Artificial Intelligence – Technical Report

Assessed Exercise (CompSci 4004)

Loch Lomond Environment: Open AI gym FrozenLake-v0

Submitted to:

Dr Bjorn Jensen
Lecturer in Computer Science
School of Computing Science
University of Glasgow

Report Prepared by:

Student GUID: 2383746W
MSc in Data Analytics
School of Mathematics & Statistics
University of Glasgow

March 23, 2019

Table of Contents

List of Tables

1 Introduction

The task is based on the OpenAI gym environment "ForzenLake-v0" with the aim to implement three different types of agents that can safely navigate from the shore to the middle of a frozen lake in order to retrieve a frisbee that was thrown there. This lake consists of uncertainties that might influence a controlled move on the ice, namely holes and slippery conditions on the ice.

The problem is represented by an 8x8 grid world which consists of 64 possible states. More precisely, our agents can only take discrete moves to get from the starting position (S) to the goal state (G) and can either move on the frozen surface (F) or encounter a hole in the ice (H).

- **S**: Starting Position (shore)
- **G**: Goal State (frisbee)
- **F**: Frozen Surface
- **H**: Hole in the ice

The task environment for the Loch Lomond frozen lake problem will be illustrated below for problem ID 1 (left) and ID 7 (right).

H	S	F	F	F	H	F	F
F	F	F	F	F	F	F	F
F	F	F	H	F	F	F	F
F	F	F	F	F	H	F	F
F	F	F	H	F	F	F	F
F	H	H	F	F	F	H	F
F	H	F	F	H	F	H	F
F	v	F	H	F	G	F	H

H	F	F	F	F	H	F	S
F	F	F	F	F	F	F	F
F	F	F	H	F	F	F	F
F	F	F	F	F	H	F	F
F	F	F	H	F	F	F	F
F	H	H	F	F	F	H	F
F	H	F	F	H	F	H	F
F	F	F	H	F	F	v	G

2 Analysis

The Following table represents the PEAS specification for our given frozen lake problem.

Table 1: PEAS Analysis

Performance measure	- *Average success per episode* - *Episodes won* (total number of times reached goal) - *Max iterations per episode* - *Mean iterations per episode* *Average success per episode* is simply the ratio calculated as the number of episodes won by the agent divided by the total number of episodes.
Environment	8x8 grid with 64 possible states. Differing start and goal state locations for each task environment.
Actions	Left: 0 , Down: 1 , Right: 2 , Up: 3
Sensors	Current state coordinates (x, y) within the frozen grid

Furthermore, the following three types of agents are considered.

- **Random Agent** (stochastic, unknown and partially observable environment, static)
- **Simple Agent** (deterministic, known and fully observable environment, static)
- **Reinforcement Agent** (stochastic, known and partially observable environment, static)

Where the first two agents take episodic decisions and the reinforcement agent is assumed to behave sequentially. In the stochastic environments, it is assumed that the agent will never oppose any desired action (after choosing “up”, impossible that “down” occurs). Consequently, each non-opposing action results in an equal probability of one third. Movement outside the grid bounds is not possible, confrontation with a wall will impede further movement. Non-terminal states have 0 rewards and the goal state is rewarded with 1. Only in case of the RL-Agent holes are punished with a negative reward of -0.01, while for random and simple agent holes are simply set to 0.

3 Methodology

In the following, we will formally define the algorithms involving our three agents.

3.1 Random Agent

Entirely stochastic behaviour, based on uncertainty (see Analysis section).

3.2 Simple Agent

The chosen implementation is based on the A* search algorithm which assumes perfect information in a fully observable environment. Using the notation from AIMA (p. 93) this is given by:

$$\boldsymbol{f(n) = g(n) + h(n)}$$

Where for the frozen lake problem we have;

- $f(n)$: the estimated cost of chthe eapest solution through n (given grid position)
- $g(n)$: path cost from the start node S to node n
- $h(n)$: ha euristic function that estimates the cost of the cheapest path from the node n to the goal

At each state, the A* algorithm explores all child nodes and lists all options that reach G as a priority queue where the cheapest path to the goal is given priority in the list. This approach allows screening for the optimal path from S to G at each step.

