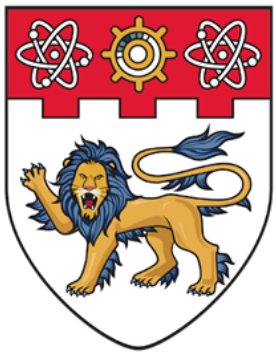

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CE 4046 Intelligent Agents

Assignment 1 Report

Agent Decision Making

Mervyn Chiong Jia Rong U1921023K

Content

Task 1: Optimal Policy

Q) Find the optimal policy and the utilities of all the (non-wall) states using both value iteration and policy iteration.

Firstly, I shall display the findings

Map used :

```
map = [['+', '#', '+', ' ', ' ', '+'],
       [' ', '-', ' ', '+', '#', '-'],
       [' ', ' ', '-', ' ', '+', ' '],
       [' ', ' ', ' ', '-', ' ', '+'],
       [' ', '#', '#', '#', '-', ' '],
       [' ', ' ', ' ', ' ', ' ', ' ']
       ]
```

With a starting position of (3,2)

1.1.1 Value iteration

number of iterations required: 916

---Utility for each state (State, Utility Scores)---

(0,0)	99.99
(0,1)	98.383
(0,2)	95.035
(0,3)	93.865
(0,4)	92.644
(0,5)	93.318
(1,0)	98.383
(1,1)	95.873
(1,2)	94.535
(1,3)	94.387

(1,4)	93.102
(1,5)	90.907
(2,0)	96.938
(2,1)	95.576
(2,2)	93.284
(2,3)	93.166
(2,4)	93.092
(2,5)	91.784
(3,0)	95.543
(3,1)	94.442
(3,2)	93.222
(3,3)	91.105
(3,4)	91.804
(3,5)	91.878
(4,0)	94.302
(4,1)	93.334
(4,2)	92.01
(4,3)	89.877

(4,4)	89.538
(4,5)	90.556
(5,0)	92.927
(5,1)	91.718
(5,2)	90.525
(5,3)	89.346
(5,4)	88.559
(5,5)	89.287

End result:

