Artificial Intelligence: Assessed Exercise

COMPSCI4004: ARTIFICIAL INTELLIGENCE (H/M)

Shaun Loughery (2193422L)

2019

Table of Figures

[Equation 1 - Best-First Search 3](#_Toc26607677)

[Equation 2 - Q Learning Algorithm 3](#_Toc26607678)

Table of Contents

[Table of Figures i](#_Toc26607679)

[Table of Contents ii](#_Toc26607680)

[1 PEAS Analysis 1](#_Toc26607681)

[2 Agent Implementation and Design 3](#_Toc26607682)

[2.1 Random Agent 3](#_Toc26607683)

[2.2 Simple Agent 3](#_Toc26607684)

[2.3 Reinforced Learning Agent 3](#_Toc26607685)

[3 Results and Discussion 5](#_Toc26607686)

[3.1 Random Agent 5](#_Toc26607687)

[3.2 Simple Agent 6](#_Toc26607688)

[3.3 RL Agent 7](#_Toc26607689)

[3.4 Overall Comparison 7](#_Toc26607690)

[4 Conclusions 9](#_Toc26607691)

[References 10](#_Toc26607692)

[Appendices 11](#_Toc26607693)

[4.1 Appendix A – Setup 11](#_Toc26607694)

# PEAS Analysis

Within this project, three different types of agents were analysed. Before these agents can be implemented, however, they must be given some sort of performance measure. This is done through constructing a PEAS analysis. The analysis of each of these agents is found in Table 1, Table 2 and Table 3.

|  |  |
| --- | --- |
| **Agent** | Random |
| **Performance Measure** | Attempt to gain the largest amount of reward through random actions. |
| **Environment** | The environment is not observable. No prior knowledge about the locations of objects in the environment. The environment is stochastic and has only one agent at any time. The environment boundaries remain fixed. Once the environment is loaded, it remains static and does not change between steps. The rules of the environment remain constant. The environment is sequential because each run does not rely on previous or future runs. The environment remains discrete. The environment is an 8x8 grid with goal states marked as G (reward of 1), hole states marked as H (reward of -1) and all other states marked as F (reward of 0). |
| **Actuators** | The agent is controlled by a list of possible discrete actions: Up, Left, Right, Down |
| **Sensors** | None |

Table - PEAS Analysis of a Random Agent

|  |  |
| --- | --- |
| **Agent** | Simple |
| **Performance Measure** | Find a solution to the environment that results in a path from the beginning position to the goal position. |
| **Environment** | The environment is fully observable. The agent is always aware as to its current position in relation to the environment and the location of the goal. The environment is deterministic. The next state of the agent is not affected by stochastic elements in the environment and will always arrive to its expected next state. The environment is static. There are no changing elements of the environment. All walls remain fixed in their positions. There is only a single agent in the environment. There are no other agents. The environment is discrete. The environment is defined by a finite number of steps. There is no continuous list of possible states. The environment is known. The rules do not change. The environment is sequential because each run does not rely on previous or future runs. The environment is an 8x8 grid with goal states marked as G (reward of 1), hole states marked as H (reward of -1) and all other states marked as F (reward of 0). |
| **Actuators** | The agent is controlled by a list of possible discrete actions: Up, Left, Right, Down |
| **Sensors** | The agent is aware of all of locations of objects (holes and goal state). The agent has full awareness of its current state and position relative to the goal state. |

Table - PEAS Analysis of a Simple Agent

|  |  |
| --- | --- |
| **Agent** | Reinforced Learning |
| **Performance Measure** | Attempt to gain the largest amount of reward and provide value to different actions between different states. Attempt to achieve a higher success rate between runs. |
| **Environment** | The environment is not fully observable, the agent only has information regarding its current state and the actions in that state. The agent does not know its position relative to the goal states. The environment is stochastic and there is no guarantee the agent will arrive to its expected state. The environment does not change when it has been constructed. All walls remain fixed. There is only a single agent within the environment. The environment is discrete. The environment is defined by a finite number of steps. The rules do not change. The environment is sequential because each run does not rely on previous or future runs. The environment is an 8x8 grid with goal states marked as G (reward of 1), hole states marked as H (reward of -1) and all other states marked as F (reward of 0). |
| **Actuators** | The agent is controlled by a list of possible discrete actions: Up, Left, Right, Down |
| **Sensors** | The agent has perfect information about the current state it is in and the list of available actions within that state. The agent has no knowledge of the environment. |

