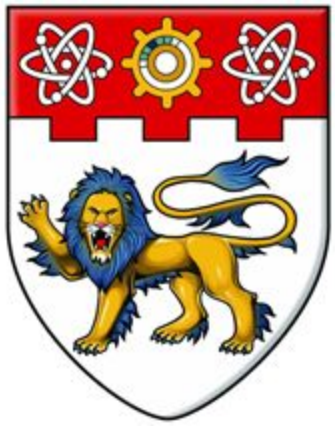


NANYANG TECHNOLOGICAL UNIVERSITY

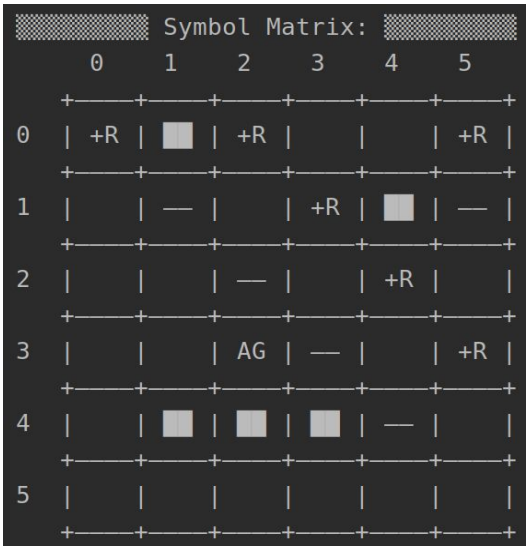
Assignment 1 : Agent Decision Making

CZ4046 – INTELLIGENT AGENTS
(Semester 2, AY 2019/2020)

WILSON THURMAN TENG

1. Overview

1.1 Problem Description



In this assignment, two algorithms, namely “Value Iteration” & “Policy Iteration” are introduced to find an optimal policy for a given gridworld environment.

The environment is a 6x6 gridworld with reward, penalty, wall, as well as start tiles. The transition model depicts that the agent will have a 0.8 probability of going in it’s intended direction and 0.1 probability of going in the clockwise as well as anti-clockwise direction.

To conclude, the environment has a discrete state space and is stochastic in nature.

1.2 Optimal Policy

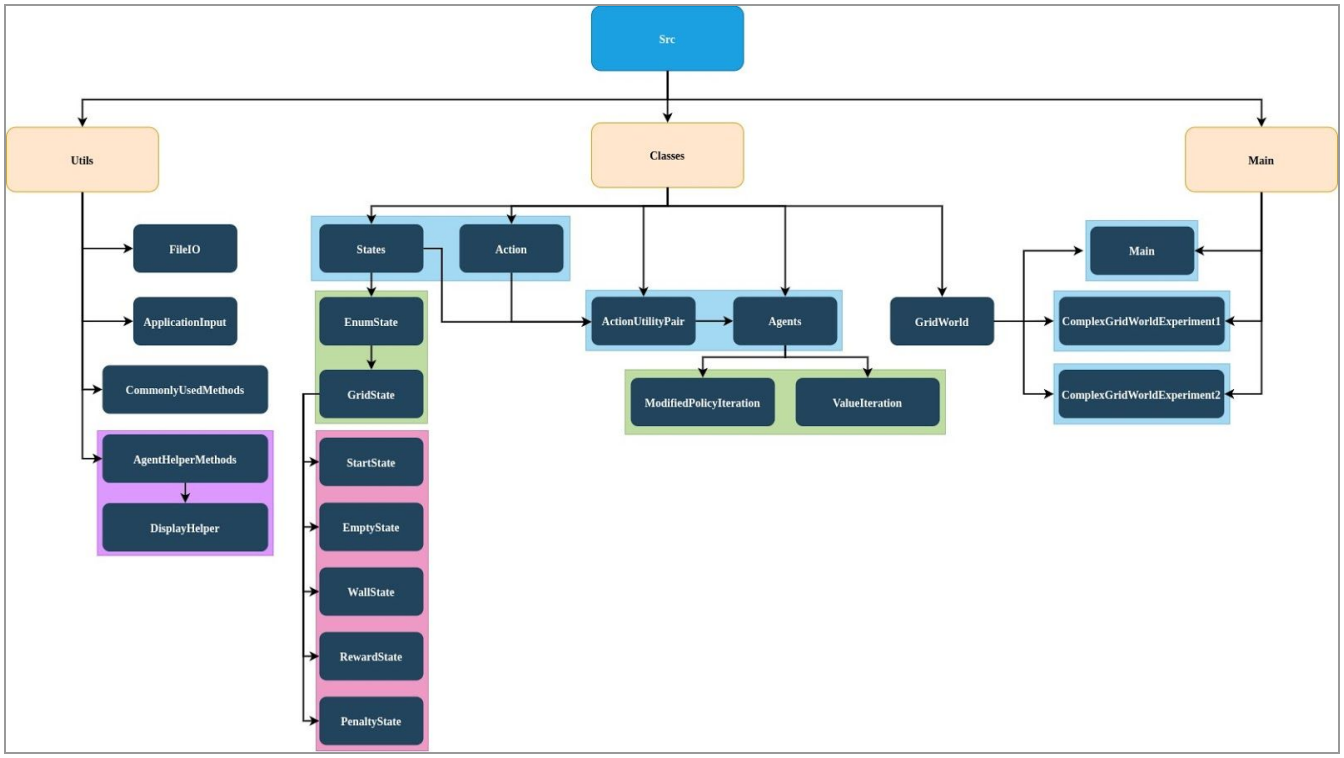
In this problem, the optimal policy would be to get to state (0,0) as fast as possible and move upwards. Moving upwards will result in the agent remaining stationary as the clockwise and anti-clockwise directions, left and right respectively when the agent is in (0,0) is unvisitable. This will cause the agent to stay in (0,0) indefinitely and hence accumulate the most rewards.