3.3 Reinforcement Agent

A tabular Q-learning agent operates model-free. For each episode, the agent takes an action (either random or one with the highest reward based on the premature Q-table), then observes the effect/reward of the action taken to update the temporal difference Q-table using the formula (AIMA, page 844):

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[R(s) + \gamma\, max_{a'}\, Q(s',a') - Q(s,a)]}$$

Where:

- s: is the current state (i.e current position in the grid)
- s': is the next state after reaching s
- a: is the current action (i.e left, up, right or down)
- a': is the next action taken after a
- α: is the learning rate whith range $[0,1]$
- γ: discount factor whith range $[0,1]$
- $R(s)$: reward of the current state
- $Q(s,a)$: value of action a in state s

3.4 Q-value maximation and utilities in a model-free approach

As suggested in the literature (AIMA, p. 843) the Q-values are related to utilities in the following way:

$$\boldsymbol{U(s) = max_a\, Q(s,a)}$$

This is said to be a model-free temporal difference Q-learner since it does not require a transition model of the form $P(s'|s,a)$ for either learning or action selection. Justification of this approach is due to the partially observable environment

4 Implementation

Next, it is described on which functions and toolboxes the implementation of our agents will be based.

4.1 Random Agent

For random agent *env.action_space.sample()* is used to obtain a random action at each iteration within the given episode n.

4.2 Simple Agent

Within the AIMA Toolbox we used the classes *UndirectedGraph* as well as *GrapProblem* for mapping the environment to our frozen lake context calling the *astar_search* function.

The latter function follows the logic of a priority queue and ensures the cheapest path with minimal distance from S to G is found. Since diagonal moves in the grid are invalid, the heuristic used is the *Manhatten Distance*.

4.3 Reinforcement Agent

Since our RL-Agent (Q-learning) learns through exploration and in order to update the values in the Q-table we must set the right combination of parameters, namely learning rate and discount factor. Hole states were punished with a reward of -0.01.

Getting inspired by labs and lectures, $\alpha = 0.8$ and $\gamma = 0.8$ was chosen for the learning rate and discount factor respectively. Since the discount factor is high, we guarantee that our agent has a preference for long term (future) rewards instead of short term (immediate) once. With the high learning rate w,e ensure that our agent becomes more "deterministic" while "uncertainty" is reduced. Hence, recent information is considered more important than exploring prior knowledge.

Actions are selected probabilistically with $\frac{1}{(episode+1)}$, that is with increasing number of episodes the RL-Agent becomes less stochastic and more deterministic. Hence, the agent finds an optimal policy to navigate from S to G with more confidence. To detect potential learning we also evaluated the final 1000 episodes and compared these to the overall results.

5 Evaluation

In comparison to all other task environments (ID 1 to 7), problem ID 0 is the most difficult task environment and all our implemented agents struggle in reaching the goal state, with exception of the simple agent that always reaches G. The performance of the random agent and of the reinforcement agent differ according to the difficulty of the task environment. Visual results for our three agents will be provided for all problem ID's within the Appendix section.

5.1 Performance measures and setup

For the general setup, we chose a total number of 10 000 episodes to be run, where each episode consists of at max. 2000 iterations. In order to add some challenge to the simple agent, we limited the maximum number of iterations to 100. The performance measures are:

- Max iterations per episode
- Mean iterations per episode
- Average success per episode (Mean rewards)
- Episodes won (total reward)

5.2 Results Random Agent

Table 2, shows that Max iterations per episode was lowest for ID 3 and apart from task environment 0, highest for ID 5. Aside from problem ID 7, our agent won most episodes in task environment 2 which is further reflected in *Average success per episode*.

Table 2: Overall performance measure random agent

Problem ID	Average success per episode	Episodes won	Mean iterations per episode	Max iterations per episode
0	0.0001	1	28.2478	205
1	0.0002	2	9.0713	103
2	0.0037	37	11.5314	95
3	0.0006	6	10.4926	72
4	0.0003	3	7.8951	84
5	0.0009	9	24.2741	155
6	0.0001	1	10.3318	134
7	0.012	120	14.6973	112

5.3 Results for Simple Agent (A*)

It is evident from table 3 below, that the simple agent needs very few iterations (steps) to reach the goal state.

Table 3: Overall performance measure simple agent

Problem ID	Average success per episode	Episodes won	Mean iterations per episode	Max iterations per episode
0	1.0	10 000	13.0	13
1	1.0	10 000	11.0	11
2	1.0	10 000	9.0	9
3	1.0	10 000	10.0	10
4	1.0	10 000	9.0	9
5	1.0	10 000	9.0	9
6	1.0	10 000	11.0	11
7	1.0	10 000	7.0	7

Since the simple A* agent always reaches the goal (mean reward 1) for each episode, it is of no further interest to plot these results in this section. Visual results, however, will be provided in the Appendix (see section 8).