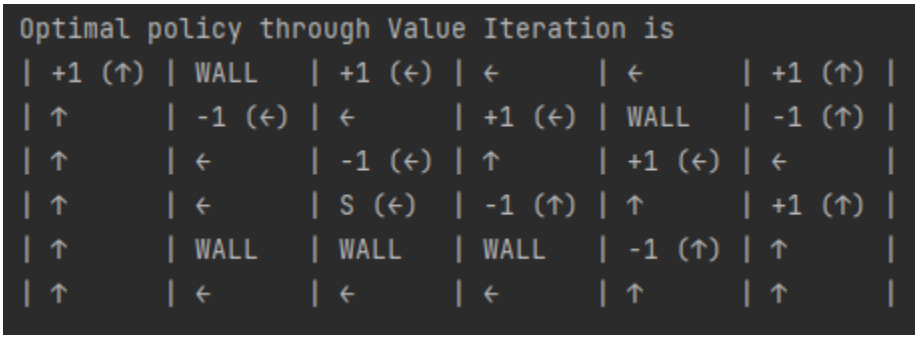

Fig 1.1.1 Plot of Optimal Policy Value iteration

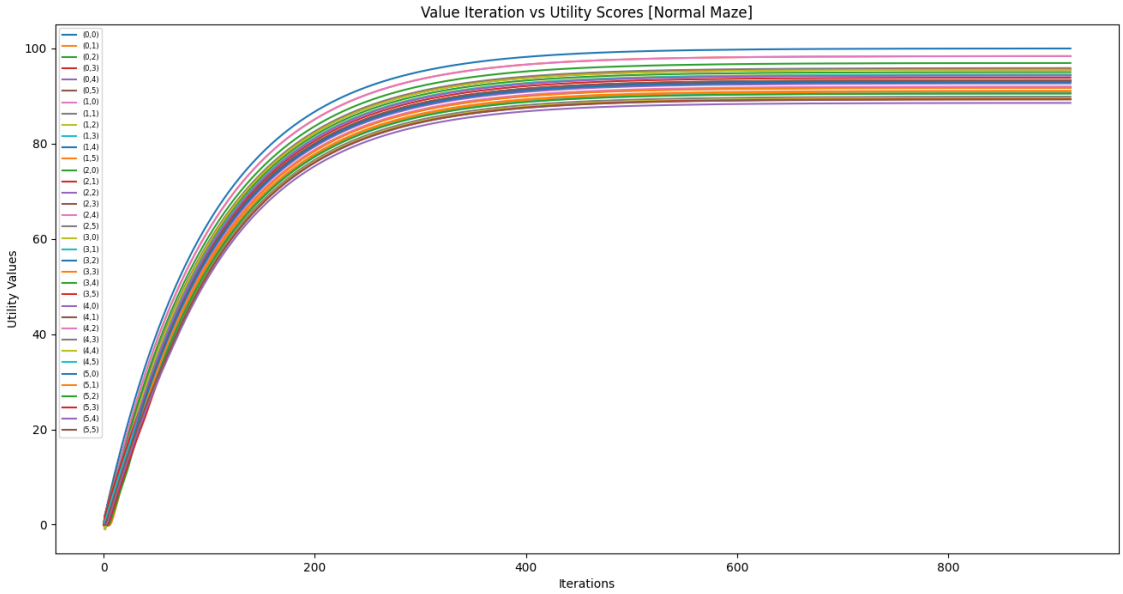

Fig 1.1.2 Graph of Iteration vs Utility scores for all squares [Value Iteration]

1.2.1 Policy iteration:

number of iterations required: 5
---Utility for each state (State, Utility Scores)----

(0,0)	99.999
(0,1)	98.393
(0,2)	95.045
(0,3)	93.874
(0,4)	92.654
(0,5)	93.328

(1,0)	98.393
(1,1)	95.883
(1,2)	94.544
(1,3)	94.397
(1,4)	93.112
(1,5)	90.917
(2,0)	96.948
(2,1)	95.586
(2,2)	93.294
(2,3)	93.176
(2,4)	93.102
(2,5)	91.794
(3,0)	95.553
(3,1)	94.452
(3,2)	93.232
(3,3)	91.115
(3,4)	91.814
(3,5)	91.888

(4,0)	94.312
(4,1)	93.344
(4,2)	92.02
(4,3)	89.887
(4,4)	89.548
(4,5)	90.566
(5,0)	92.937
(5,1)	91.728
(5,2)	90.535
(5,3)	89.356
(5,4)	88.569
(5,5)	89.297

End result:

```
Optimal policy through Policy Iteration is
| +1 (↑) | WALL   | +1 (←) | Left   | Left   | +1 (↑) |
| Up     | -1 (←) | Left   | +1 (←) | WALL   | -1 (↑) |
| Up     | Left   | -1 (←) | Up     | +1 (←) | Left   |
| Up     | Left   | S (←)  | -1 (↑) | Up     | +1 (↑) |
| Up     | WALL   | WALL   | WALL   | -1 (↑) | Up     |
| Up     | Left   | Left   | Left   | Up     | Up     |
```