1.3 File Organisation

Important files to take note of:

- Src folder :
  > Contains the code implementation to solve the given problem description.
- Results folder :
  > Contains the outputs obtained from running the executable files in the src folder.
- *Interactive Graph Results.html* :
  > Contains interactive graph plots of the state utility values against number of iterations.
  > Toggle the *"Compare Data on hover"* as well as the *"Toggle spike lines"* options for a more interactive session with the graph.
- Experiments.pdf
  > Contains experimentation on discount values to obtain optimal policy.
- *CZ4046 Assignment 1 Class Diagram.jpg* :
  > Simplified Class diagram of code implementation.

1.4 Code Implementation



The java code is split into 3 main packages:

- Utils
- Classes
- Main

1.4.1 Utils Package

The **Utils package** contains 4 files.

1) *FileIO.java* manages the saving of State Utility values.
2) *ApplicationInput.java* stores constant values required by the two aforementioned algorithms. (E.g. Discount, Rmax, C, K, etc)
3) *CommonlyUsedMethods.java* contains common methods. (E.g. Generating random integers values, etc)
4) *DisplayHelper.java* under AgentHelperFunction Package contains useful methods to help evaluate the policies generated.

1.4.2 Classes Package

The **Classes Package** contains the bulk of the agent's logic and keeps track of the utility values of states. To do this efficiently, it is further divided into 5 separate parts:

1) States Package:
    a) Constraints GridWorld.java to a finite number of state types.
2) *Action.java*:
    a) Keeps track of the transition state probabilities as well as the actions available to the agent.
3) *ActionUtilityPair.java*:
    a) Derived from States and Action.java. Used to keep track of the policy as well as utility value of a given state.
4) Agents Package:
    a) Contains the logic for the 2 algorithms which will be explored further in later sections.
5) *GridWorld.java*
    a) Stipulates the environment that the agent operates in.

1.4.3 Main Package

The **Main Package** contains 3 files:

1) *Main.java*
    a) The executable file for the original gridworld in this assignment.
2) *ComplexGridWorldExperiment1.java*
3) *ComplexGridWorldExperiment2.java*

The two ComplexGridWorldExperiment.java files will be explored further in the *BonusQuestion.java* section.

2. Value Iteration

2.1 Description

A set of constants are first instantiated in *ApplicationInput.java*. The important constants to take note of are **c** and **Rmax**. These constants are used to calculate **Epsilon** for Value Iteration (Epsilon = c*Rmax). Epsilon can be interpreted as the maximum error allowable for a given state before accounting for the discount factor.

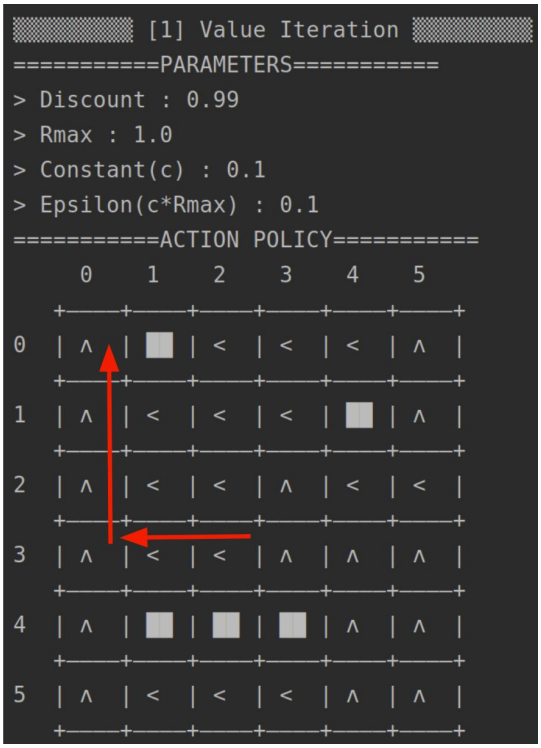
A GridWorld is then instantiated with states which contain reward values, as well as whether the state is visitable or not. (E.g. Wall is not visitable).

The *ValueIteration.java* class is called and the MDP representation, the GridWorld object are passed together with the epsilon value and discount value. Every state is then replaced with a StateActionUtilityPair. This is to keep track of the action with the highest utility for a given state.

An initial policy with no actions and 0 utility for every state is then initialised and Bellman Update will then be performed on state utility values. A bellman update is useful as it allows the value of a state to be represented by the values of other states. This bellman update continues until the maximum change in utility values amongst all the states in that particular iteration is less than the convergence criteria. The convergence criteria is calculated by: Epsilon * ((1-discount) / discount).

Finally, an array of StateActionUtilityPairs where the state's highest utility action along with its utility value is obtained.