5.4 Results for RL-Agent (Q-learning)

Our RL-Agent had the highest *average success per episode* in problem ID 5, but on average required fewer steps to reach the goal in ID 7. Task environments 2 to 4, return similar numerical results and can, be seen as similar in difficulty.

Table 4: Overall performance measure RL-Agent

Problem ID	Average success per episode	Episodes won	Mean iterations per episode	Max iterations per episode
0	0.0001	1	1909.8472	2000
1	0.2235	2235	399.2737	2000
2	0.4392	4392	169.8095	2000
3	0.4196	4196	196.5758	2000
4	0.4253	4253	239.0579	2000
5	0.9478	9478	317.0657	2000
6	0.2607	2607	182.6703	2000
7	0.7872	7872	86.5917	2000

Since learning takes place through experience, we limit ourselves to the final 1000 episodes in each task environment and compare these results to the overall table. We note that despite a few losses, on average the agent was able to learn.

Table 5: Performance measure last 1000 episodes RL-Agent

Problem ID	Average success per episode	Episodes won	Mean iterations per episode	Max iterations per episode
0	0.0	0	1980.994	2000
1	0.238	238	376.842	2000
2	0.5	500	183.444	1176
3	0.298	298	247.924	2000
4	0.361	361	292.047	2000
5	0.984	984	330.84	2000
6	0.435	435	194.378	1792
7	0.895	895	72.992	1010

Based on *average success per episode* our agent was able to learn:

- *Significantly* in problem ID’s 2, 6 and 7 (improvement 40% , 12% and 12% respectively)
- *Slightly* in problem ID’s 1 and 5 (improvement approx. 4% in both cases)

The agent was not able to learn in task environments 3 and 4, which results in a performance loss of 29% and 15% respectively.

5.5 Policy

The suggested policies for our RL-Agent for the hardest (ID 0) and easiest maps (ID 7) are displayed left and right respectively. Policies for the other ID’s can be found in the Appendix.

S	^	<	^	<	.	>	v
<	^	^	^	^	v	>	<
<	<	<	.	>	^	v	>
<	^	<	v	<	.	>	<
<	^	<	.	>	v	^	^
<	.	.	v	^	<	.	>
<	.	v	^	.	^	.	^
<	v	<	.	G			.

.	>	^	v	<	.	>	S
v	^	^	^	^	v	>	v
<	v	<	.	>	^	>	<
<	^	^	^	<	.	>	>
<	^	v	.	>	v	^	>
<	.	.	v	^	<	.	>
<	.	>	>	.	>	.	>
<	v	<	.	>	>	v	G

Where dots denote holes (H) and blank squares represent unvisited frozen states (F).

6 Conclusion

In terms of success rates in reaching the goal state (G), the random agent performs worst due to the unknown environment and stochastic nature of the problem. Since the A* agent is perfectly informed in a fully observable and non-stochastic environment, it is guaranteed that it will always reach the goal state without moving into hole states.

The Q-learner is a mix of both agents (random and A*) where the initial behaviour tends to be probabilistic in an unknown environment, but knowledge of the environment is sequentially improved with increasing number of episodes. Hence, actions towards the final episodes are taken more deterministically, in some problem ID’s (5 and 7) approximating the performance of the A* solution.

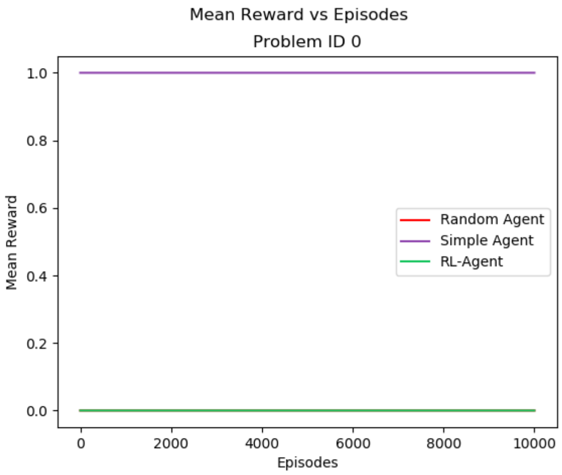
7 References

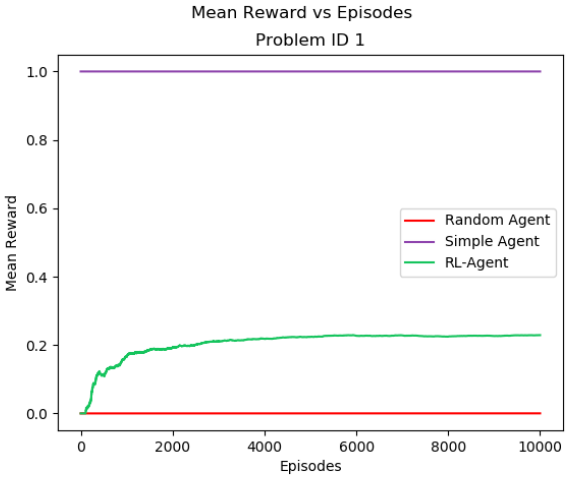
[1] S. Russell, and P. Norvig (2009). Artificial Intelligence: A Modern Approach (3rd Edition), Prentice Hall.