Table - PEAS Analysis of a Reinforced Learning Agent

# Agent Implementation and Design

## Random Agent

The random agent does not need to follow any specific formula or rules. It is an uninformed search agent. The agent will take a random action in any states and can only determine the reward of the current state.

To implement the random agent, the environment is reset every episode that the agent is ran. The environment is running for a predetermines number of steps. Within each of these steps, the agent will randomly either move up, down, left or right. If the agent hits a hole or the goal state, the environment resets and the episode is finished. This repeats for a predetermined number of episodes.

## Simple Agent

The simple agent was implemented using the A\* algorithm. This agent is a fully informed agent that has complete information regarding its environment and the position of elements within that environment. This agent will calculate a solution path from the beginning state to the goal state. The environment for the simple agent is set to be not stochastic. Due to these conditions, the simple agent is a perfect baseline which will always return a positive reward and will never reach a hole state.

A type of best-first search algorithm is used for the A\*. When located at a node, any of the adjacent nodes are considered. This algorithm determines the most promising node to explore by using an evaluation function alongside any appropriate heuristics [1]. To calculate the value of each node, a combination of the cost to reach the node (g(n)) and the cos to get to goal from the node (h(n)) are used, shown in Equation 1 [2]. To implement this agent, the AIMA toolbox was used [3].

F(n) = g(n) + h(n)

Equation 1 - Best-First Search

## Reinforced Learning Agent

Q-learning is a type of reinforcement learning algorithm that determines what the best action to take given the current state is. It is an off policy algorithm because it changes its knowledge based on actions that are outside of the current policy and seeks to discover the policy that maximizes the total reward [4]. There are two ways the agent decides the next action to take: exploiting and exploring. The agent can initially explore the environment by taking pseudo-random actions or use the knowledge it has gained, stored in a Q table, to decide the best action using the ‘max’ function. The amount the agent should explore is defined using epsilon. As the agent explores more of the environment, the value of epsilon should decrease as it becomes more confident in its knowledge.

The equation used for the Q learning algorithm is in Equation 2. The value of α represents the learning rate, Q[state, action] represents the old value, Rt represents the reward, γ represents the discount factor and max(Q(S\_(t+1),action)) represents the estimate of optimal future values.

Equation 2 - Q Learning Algorithm

The reinforced learning agent consists of two stages: training and running. To implement the agent, these two stages must be implemented. The training stage runs first for many episodes. Due to the training method of the Q-learning algorithm, it can take thousands of episodes before the agent is able to reach the optimal policy. This equation is used for each step in each episode to update the Q-table values for each action in each state. Once training has been completed, the agent is run against a problem to try to determine the optimal path. As the environment is stochastic, it is extremely unlikely that a completely optimal policy can be reached. As there is no cost for remaining in the environment, the agent could simply move back and forth between spaces to avoid hole states and reduced reward. Therefore, for this agent, epsilon is manually set every 100 episodes to reduce the likelihood of an agent being stuck in a bad policy.

# Results and Discussion

Each agent was tested with 8 different problems (defined as problem 0 to 7) and their performance measures were taken.

The results were categorised as follows:

* For both the random and reinforced learning agent, a separate plot was created for each problem showing the number of episodes against the reward gained per episode
* For both the random and reinforced learning agent, a separate plot was created for each problem showing the number of steps against the reward for each episode
* For the simple agent, the number of iterations versus the problem was shown in a plot
* Tables were generated from each agent showing the success rate of the agent and the total reward received through all the iterations of the problem
* A table was generated that shows all these agents compared against their success rate and total reward

Across each of the problems, one of the easiest problems for the agents to tackle is problem two. Therefore, each agent will be evaluated based on their performance over the second problem with an overall table showing the performance of all agents in all problems.