2.2 Plot of Optimal Policy

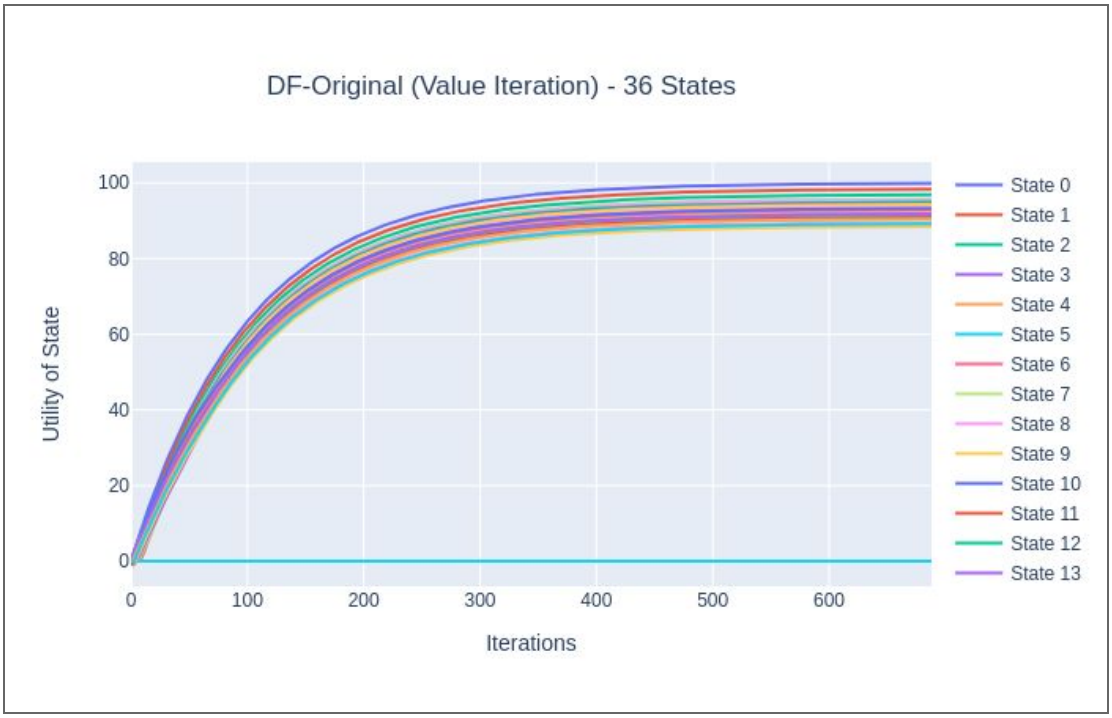


In the above image, it is clear that the agent has figured out the most optimal policy as explained in Section 1.2, which is to reach (0,0).

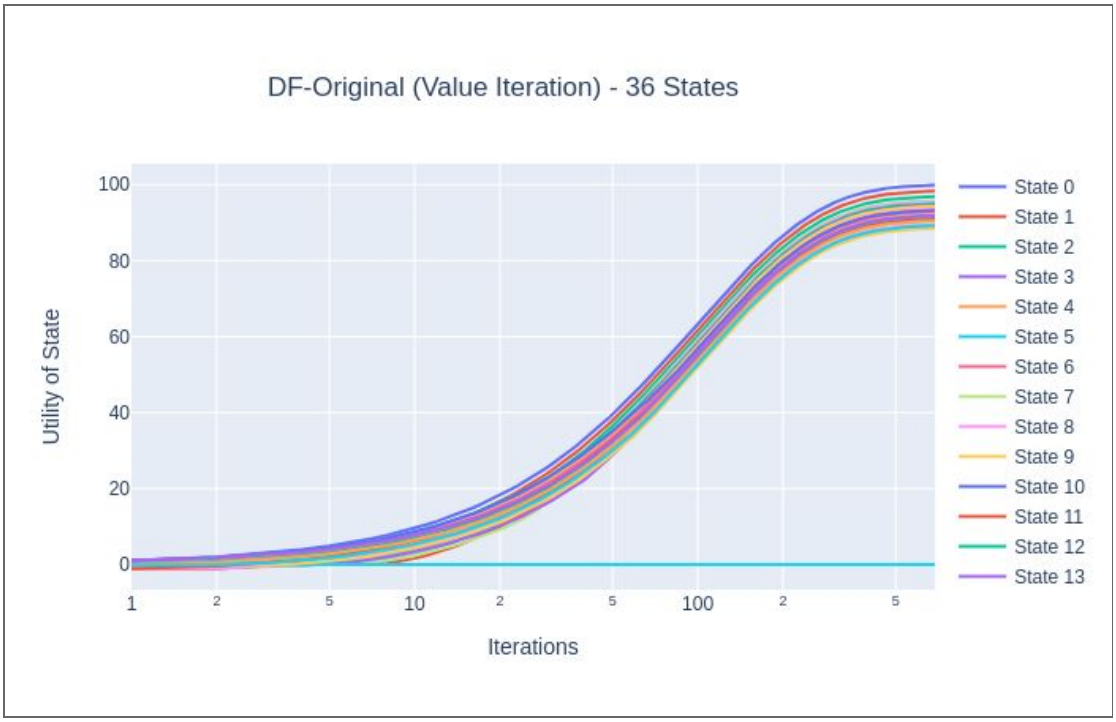
2.3 Utilities of all states

Refer to Appendix.

2.4 Plot of Utility Estimates as a function of the number of iterations



Value Iteration Graph from "Interactive Graph Results.html"



Value Iteration Graph with Log(Number of Iterations) from "Interactive Graph Results.html"

2.5 Value Iteration Findings

In this implementation, a total of 688 iterations are required to reach convergence.

Experimentation with c or Rmax parameters yielded no significant change in results. The optimal policy was still found, albeit at a faster or slower pace depending on whether epsilon was increased or decreased respectively.

In this graph, we observe that the relative positions in utility values of the different states stay the same after about iteration 200 which highlights that the optimal policy could have been reached much earlier but because of the epsilon value, the number of iterations is already pre-set.

Another interesting observation is that the utility values plotted after iteration 200 are very similar for most states and increase at roughly the same rate. This implies that the utility values for these states may be overestimated as the number of iterations increase.

3. Policy Iteration

3.1 Description

As discussed in the previous section on Value Iteration, it should be possible to achieve the optimal policy before the utility values reach convergence even if the utility values are inaccurate. Upon further inspection, this idea becomes intuitive. If an action for a particular state is clearly more optimal than the other options available, the exact value of the utility for that action holds little significance.

In this implementation, Modified Policy Iteration is used where a simplified form of Bellman Update calculates the approximate utility instead of the exact utility. This is to account for the growing state space in the bonus question's implementation.

A set of constants are first instantiated in *ApplicationInput.java*. The important constant to take note of is K. K depicts the number of times simplified Bellman Update is performed on the policy during the policy evaluation step of the policy iteration algorithm for each iteration.

A GridWorld is then instantiated with states which contain reward values, as well as whether the state is visitable or not. (E.g. Wall is not visitable).

The *PolicyIteration.java* class is called and the MDP representation, the GridWorld object, is passed into the object. Every state is replaced with a StateActionUtilityPair. This is to keep track of the action with the highest utility for a given state.