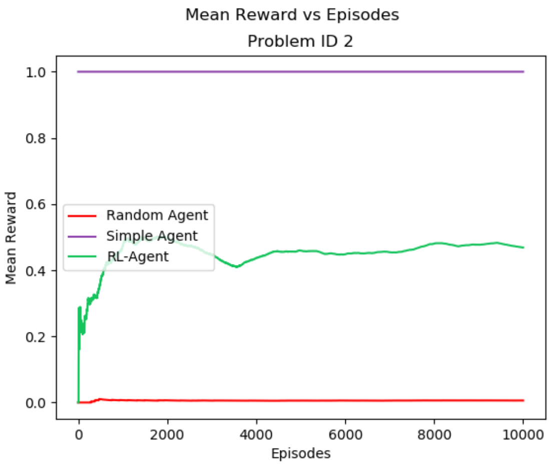
[2] Lecture and lab material from the course Artificial Intelligence H (CompSci 4004) at the University og Glasgow, 2019.

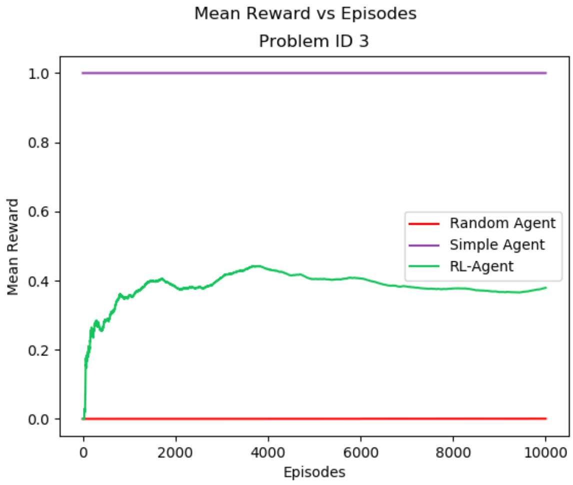
8 Appendix

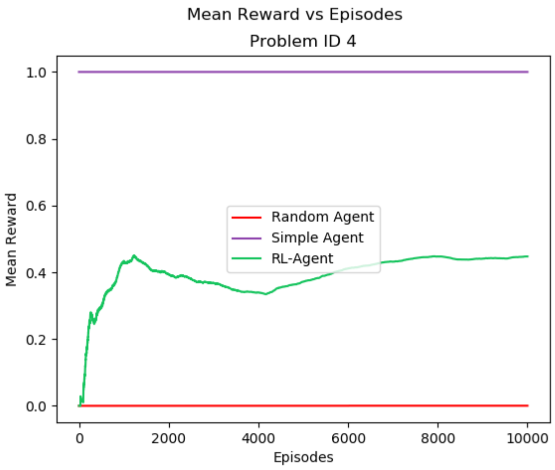
8.1 Visual Results Agent Comparison