Fig 1.2.1 Plot of Optimal Policy, Policy iteration

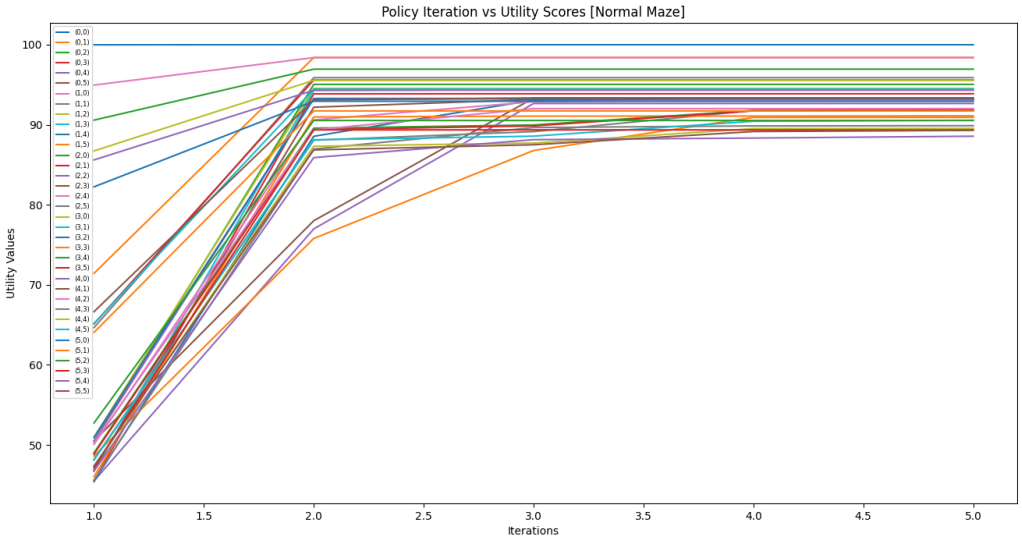

Fig 1.2.2 Graph of Iteration vs Utility scores for all squares [Policy Iteration]

It is quite evident that Policy iteration performs much better as compared to value iteration. Value iteration took 916 iterations and still gave utility scores of 0.01 lower when compared to policy iteration which took only 5 iterations.
Assuming that there are no errors, we shall pull up Value iteration at iteration 5 to see its performance.

(0,0)	5	5.851
(0,1)	5	4.265
(0,2)	5	5.345
(0,3)	5	4.269
(0,4)	5	4.153
(0,5)	5	5.308
(1,0)	5	4.265
(1,1)	5	1.823
(1,2)	5	4.024
(1,3)	5	4.908
(1,4)	5	3.968
(1,5)	5	2.957
(2,0)	5	2.888
(2,1)	5	1.668
(2,2)	5	1.908
(2,3)	5	3.75
(2,4)	5	4.864
(2,5)	5	3.913
(3,0)	5	1.664
(3,1)	5	0.79
(3,2)	5	0.962
(3,3)	5	1.746
(3,4)	5	3.719
(3,5)	5	4.826
(4,0)	5	0.716
(4,1)	5	0.133
(4,2)	5	0.124
(4,3)	5	0.875
(4,4)	5	1.729
(4,5)	5	3.571
(5,0)	5	0.071
(5,1)	5	-0.196
(5,2)	5	0.055
(5,3)	5	0.668
(5,4)	5	1.525
(5,5)	5	2.457

Fig 1.2.1 Value iteration values at iteration 5

This further confirms the initial hypothesis that Policy iteration performs much quicker than value iteration.

However, the answer for Q1 will be at the conclusion because I believe to find the optimal policy we require more criteria aside from iteration and utility scores to decide it and in part 2 we explore the effects of a more complicated maze.

1.3.1 Code Explanation

Value_iteration.py: Code to run value iteration, inside it has all respective functions to call and create a csv file.

Policy_iteration.py: Code to run policy iteration, inside it has all respective functions to call and create a csv file.

Setup.py: Predefined maze, complicated maze and the randomizer function to create a random user generated maze. Random function will generate it based of a ratio which is very similar to the example given. These files are then fed to the value and policy iteration files to be used.

Graph_csvs.py: To get the plots for the analysis. After the value and iteration files produce the csv files, we then are able to extract the data, plot it and save it

The remainder of the files are simply the csvs + png files created from the plotter and the iteration files

Part 2 Complicated Maze Exploration

Q How does the number of states and the complexity of the environment affect convergence?
Let us first show what needs to be changed to increase complexity. This will be showed in the explanations of the graphs and the theories we create

Q How complex can you make the environment and still be able to learn the right policy?
For now, we impose certain criteria that our maze should have and from these criteria the optimal policy should be the one with the least amount of iterations while still presenting an optimal solution.