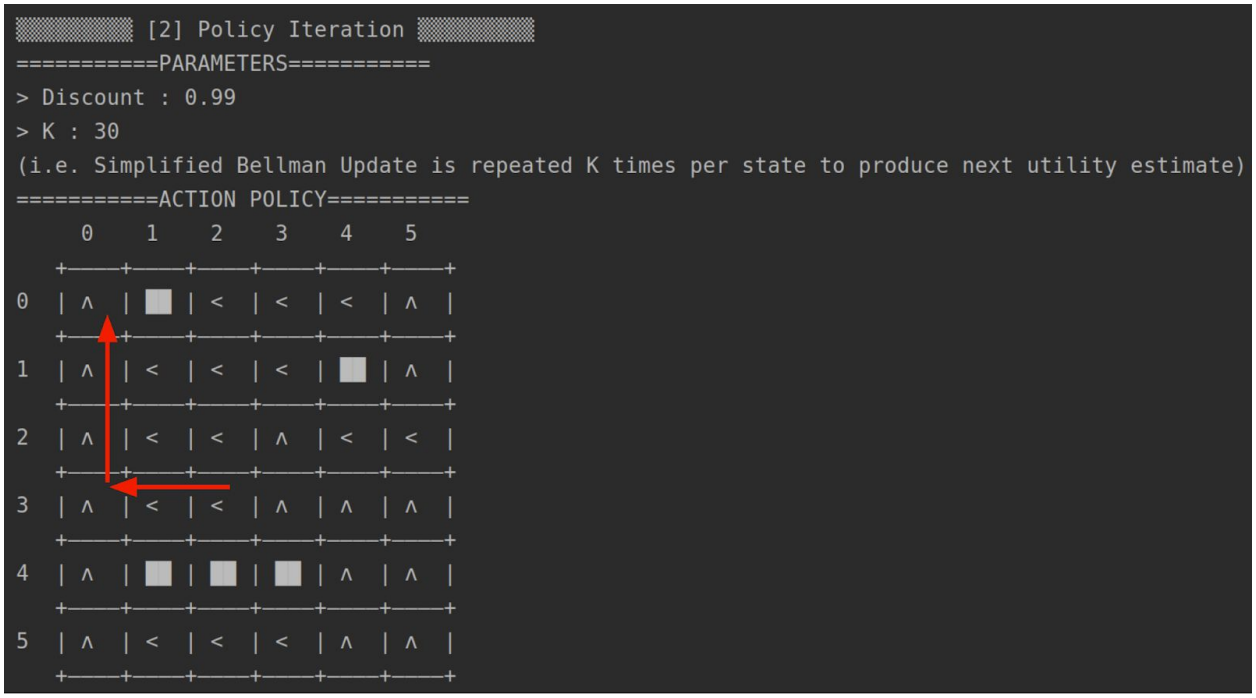
An initial policy with random actions and 0 utility for every state is then initialised.

For each iteration, policy iteration is divided into 2 steps:
Step 1) Policy Evaluation : Perform simplified bellman update K times.
Step 2) Policy Improvement : Calculate new policy using one-step lookahead.

The algorithm continues until the policy is unchanged between two consective iterations.

Finally, an array of StateActionUtilityPairs where the state's highest utility action along with its utility value is obtained.

3.2 Plot of Optimal Policy

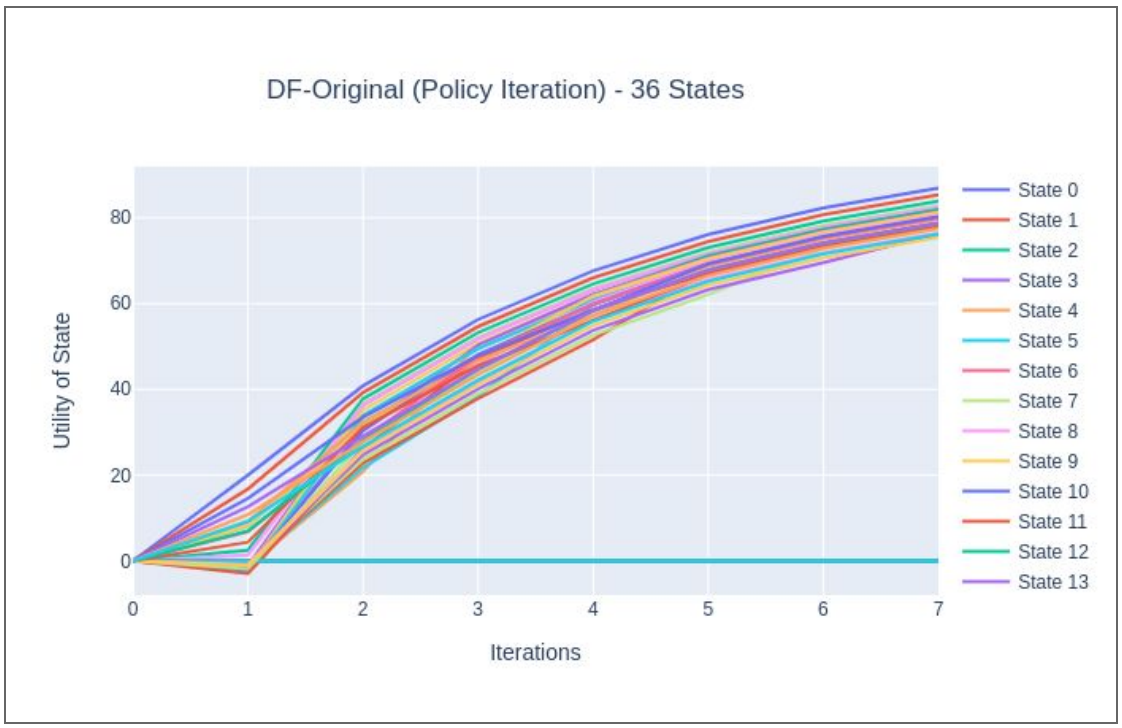


In the above image, it is clear that the agent has figured out the most optimal policy as explained in Section 1.2, which is to reach (0,0).

