SC4003 Intelligent Agents Assignment 1

Amirul Hakim

March 18, 2025

1 Part 1 - Value Iteration and Policy Iteration

1.1 Method of value iteration

1.1.1 Description of Implemented Solution

Initialization: The environment is pass into the function by a class contains all the walls and rewards. A SummaryWriter object from tensorboardX is created for logging purposes. A utility values array V if initialized with zeros for each state in the environment.

Iteration: The update step in the value iteration algorithm is:

$$V_{k+1}(s) = max_a \sum_{s'} P(s'|s,a)[R(s,a,s'), +\gamma V_k(s')]$$

where $V_k(s)$ is the utility of state s at iteration k, max_a denotes the maximum value over all possible actions a, $P(s'|s,a)$ is the probability of transitioning to state s' from state s to state s' via action a, γ is the discount factor which prioritizes immediate rewards over distant rewards, $V_k(s')$ is the utility of state s' at iteration k.

Minimumly, it will run for 50 iterations while keep track of the maximum change in utility values δ. Once it falls below a specified threshold, this indicates convergence

Plotting: Plot the values and policies calculated from the above algorithm.

1.1.2 Plot of Optimal Policy

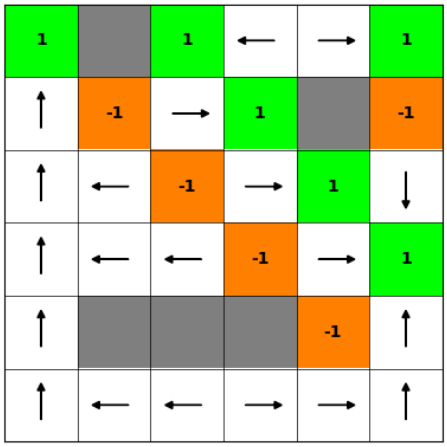


Figure 1: The optimal policy for the gridworld is illustrated, where arrows indicate the recommended action in each state. The color of the arrows reflects the value associated with each action.

1.1.3 Utility of all states

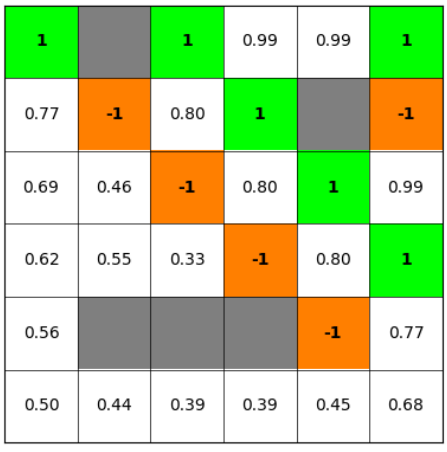


Figure 2: The utility values of all states within the gridworld are shown, with the color of each cell representing the state's value.

The more detailed value for the final utility is:

0.0	0.0	0.0	0.99445061	0.98753071	0.0
0.77247503	0.0	0.8	0.0	0.0	0.0
0.68529708	0.4611854	0.0	0.8	0.0	0.99445061
0.61840233	0.54986211	0.33239009	0.0	0.8	0.0
0.56078941	0.0	0.0	0.0	0.0	0.77247503
0.49686767	0.4406518	0.38504299	0.3947778	0.45040336	0.68411722

1.1.4 Convergence of value iteration

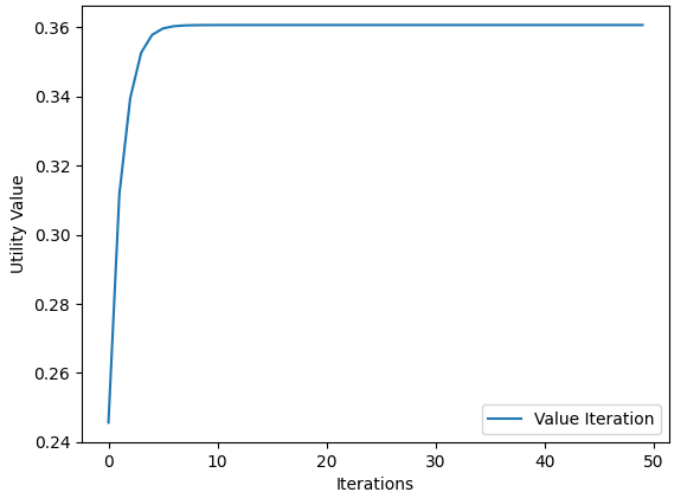


Figure 3: Convergence of value iteration. The y-axis represents the maximum change in utility values across all states. The x-axis represents the number of iterations.