2.1 Defining Complicated Maze

Starting positing is (9,0) [bottom left]
The created maze looks like this

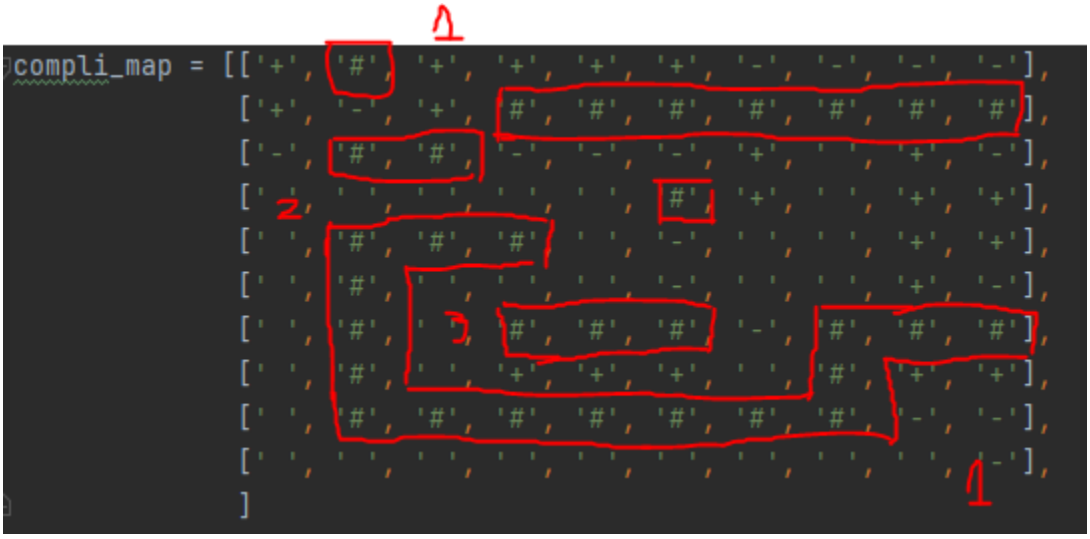

Fig 2.1 User created complicated maze

Complexity if [10x10] maze is based on the following reasons

1) Number of forks leading into dead ends or loops
    a) In this map, I have made 2 dead ends and also a loop in the middle, naturally a dead end is represented by negative reward values, evident in the top and bottom of the right side
2) Number of forks along the correct path
    a) I also implemented several forking paths from the user which ties into dead ends. In this case there is no actual “correct path”. The closest to a correct path would be to arrive at the middle of the right side
3) Length of incorrect paths

    a) Similar to dead ends just that they are meant to waste time or provide an incentive that is not worth it. As such i've created a loop in the middle, that simply is there to waste time with very mild incentive compared to going to the right side
4) Size of the maze
    a) The maze here is a 10x10 maze as such this will be explored using a different later on in the report

2.2 Value Iteration

Number of iterations required: 916
---Utility for each state (State, Utility Scores)----

(0,0)	99.99
(0,1)	98.561
(0,2)	99.99
(0,3)	99.99
(0,4)	99.99
(0,5)	99.99
(0,6)	97.496
(0,7)	95.033
(0,8)	92.601
(0,9)	90.2
(1,0)	99.716
(1,1)	97.496
(1,2)	99.99
(1,3)	98.835
(1,4)	98.693
(1,5)	98.693

(1,6)	96.684
(1,7)	96.642
(1,8)	97.876
(1,9)	96.056
(2,0)	97.225
(2,1)	96.339
(2,2)	97.813
(2,3)	90.681
(2,4)	92.929
(2,5)	95.487
(2,6)	97.955
(2,7)	97.913
(2,8)	99.163
(2,9)	97.319
(3,0)	95.809
(3,1)	94.565
(3,2)	93.336
(3,3)	91.963

(3,4)	91.954
(3,5)	96.454
(3,6)	98.085
(3,7)	98.211
(3,8)	99.539
(3,9)	99.58
(4,0)	94.565
(4,1)	93.471
(4,2)	94.595
(4,3)	93.343
(4,4)	93.165
(4,5)	94.521
(4,6)	97.018
(4,7)	98.211
(4,8)	99.539
(4,9)	99.58
(5,0)	93.336
(5,1)	94.595

(5,2)	95.84
(5,3)	94.595
(5,4)	93.343
(5,5)	94.196
(5,6)	96.607
(5,7)	97.913
(5,8)	99.163
(5,9)	97.319
(6,0)	92.122
(6,1)	95.994
(6,2)	97.256
(6,3)	98.535
(6,4)	98.693
(6,5)	98.387
(6,6)	95.914
(6,7)	96.562
(6,8)	98.693
(6,9)	98.693