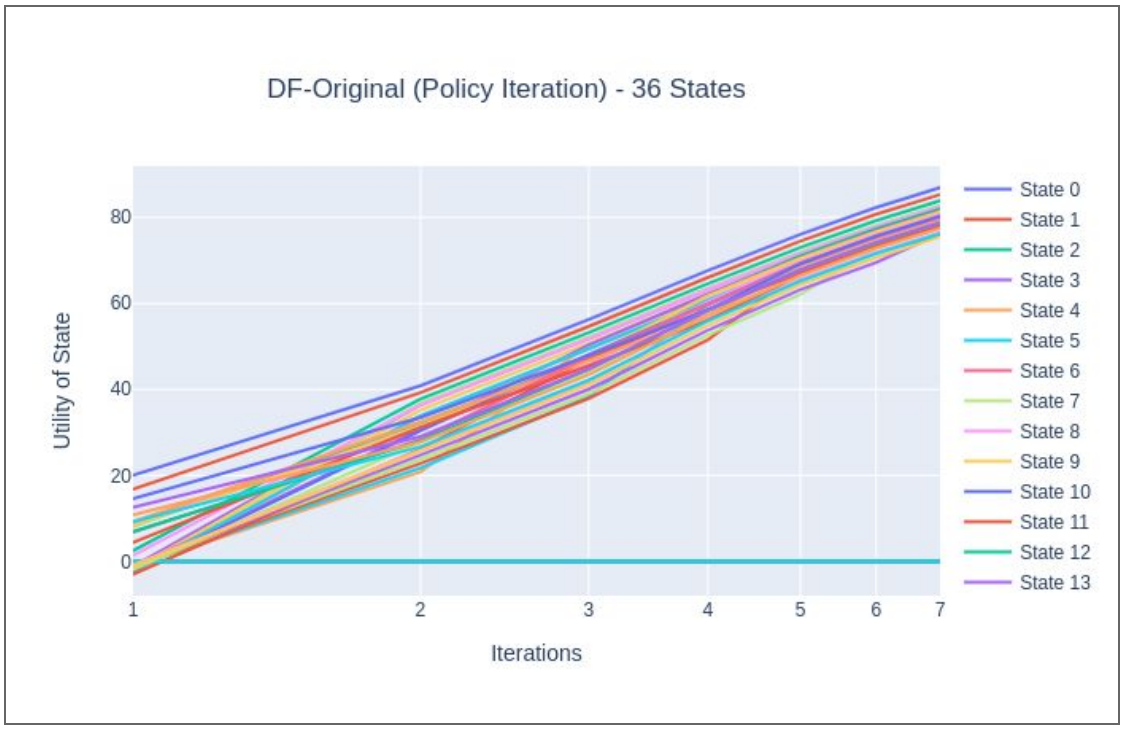
3.3 Utilities of all states

Refer to Appendix.

3.4 Plot of Utility Estimates as a function of the number of iterations



Policy Iteration Graph from "Interactive Graph Results.html"



Policy Iteration Graph with Log(Number of Iterations) from "Interactive Graph Results.html"

3.5 Policy Iteration Findings

In this implementation, a total of 7 iterations are required to reach convergence.

Experimentation with the K parameter yielded no significant change in results. The optimal policy was still found, albeit at a faster or slower pace depending on whether K was decreased or increased respectively.

We observe that Policy Iteration finished in significantly lesser iterations as compared to value iteration. However, each iteration took significantly longer as compared to value iteration.

4. Comparison between Value and Policy Iteration

Comparing both algorithms, we can see that value iteration converges faster than policy iteration to the same policy despite having a higher number of iterations. Upon further analysis, we observe that value iteration relies on finding the max utility value action every single iteration which would be an O(n) operation. This implies that value iteration would perform faster if the action space were to be small as is the case with the given gridworld problem with an action space of only 4. Policy iteration, which converges in a significantly lesser number of iterations is prefered when the action space is large.

We can also observe that policy iteration takes different numbers of iterations to converge after running it multiple times. This is so since the policy is instantiated with random actions before policy iteration is performed. We can hypothesize that convergence was achieved more quickly in these cases because the random policy initialised is already good to begin with.

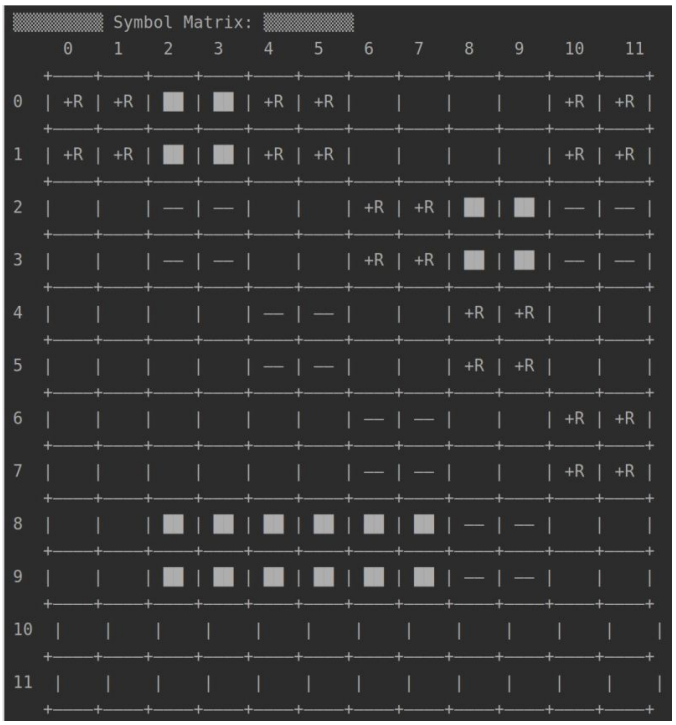
To conclude, with the current gridworld problem, policy iteration seems to be a better algorithm than value iteration.