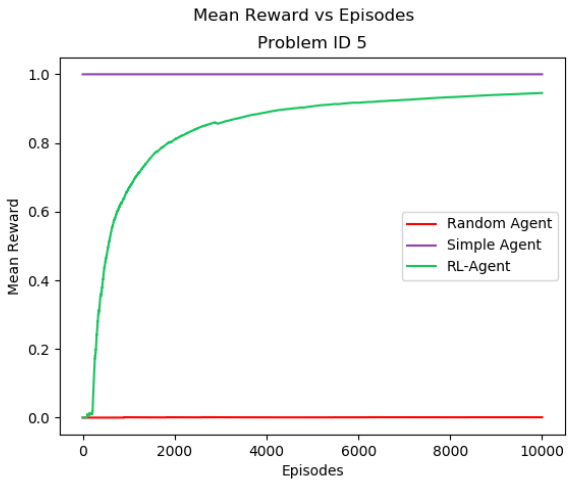


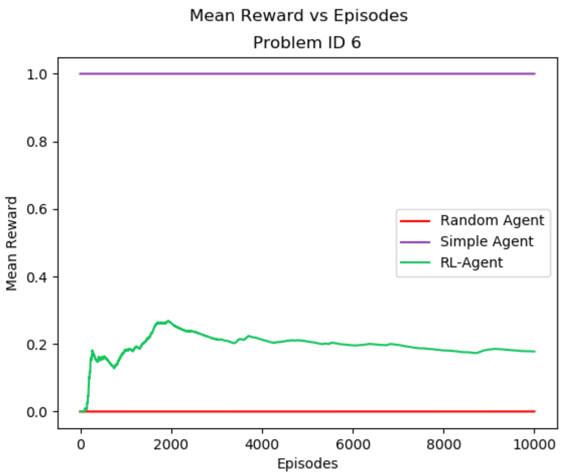


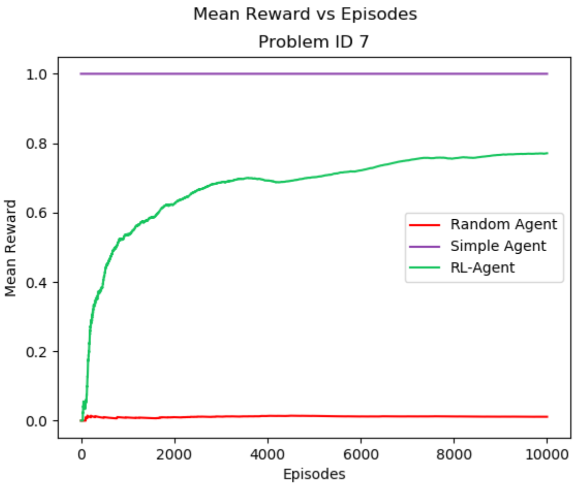




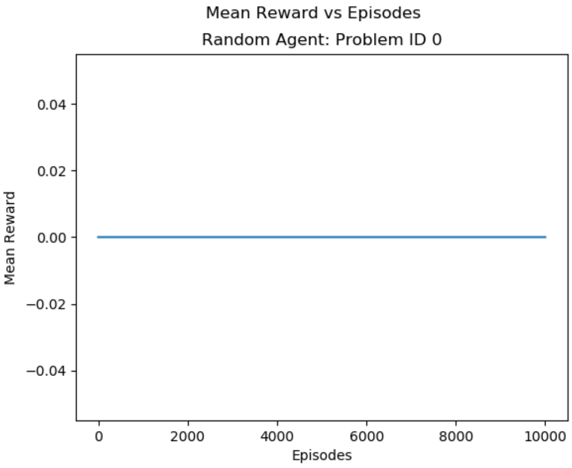


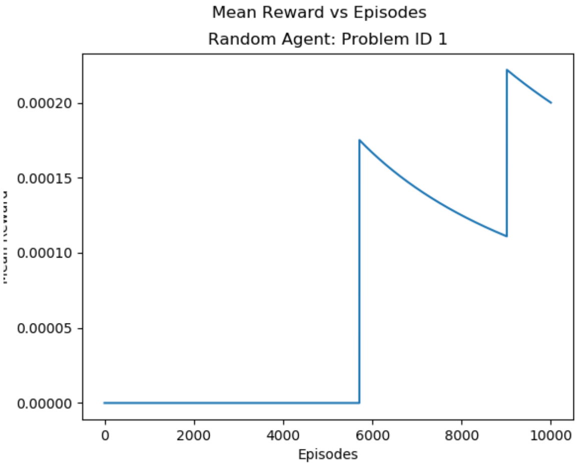


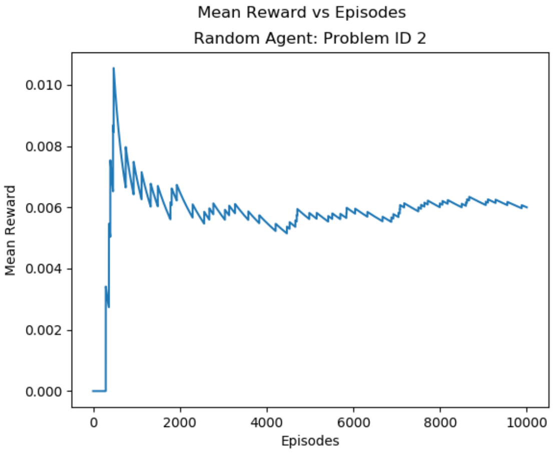