(7,0)	90.923
(7,1)	97.256
(7,2)	98.535
(7,3)	99.99
(7,4)	99.99
(7,5)	99.99
(7,6)	98.387
(7,7)	99.99
(7,8)	99.99
(7,9)	99.99
(8,0)	89.739
(8,1)	88.46
(8,2)	97.256
(8,3)	98.693
(8,4)	98.693
(8,5)	98.693
(8,6)	97.111
(8,7)	96.065

(8,8)	97.496
(8,9)	97.496
(9,0)	88.46
(9,1)	87.566
(9,2)	88.722
(9,3)	89.893
(9,4)	91.079
(9,5)	92.279
(9,6)	93.495
(9,7)	94.726
(9,8)	95.973
(9,9)	95.136

End Result:

```
Optimal policy through Value Iteration is
| +1 (↑) | WALL   | +1 (↑) | +1 (↑) | +1 (↑) | +1 (←) | -1 (←) | -1 (←) | -1 (←) | -1 (←) |
| +1 (↑) | -1 (→) | +1 (→) | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   |
| -1 (↑) | WALL   | WALL   | -1 (→) | -1 (→) | -1 (→) | +1 (→) | →      | +1 (↓) | -1 (↓) |
| ↑      | ←      | ←      | ←      | ↓      | WALL   | +1 (→) | →      | +1 (→) | +1 (↓) |
| ↑      | WALL   | WALL   | WALL   | →      | -1 (→) | →      | →      | +1 (→) | +1 (↑) |
| ↑      | WALL   | ↓      | ←      | ←      | -1 (→) | →      | →      | +1 (↑) | -1 (↑) |
| ↑      | WALL   | ↓      | WALL   | WALL   | WALL   | -1 (↓) | WALL   | WALL   | WALL   |
| ↑      | WALL   | →      | +1 (→) | +1 (↑) | +1 (←) | ←      | WALL   | +1 (↑) | +1 (↑) |
| ↑      | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   | -1 (↑) | -1 (↑) |
| S (↑)  | →      | →      | →      | →      | →      | →      | →      | ↑      | -1 (↑) |
```

Fig 2.2.1 Outcome post Value iteration

Value iteration Shows no increase in total number of iterations required. Showing that despite an increase in complexity the run time it took to process the final result is ultimately the same as a 5x5 matrix

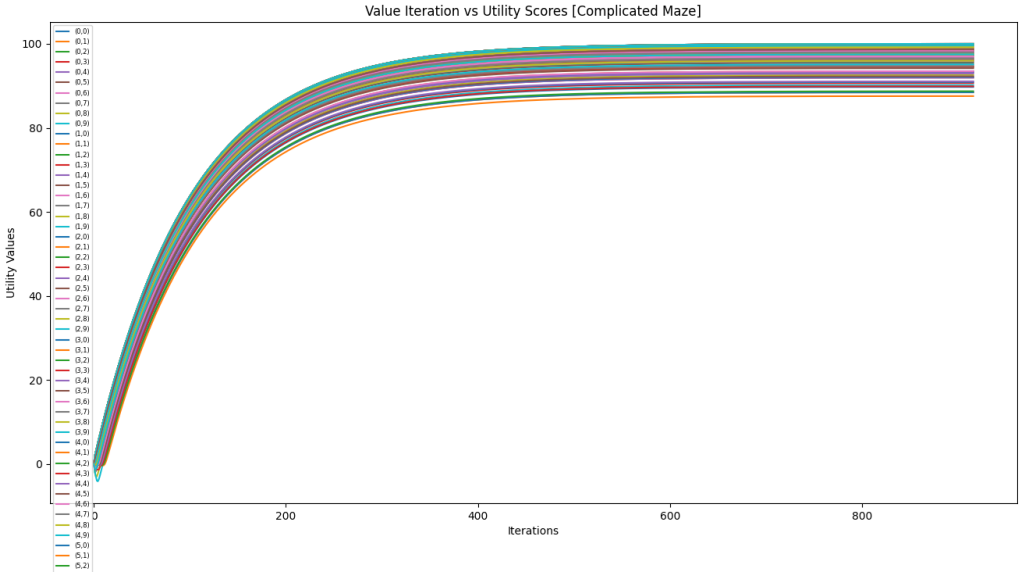

Fig 2.2.2 Graph of Iteration vs Utility scores for all squares [Value Iteration]

Utility values from Fig 2.2.2 are somewhat similar to Fig 1.1.2 in which they all reach a saturation after much more iterations. From this it is rather safe to conclude that there is no effect on value iteration regarding its convergence.