1.2 Method of policy iteration

1.2.1 Description of Implemented Solution

Initialization: The setup is largely similar to value iteration, with the difference being that the initial policy assigns all actions to move right (0, 1).

Policy Evaluation: For every state, the expected utility is calculated according to the current policy's action, taking into account the transition probabilities to all possible next states and their associated rewards.

Policy iteration: The iteration consists of two steps: evaluation and improvement.
Policy Evaluation:

$$V^{\pi}(s) = \sum_{s',r} P(s', r|s, \pi(s))[r + \gamma V^{\pi}(s')]$$

where $V^{\pi}(s)$ is the utility of state s under policy π, $P(s', r|s, \pi(s))$ is the probability of transitioning to state s' with reward r from state s when following policy π. γ is the discount factor and $\pi(s)$ is the action prescribed by policy π in state s.

Policy Improvement:

$$\pi'(s) = argmax_a \sum_{s',r} P(s', r|s, a)[r + \gamma V^{\pi}(s')]$$

where $pi'(s)$ is the updated action for state s under the improved policy π', $argmax_a$ denotes the action that maximizes the expected utility.

1.2.2 Plot of Optimal Policy

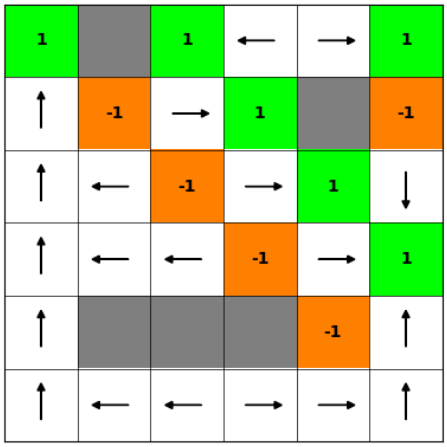


Figure 4: The optimal policy for the gridworld is illustrated, where arrows indicate the recommended action in each state. The color of the arrows reflects the value associated with each action.

1.2.3 Utility of all states

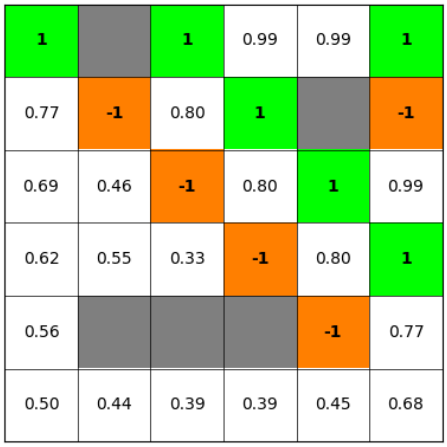


Figure 5: The utility values of all states within the gridworld are shown, with the color of each cell representing the state's value.

0.0	0.0	0.0	0.99445061	0.98753071	0.0
0.77247503	0.0	0.8	0.0	0.0	0.0
0.68529708	0.4611854	0.0	0.8	0.0	0.99445061
0.61840233	0.54986211	0.33239009	0.0	0.8	0.0
0.56078941	0.0	0.0	0.0	0.0	0.77247503
0.49686767	0.4406518	0.38504299	0.3947778	0.45040336	0.68411722

This indicates that, under the default maze configuration, both policy iteration and value iteration converge to identical utility values.

1.2.4 Convergence of policy iteration

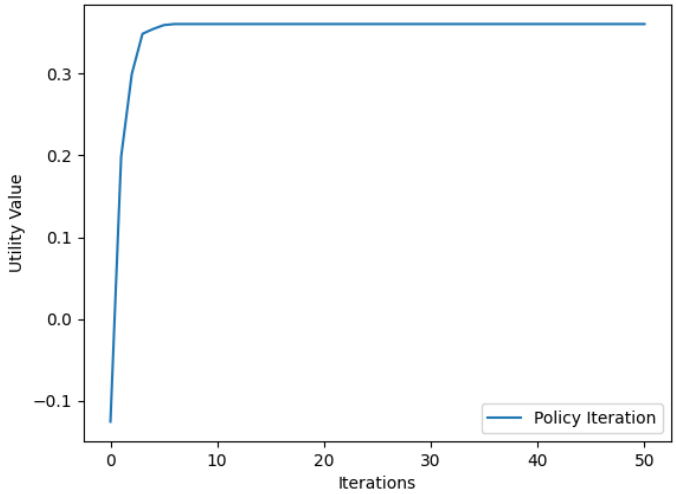
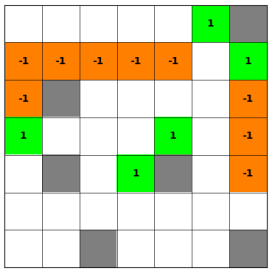


Figure 6: Convergence of policy iteration is illustrated, where the y-axis indicates the maximum change in utility values across all states, and the x-axis shows the number of iterations.