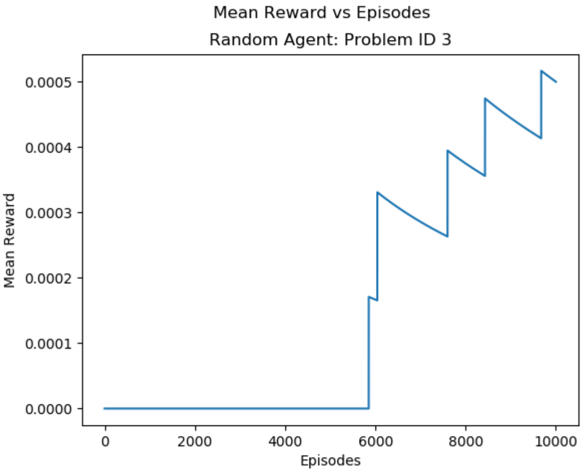


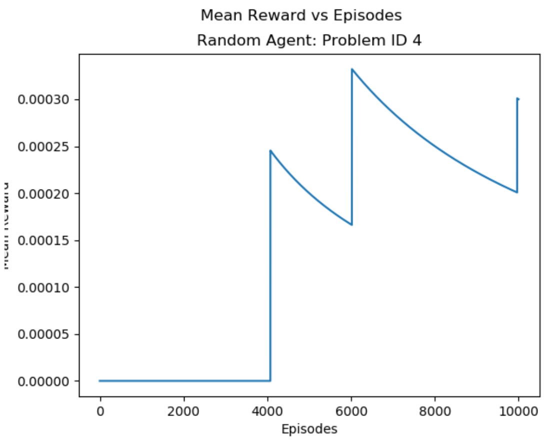
8.2 Visual Results Random Agent

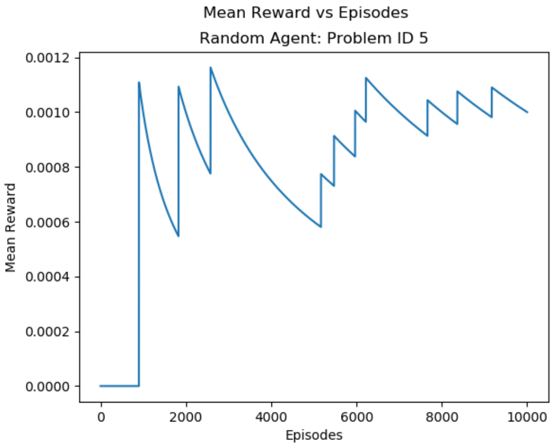


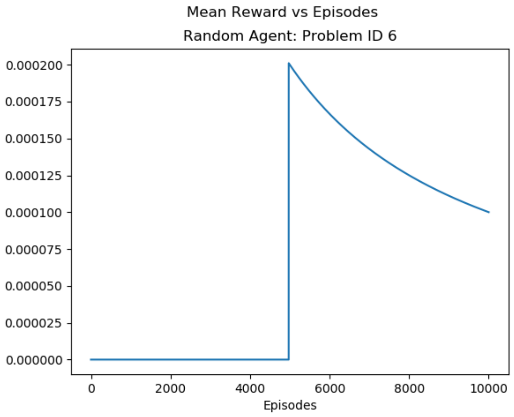


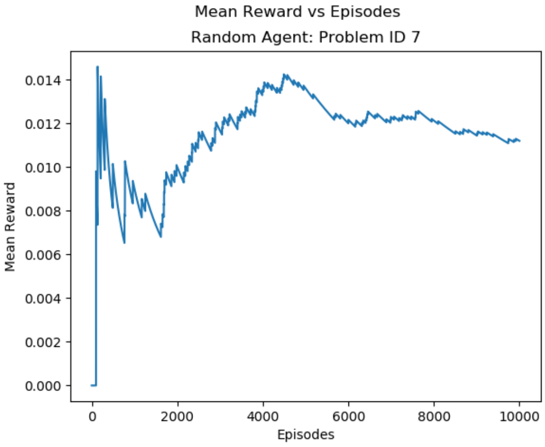




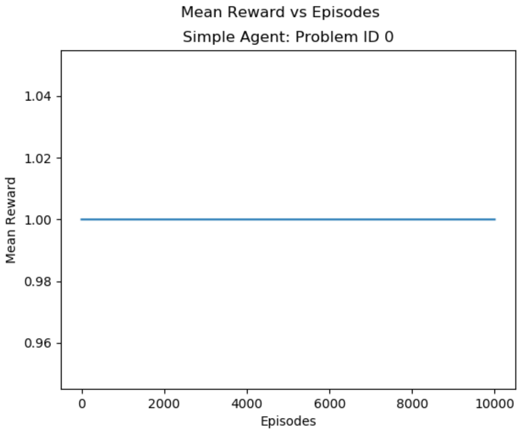