## Random Agent

The random agent generally performed the worst amongst all the problems. The agent was successful in reaching the goal a few times, but in general did not perform rather well. The results of this agent are found in Figure 3‑1 and Figure 3‑2. The results also show that the more steps the agent had to take in the problem, the less likely it would find the goal state.

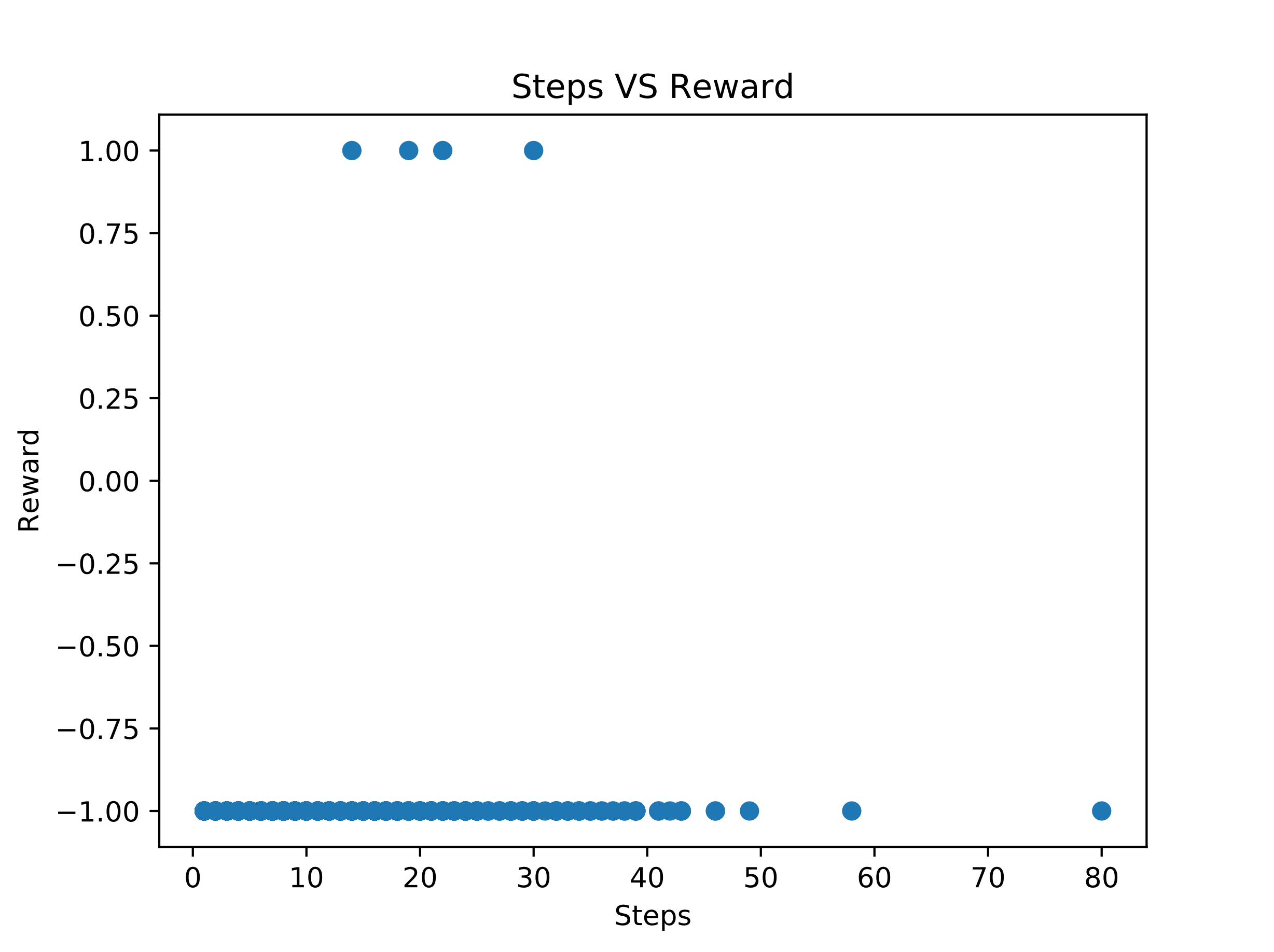


Figure ‑ - Random Agent Steps vs Reward

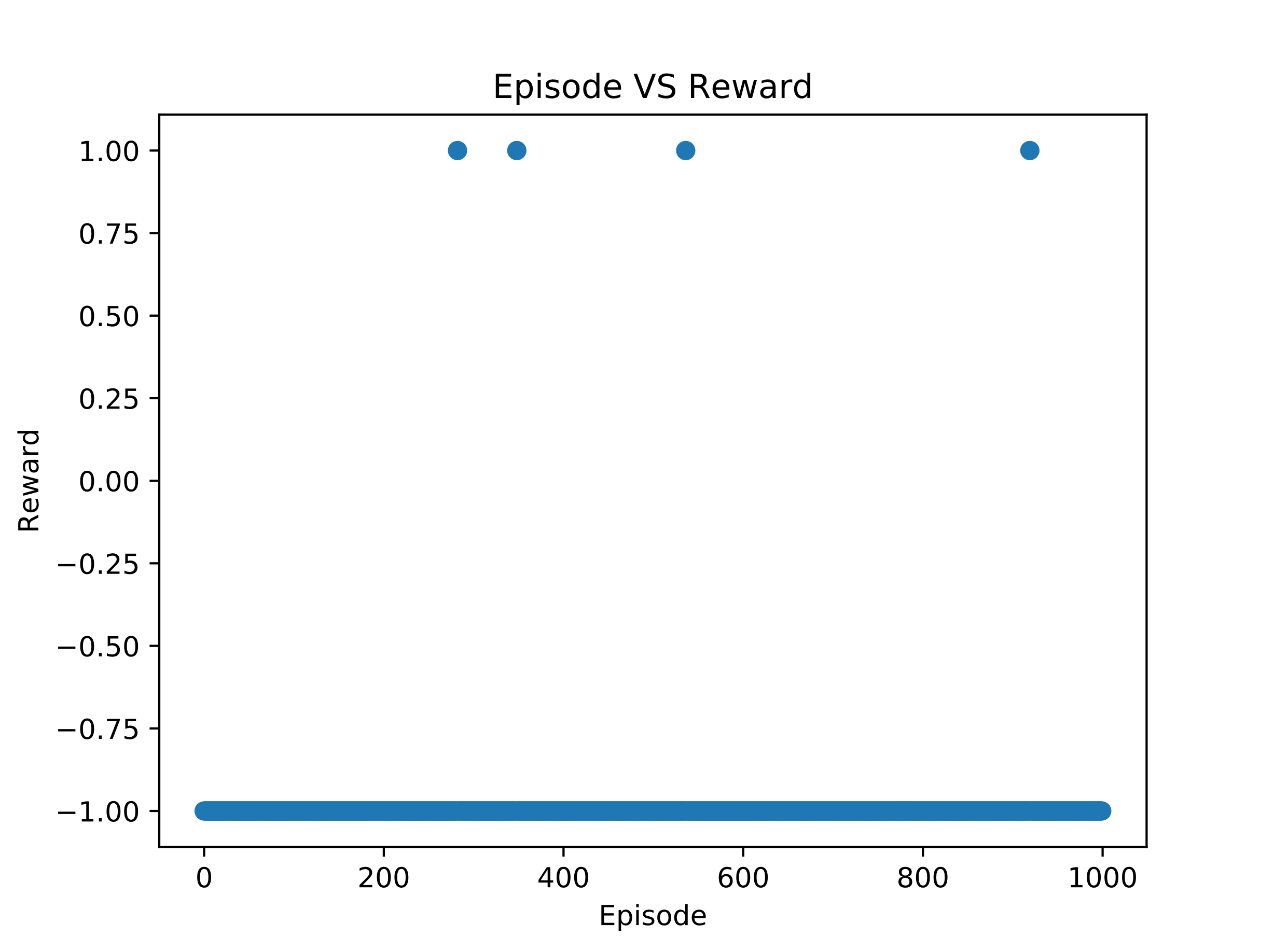


Figure ‑ - Random Agent Episode vs Reward

## Simple Agent

The simple agent was not compared against the same metrics. This is because the simple agent is an ‘ideal’ baseline. This agent will always find a solution from the starting position to the end goal. As this is the case, the reward will always be 1. However, it is interesting to note that the agent does not necessarily find the shortest or ‘optimal’ path. This is because there is no punishment for the agent remaining in the environment. The agent can take any path length, if it provides a solution to the goal state. Also, as this is an ideal baseline, it provides a good basis for how ‘difficult’ the problems are for the agents to solve (the shorter the number of iterations to reach the goal, the easier the problem theoretically is). The results of this agent are shown in Figure 3‑3. As the figure shows, it appears that problem two and seven are theoretically the easiest problems for the agents to solve, while problem 0 is theoretically the most difficult problem.