5. Bonus Question

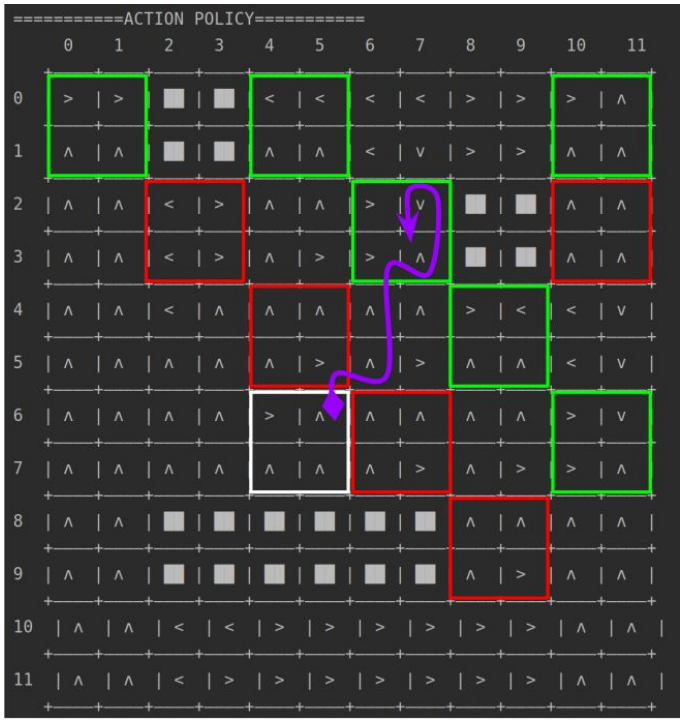
To answer the question on how the number of states as well as the complexity can affect convergence, a more complex transition probability where a probability for the opposite intent was implemented. However, the results that the agent managed to learn the optimal policy as long as the probability of the intent probability was the highest was expected. Therefore, a scaled version of the original state space was implemented.

5.1 Scale=2

Symbol Matrix when scale = 2:



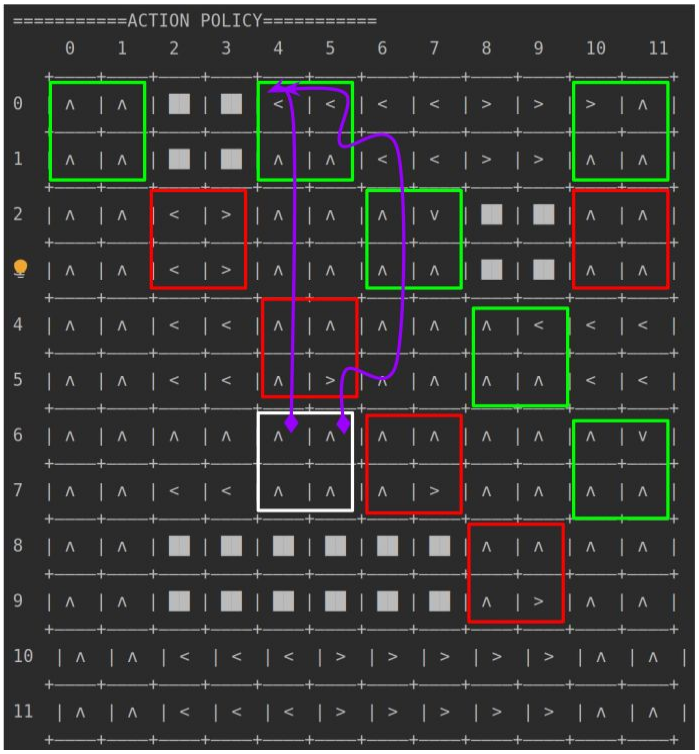
Optimal policy is to reach either (0:1, 0:1) or (0:1, 4:5) as they allow the agent to stay in the reward regions indefinitely.



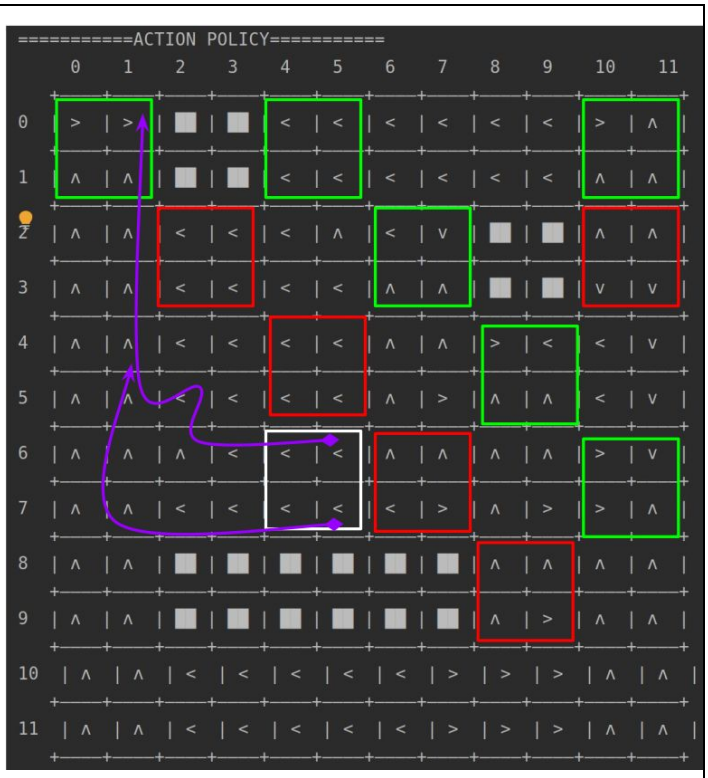
Scale =2 Diagram in Experiments.pdf

After running both algorithms, both policies yielded similar policies and were unable to reach the optimal regions as mentioned earlier The agent instead prefered to take the shortest route to a reward state and risk being thrown off in the case of an unintended action.