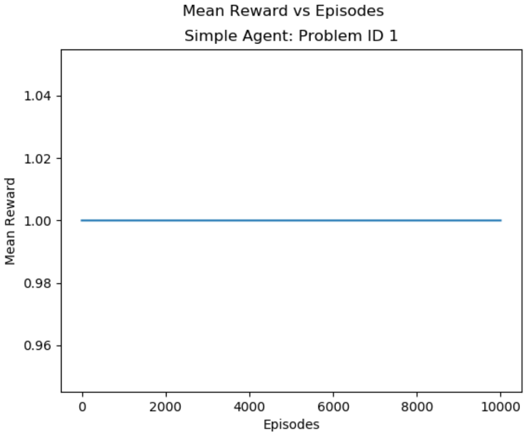






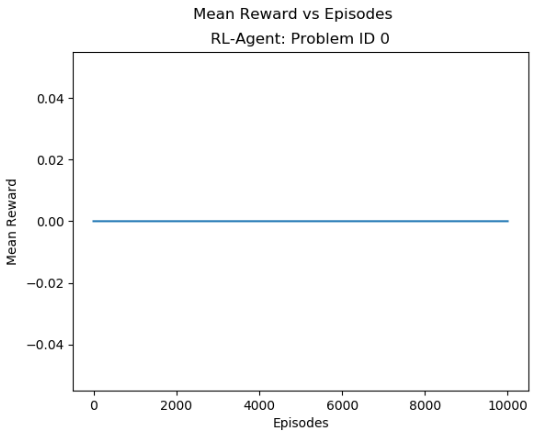
8.3 Visual Results Simple Agent

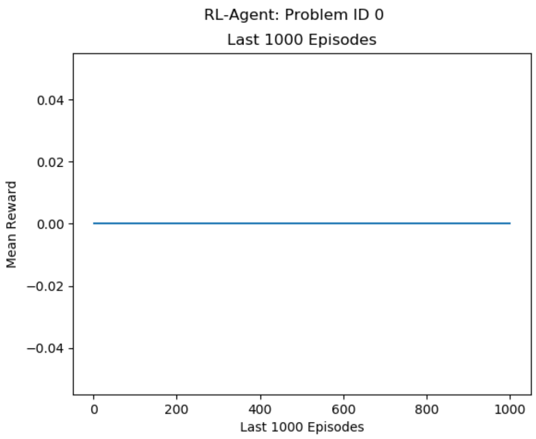


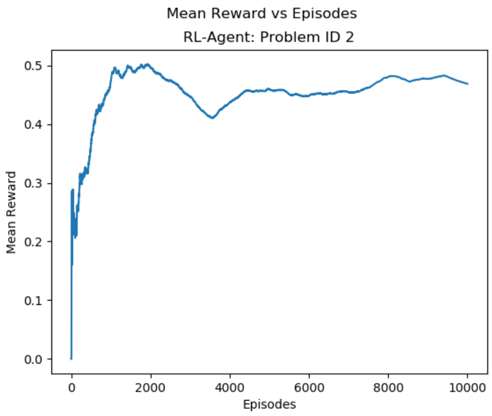


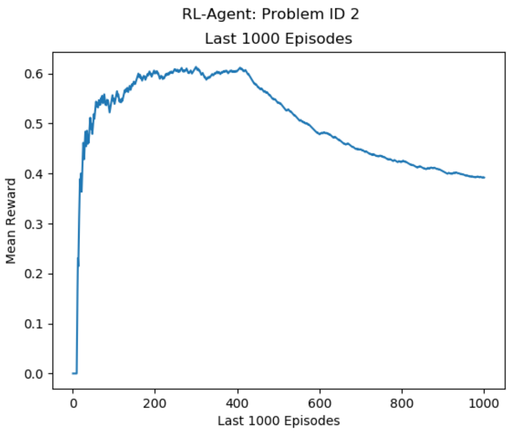
And so it continues.