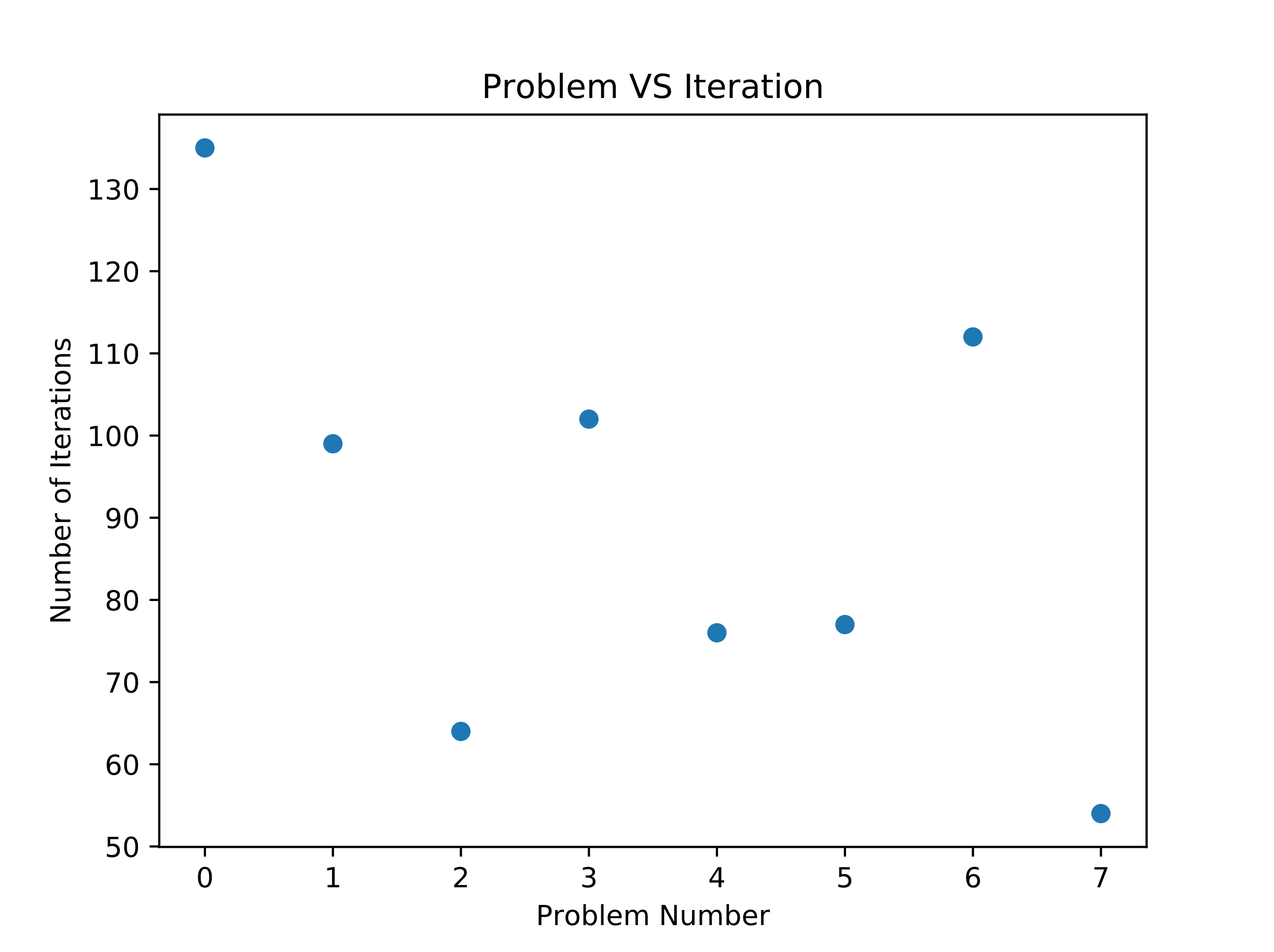


Figure ‑ - Simple Agent Problem vs Iterations

## RL Agent

The Q Learning agent performed far better than all other agents. The agent was able to reach the goal an increasing number of times. The agent typically did not require more than 70 steps to complete the problem. The agent was trained for 35000 episodes. The results are observed in Figure 4 and Figure 5.

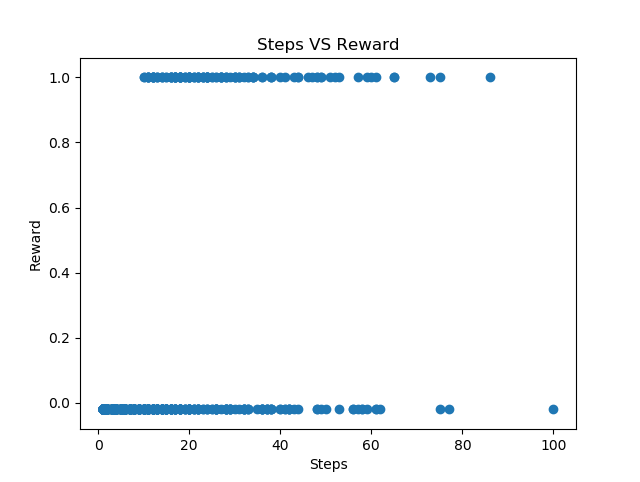


Figure - RL Steps VS Reward

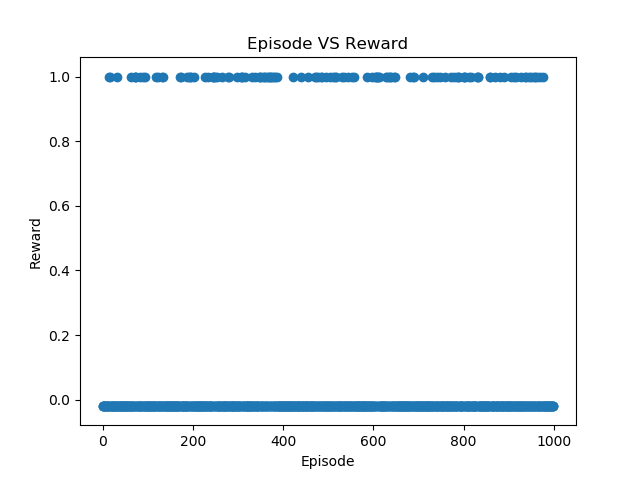


Figure - RL Episode VS Reward

## Overall Comparison

Comparing the agents overall, it is observed that the RL agent performs far better in finding the goal state than the random agent. The simple agent received the highest success rate and reward, but this is expected as the environment is known and not stochastic. The overall comparison of each agent is shown in Table 4. Although the RL agent returns the best result, it is also far slower than any other of the agents. There is no metric around how long the agent should take in the environment so for this application the time taken is not considered.