2.3.1 Policy iteration

Number of iterations required: 7
---Utility for each state (State, Utility Scores)----

(0,0)	99.999
(0,1)	98.571
(0,2)	99.999
(0,3)	99.999
(0,4)	99.999

(0,5)	99.999
(0,6)	97.506
(0,7)	95.043
(0,8)	92.611
(0,9)	90.209
(1,0)	99.725
(1,1)	97.506
(1,2)	99.999
(1,3)	98.845
(1,4)	98.703
(1,5)	98.703
(1,6)	96.686
(1,7)	96.645
(1,8)	97.879
(1,9)	96.058
(2,0)	97.235
(2,1)	96.349
(2,2)	97.822

(2,3)	90.685
(2,4)	92.932
(2,5)	95.489
(2,6)	97.958
(2,7)	97.916
(2,8)	99.165
(2,9)	97.322
(3,0)	95.819
(3,1)	94.575
(3,2)	93.345
(3,3)	91.973
(3,4)	91.958
(3,5)	96.457
(3,6)	98.088
(3,7)	98.214
(3,8)	99.542
(3,9)	99.583
(4,0)	94.575

(4,1)	93.481
(4,2)	94.605
(4,3)	93.353
(4,4)	93.168
(4,5)	94.524
(4,6)	97.02
(4,7)	98.214
(4,8)	99.542
(4,9)	99.583
(5,0)	93.345
(5,1)	94.605
(5,2)	95.85
(5,3)	94.605
(5,4)	93.353
(5,5)	94.199
(5,6)	96.611
(5,7)	97.916
(5,8)	99.165

(5,9)	97.322
(6,0)	92.132
(6,1)	96.004
(6,2)	97.266
(6,3)	98.545
(6,4)	98.703
(6,5)	98.397
(6,6)	95.924
(6,7)	96.566
(6,8)	98.703
(6,9)	98.703
(7,0)	90.933
(7,1)	97.266
(7,2)	98.545
(7,3)	99.999
(7,4)	99.999
(7,5)	99.999
(7,6)	98.397

(7,7)	99.999
(7,8)	99.999
(7,9)	99.999
(8,0)	89.749
(8,1)	88.47
(8,2)	97.266
(8,3)	98.703
(8,4)	98.703
(8,5)	98.703
(8,6)	97.121
(8,7)	96.075
(8,8)	97.506
(8,9)	97.506
(9,0)	88.47
(9,1)	87.576
(9,2)	88.732
(9,3)	89.903
(9,4)	91.089

(9,5) 92.289

(9,6) 93.505

(9,7) 94.736

(9,8) 95.983

(9,9) 95.146

End result:

```
Optimal policy through Policy Iteration is
| +1 (↑) | WALL   | +1 (↑) | +1 (→) | +1 (→) | +1 (←) | -1 (←) | -1 (←) | -1 (←) | -1 (←) |
| +1 (↑) | -1 (→) | +1 (→) | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   |
| -1 (↑) | WALL   | WALL   | -1 (→) | -1 (→) | -1 (→) | +1 (→) | →      | +1 (↓) | -1 (↓) |
| ↑      | ←      | ←      | ←      | ↓      | WALL   | +1 (→) | →      | +1 (→) | +1 (↓) |
| ↑      | WALL   | WALL   | WALL   | →      | -1 (→) | →      | →      | +1 (→) | +1 (↑) |
| ↑      | WALL   | ↓      | ←      | ←      | -1 (→) | →      | →      | +1 (↑) | -1 (↑) |
| ↑      | WALL   | ↓      | WALL   | WALL   | WALL   | -1 (↓) | WALL   | WALL   | WALL   |
| ↑      | WALL   | →      | +1 (→) | +1 (↑) | +1 (←) | ←      | WALL   | +1 (↑) | +1 (↑) |
| ↑      | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   | WALL   | -1 (↑) | -1 (↑) |
| S (↑)  | →      | →      | →      | →      | →      | →      | →      | ↑      | -1 (↑) |
```

