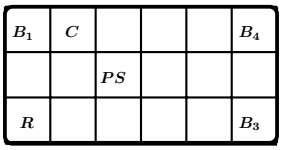
B.1 Policy for $\lambda = 0.8$

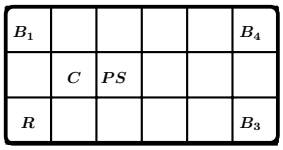


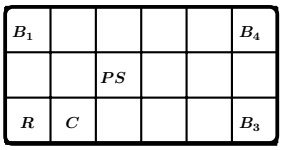


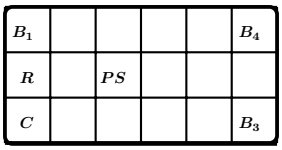


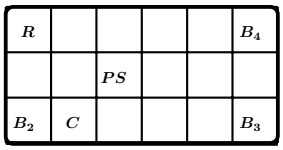












B.2 Policy for $\lambda = 0$