|  |  |  |  |
| --- | --- | --- | --- |
| **Agent** | **Problem** | **Success Rate** | **Total Reward** |
| Random Agent | 0 | 0.0 | -1000 |
| Random Agent | 1 | 0.0 | -1000 |
| Random Agent | 2 | 0.0 | -990 |
| Random Agent | 3 | 0.0 | -996 |
| Random Agent | 4 | 0.0 | -1000 |
| Random Agent | 5 | 0.0 | -1000 |
| Random Agent | 6 | 0.0 | -1000 |
| Random Agent | 7 | 0.0 | -998 |
| Simple Agent | 0 | 100 | 1 |
| Simple Agent | 1 | 100 | 1 |
| Simple Agent | 2 | 100 | 1 |
| Simple Agent | 3 | 100 | 1 |
| Simple Agent | 4 | 100 | 1 |
| Simple Agent | 5 | 100 | 1 |
| Simple Agent | 6 | 100 | 1 |
| Simple Agent | 7 | 100 | 1 |
| RL Agent | 0 | 0.0 | 0 |
| RL Agent | 1 | 0.0 | -20.00 |
| RL Agent | 2 | 12.5 | 107.50 |
| RL Agent | 3 | 1.6 | -3.680 |
| RL Agent | 4 | 0.0 | -20.0 |
| RL Agent | 5 | 0.0 | -20.0 |
| RL Agent | 6 | 0.0 | -20.0 |
| RL Agent | 7 | 0.0 | -20.0 |

Table - Comparison of All Agents on All Problems

# Conclusions

In conclusion, three agents were evaluated against each other when trying to solve the same problem. It is observed that the random agent performs poorly compared to other agents and that the simple agent performs exceptionally well due to the amount of available information to the agent.

References

|  |  |
| --- | --- |
| [1] | M. Raimann, “Best First Search (Informed Search),” GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/best-first-search-informed-search/. [Accessed 6 December 2019]. |
| [2] | R. Stuart and P. Norvig, Artificial Intelligence: A Modern Approach, Pearson, 2010. |
| [3] | D. Bacon, P. Ruggera, P. Shao, A. N. T. Patil, J. Martin and B. Catanzariti, “Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach",” GitHub, 3 December 2019. [Online]. Available: 6. [Accessed December 2019 2019]. |
| [4] | A. Violante, “Simple Reinforcement Learning: Q-learning,” Medium, 18 March 2019. [Online]. Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56. [Accessed 6 December 2019]. |
| [5] | magnarch, “A random, simple and Q-learning agent in OpenAI Gym's FrozenLake environment - Artificial Intelligence (H) Assessed Exercise,” GitHub, 29 May 2019. [Online]. Available: https://github.com/magnarch/AI-Exercise. [Accessed 6 December 2019]. |
| [6] | siomoninithomas, “Deep\_reinforcement\_learning\_Course/Q learning/FrozenLake/Q Learning with FrozenLake.ipynb,” GitHub, 11 August 2018. [Online]. Available: https://github.com/simoninithomas/Deep\_reinforcement\_learning\_Course/blob/master/Q%20learning/FrozenLake/Q%20Learning%20with%20FrozenLake.ipynb. [Accessed 6 December 2019]. |

Appendices

## Appendix A – Setup

To run the scripts, please first install the requirements by running the command:

Pip install -r requirements.txt

For run\_rl.py, run\_simple.py and run\_random.py, the following command structure must be used:

Python ./run\_(agent).py ProblemID MapID

Where ‘ProblemID’ is a number between 0 and 7 and MapID is either “4x4-base”, “8x8-base” and “16x16-base”.

To run the overall evaluation script, the following command is used:

Python ./run\_eval.py MapID

Where MapID is either “4x4-base”, “8x8-base” and “16x16-base”.