Fig 2.3.1 Outcome post Policy iteration

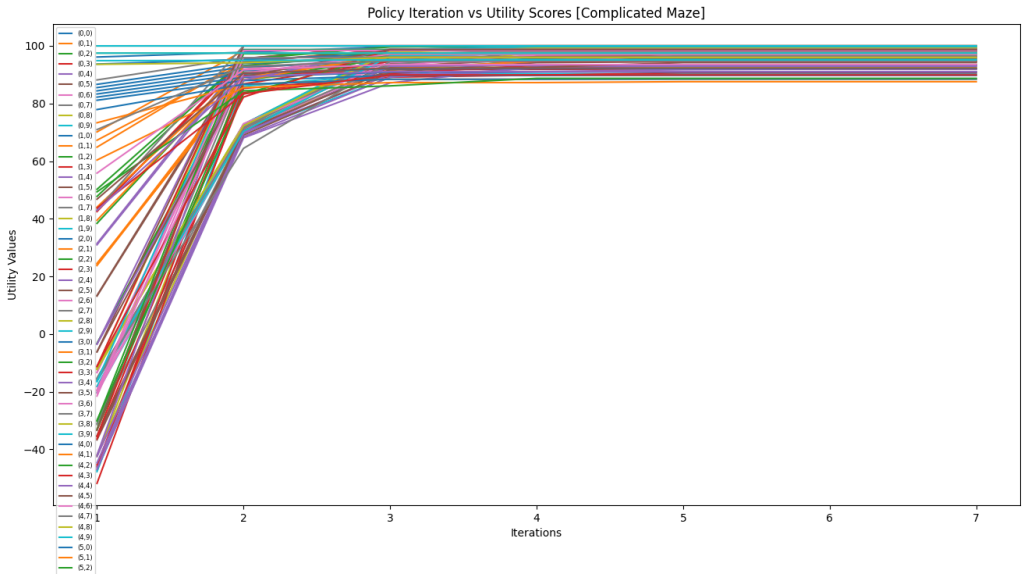

Fig 2.3.2 Graph of Iteration vs Utility scores for all squares [Policy Iteration]

From these 2 outcomes we can conclude certain findings:

- For Policy iteration, the change in number of iterations is as such: 5x5 maze [5] vs 10x10 [7]. We can hypothesize that potentially policy iteration is more susceptible to the increase in both size and complexity when compared to value iteration.
- Despite this it is safe to assume that Policy iteration still performs much better than Value iteration at the 7th iteration.

As such, I concur that Policy iteration's convergence is somewhat susceptible to minute changes in the environment such as an increase in complexity of the maze. But should still perform much better than Value iteration

2.4.1 Maze Exploration Effects with Size

Now to explore the effects of a larger maze on its complexity.

Because we are using a 30x30 the end result will be rather big it is stored in a txt file and as such I will only show the graphs. We can definitely use a much larger size, but the results should not change too much.