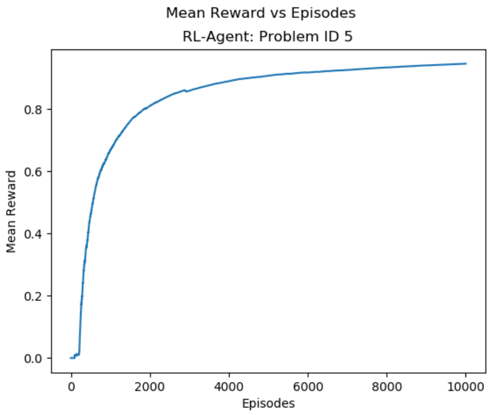
8.4 Visual Results RL-Agent

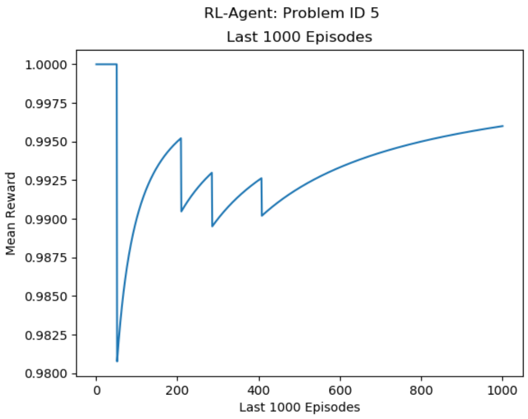


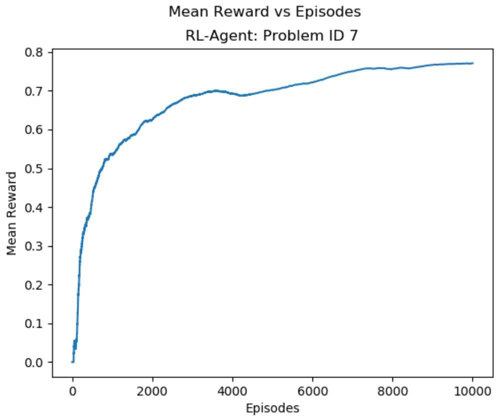


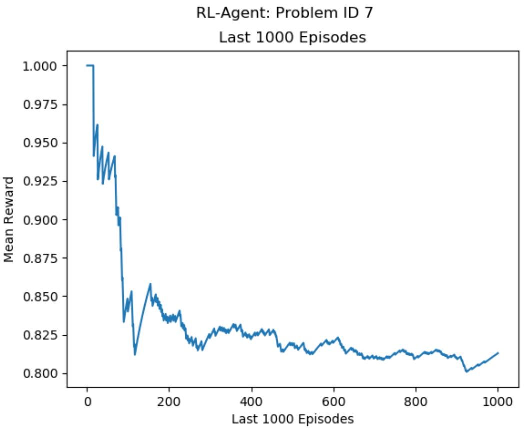












8.5 RL-Agent Policies for all Problem ID’s

Problem ID 1

.	S	>	v	<	.	>	>
v	^	^	^	^	v	>	^
<	v	<	.	>	^	>	>
<	<	^	^	<	.	>	<
<	^	^	.	>	v	^	^
<	.	.	>	^	<	.	>
<	.	v	^	.	>	.	>
<	v	<	.		G		.

Problem ID 2

.	>	S	v	<	.	>	<
v	^	^	^	^	v	>	>
<	<	<	.	>	^	>	<
<	^	>	v	<	.	>	<
<	^	<	.	>	v	^	^
<	.	.	v	^	<	.	>
<	.	>	^	.	<	.	>
G	v	<	.	>	<	<	.

Problem ID 3

.	>	<	S	<	.	>	<
v	^	^	^	^	v	^	>
<	<	<	.	>	^	>	<
<	^	<	v	<	.	>	<
<	^	<	.	>	v	^	^
<	.	.	>	^	<	.	>
<	.	>	<	.	<	.	<
v	v	G	.	>	>	<	.

Problem ID 4

	>	v	<	S	.	>	v
v	^	^	^	^	v	^	<
<	^	<	.	>	^	>	v
<	>	^	v	<	.	>	>
<	^	^	.	>	v	^	^
<	.	.	>	^	<	.	>
<	.	v	<	.	<	.	<
v	v	G	.	v	<	v	.

Problem ID 5

.	>	<	<	<	S	^	v
v	^	^	^	^	^	^	^
<	<	<	.	>	^	<	>
<	>	<	v	<	.	>	>
<	^	<	.	>	v	^	^
<	.	.	v	^	<	.	>
<	.	>	<	.	v	.	v
v	v	v	S	v	v	>	.

Problem ID 6

.	>	<	>	<	.	S	<
v	^	^	^	^	v	>	^
<	^	<	.	>	^	>	<
<	v	^	v	<	.	>	^
<	^	^	.	>	v	^	^
<	.	.	>	^	<	.	>
<	.	v	^	.	<	.	^
v	v	.	G	>	<	<	.