However, when a higher discount factor of 0.9999 was used, it allowed the agent to prioritise long-term rewards and the optimal policy was achieved.



Value Iteration (Scale=2, Discount=0.9999)

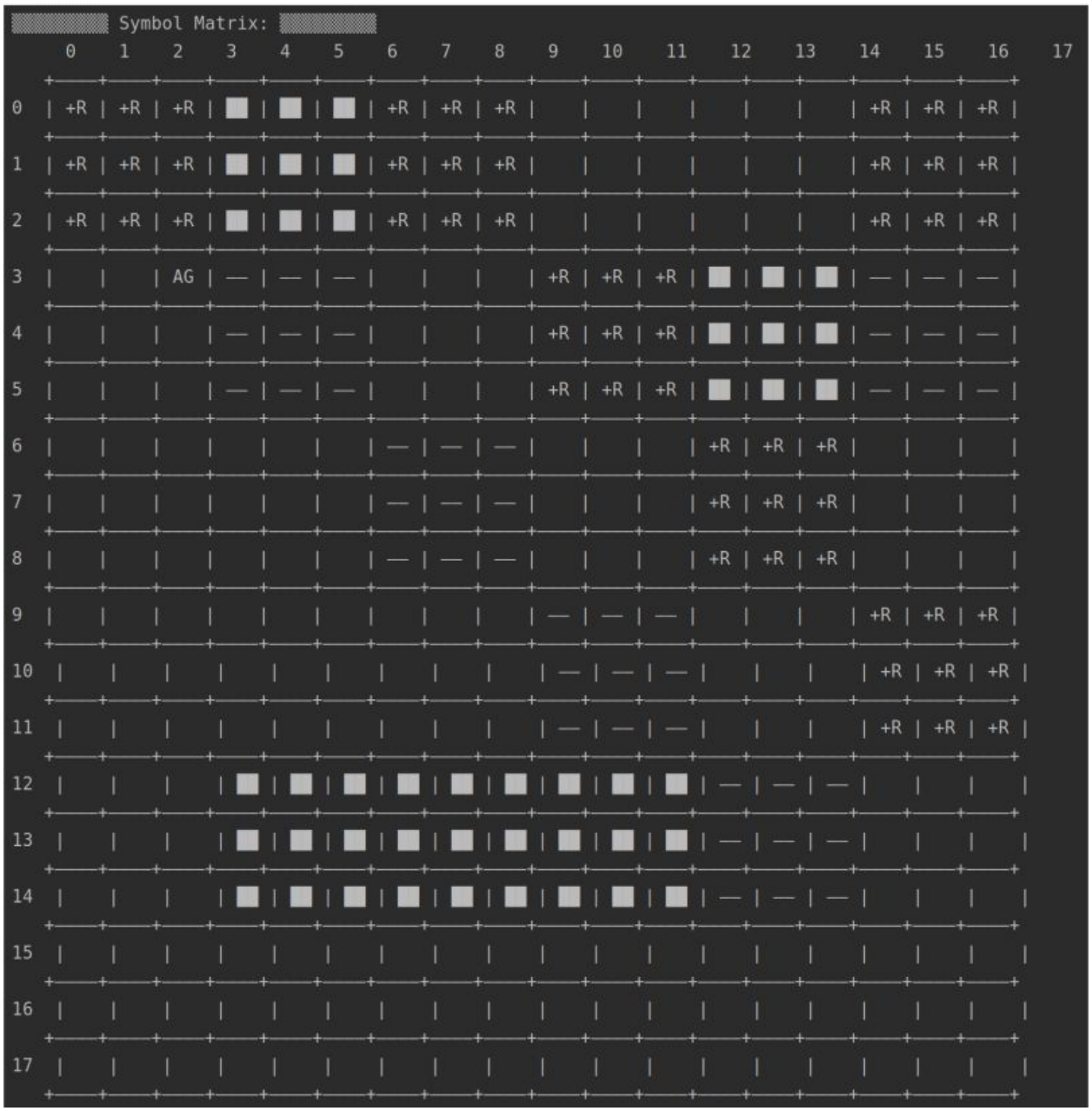


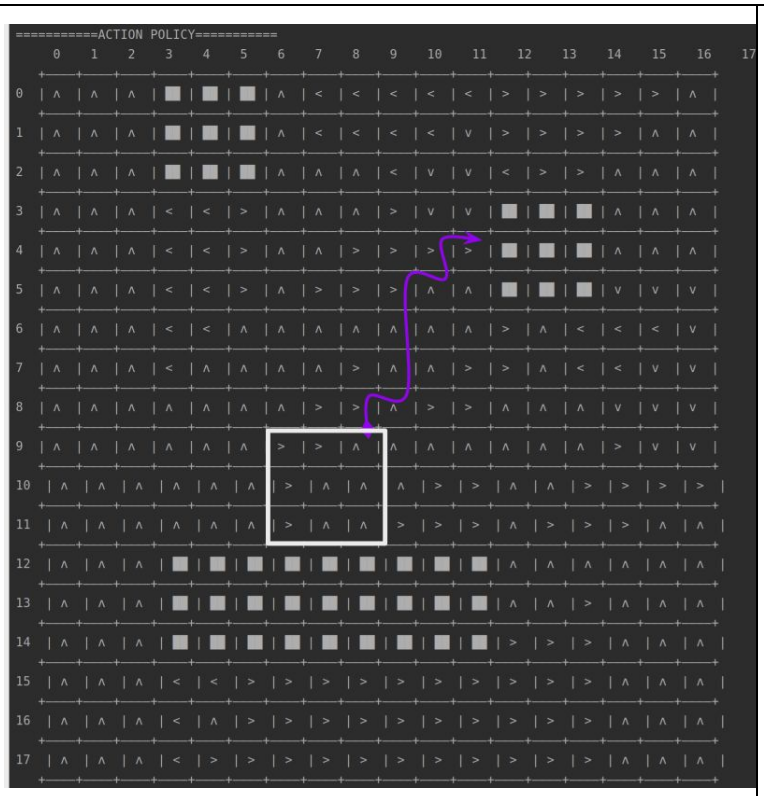
Policy Iteration (Scale=2, Discount=0.9999)

We observe that the optimal policy was obtained for both after the discount factor was changed.