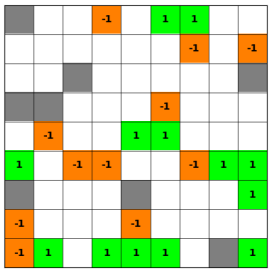
2 Part 2 - Complicated Maze Environment

2.1 Environment generation

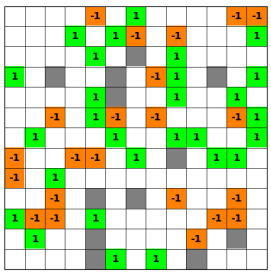
Randomly generated environment: The environment is created by randomly distributing walls and rewards within the gridworld. The user specifies the number of walls and rewards. Both elements are placed at random locations in the environment. The resulting configuration is saved as a YAML file in the config folder. Below are examples of the generated environments:



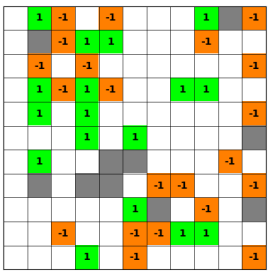
(a) Maze size 7



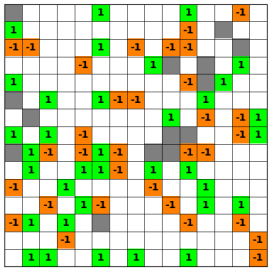
(b) Maze size 9



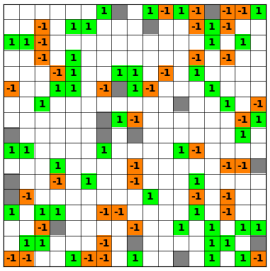
(a) Maze size 11



(b) Maze size 13



(a) Maze size 15



(b) Maze size 17

Figure 9: Randomly generated environments.

2.2 Comparison of value iteration and policy iteration

Convergence among different maze complexity: The convergence of value iteration and policy iteration for different maze size is shown in Figure 10 and Figure 11.

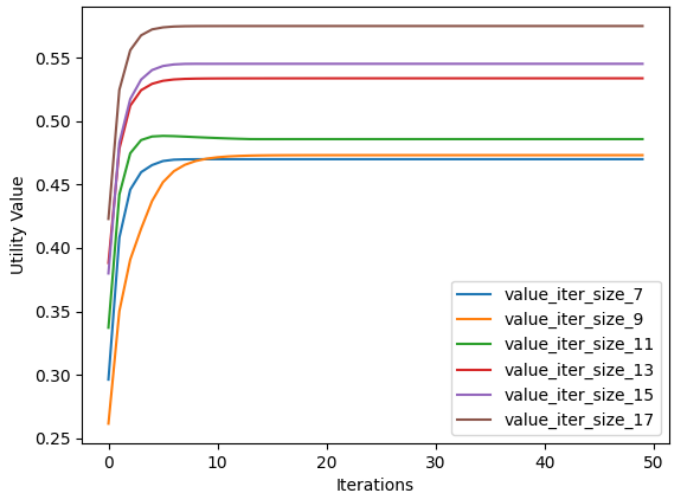


Figure 10: The value iteration curves for different maze size.

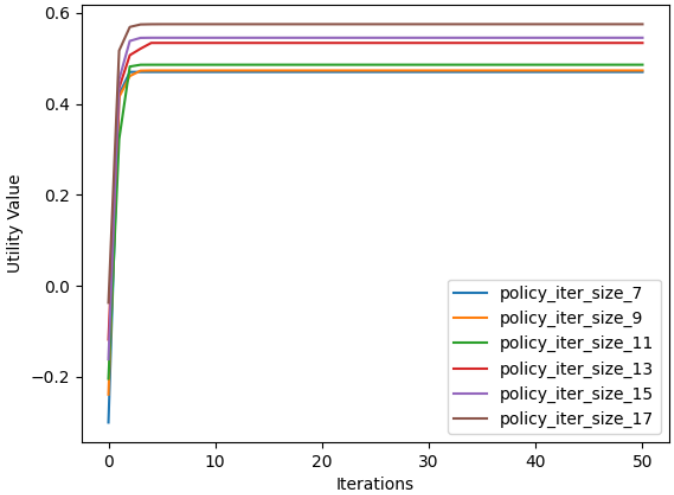


Figure 11: The policy iteration curves for different maze size.

This demonstrates that the complexity of the maze impacts the convergence of both value iteration and policy iteration. As the maze size increases, convergence becomes slower, and the final utility values tend to decrease.