Value iteration:
[Random_maze_analysis_value_it.csv]
Iterations: 916

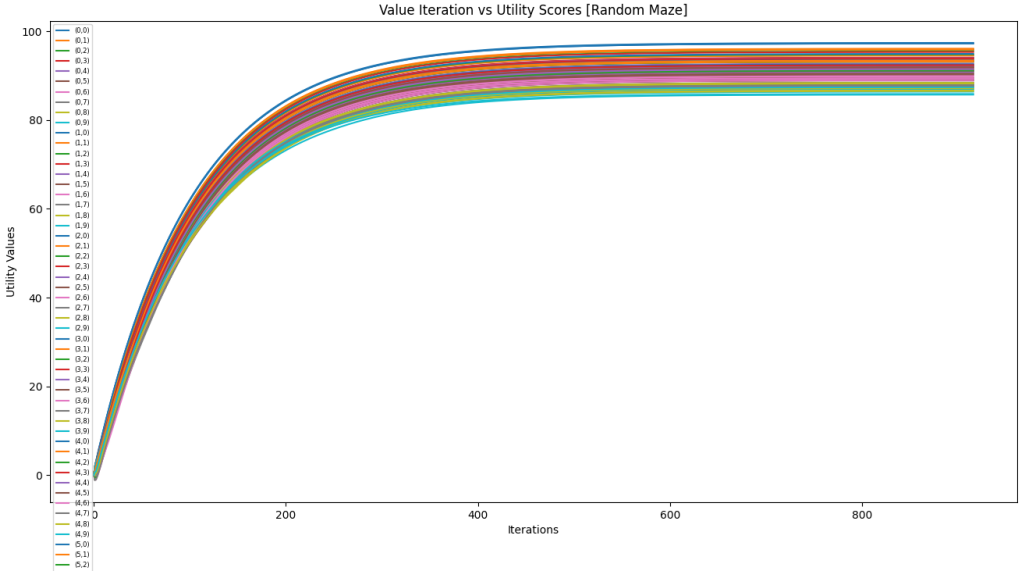

Fig 2.4.1 Graph of Iteration vs Utility scores for all squares [Value Iteration]

Size did not increase nor change anything of significant value for it. Therefore, we can conclude that value iteration will always perform the same regardless of the complexity or size

Policy iteration
[Random_maze_analysis_policy_it.csv]
Iterations: 12

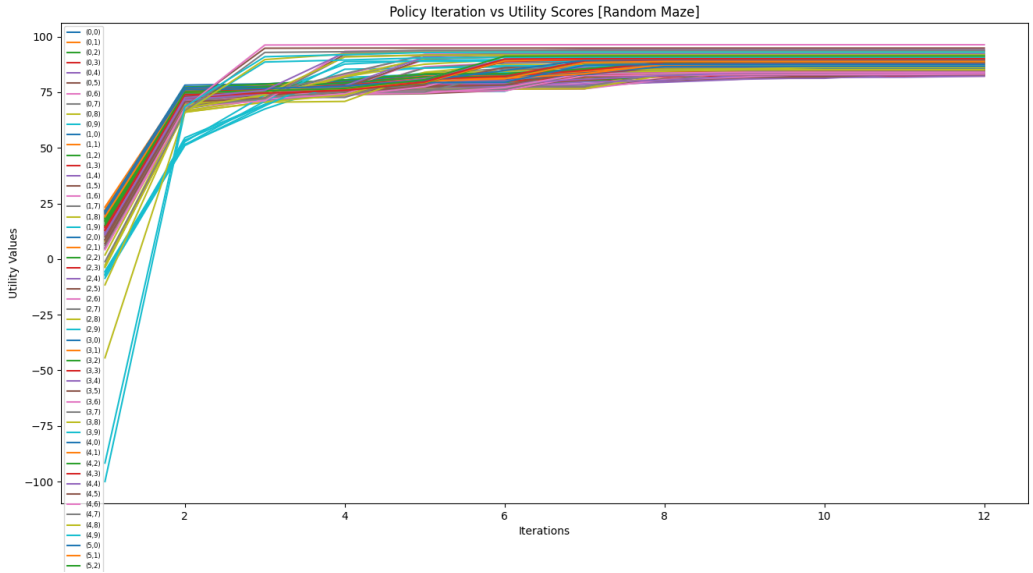

Fig 2.4.2 Graph of Iteration vs Utility scores for all squares [Policy Iteration]

Observations:

- Policy Iteration increased from 7 at 10x10, barring other complexity changes, to 12 and 30x30
- Aside from this, another hypothesis regarding the convergent point, the point at which all states converge, is also confirmed. From Fig 2.4.2 and Fig 2.3.2 we can see that there are multiple states that converge at different iteration values. As such, it is possible that this will affect its convergent point. This would also mean that it is more susceptible to an increase in complexity. Which is evident in Fig 2.4.2
    - This ties in with our earlier theory that policy iteration's convergence is dependent on complexity thereby confirming it.
- Additionally, it is rather clear that the size of the maze clearly affects the convergent point for both policies, but Policy iteration is still much more effective than Value iteration at the same iteration count.

Conclusion

From the results it is quite evident to me that Policy iteration in this scenario performs much better in most cases. Not only in iteration count but just in general when handling a larger scale maze up to a certain point. I believe that there will come a point where Policy iteration will be better, but I believe that the only case that happens is when an extremely big maze, say around 1000x1000 with an equal increase in complexity in it. Such as, adding multiple dead ends, loops, and more negative rewards. Will Value iteration be able to be more efficient than policy but even so, it should produce the same results. As such, Policy Iteration is much better than Value Iteration.