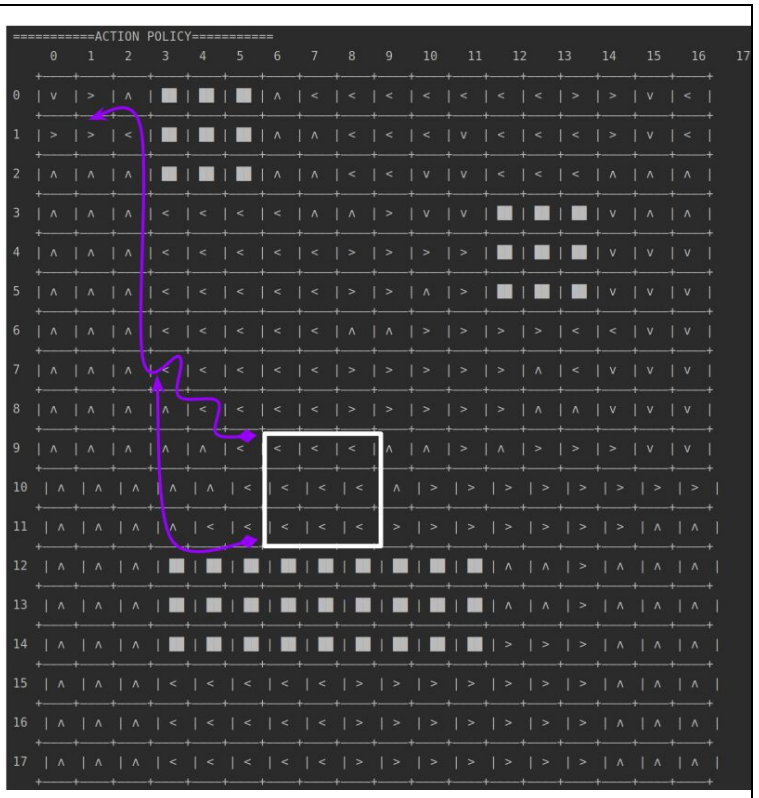
5.2 Scale=3

Symbol Matrix for scale = 3:





Value Iteration (Scale=3, Discount=0.999999)

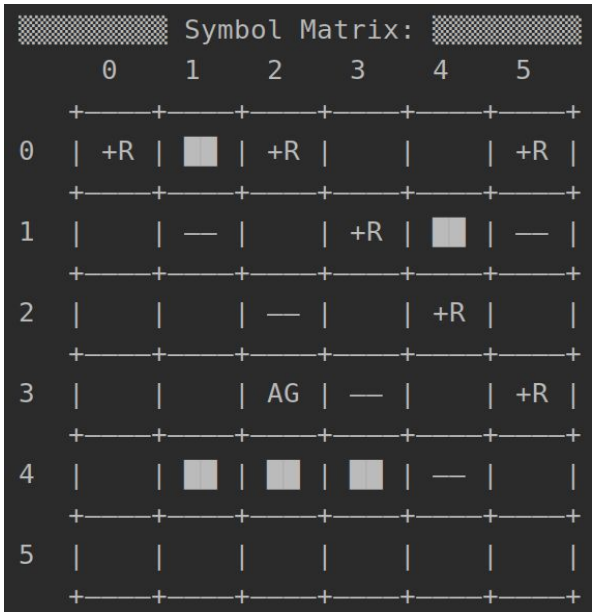


Policy Iteration (Scale=3, Discount=0.999999)

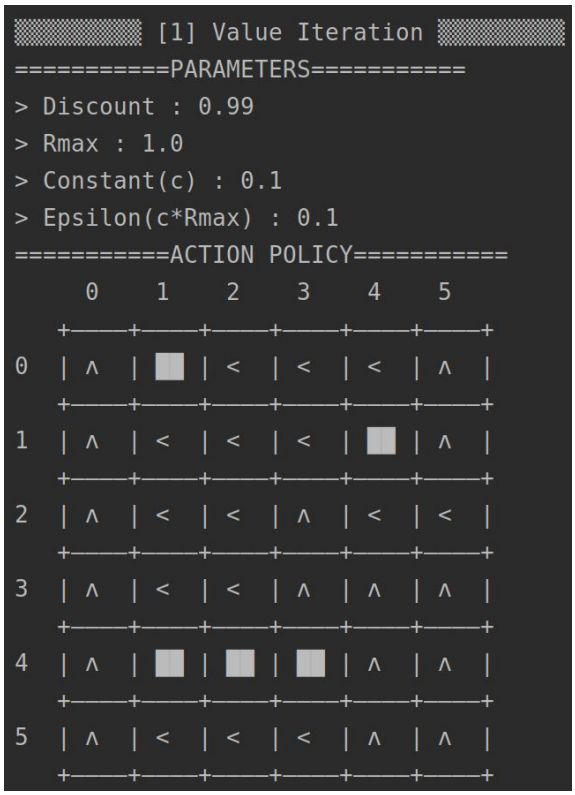
After learning that the discount rate was able to allow the agent to prioritise long-term rewards and hence achieve the optimal policy, the discount factor was set to 0.999999. However, the agent using value iteration was unable to achieve the optimal policy while the agent using policy iteration was only able to achieve the optimal policy on some occasions depending on the random policy initialised at first. This suggests that certain initial policies could cause the policy iteration agent to be stuck in a suboptimal policy.

Appendix

Main.java Results (Original GridWorld, scale=1)



Value Iteration (Scale=1):



```
===========UTILITIES GRID===========
        0          1          2          3          4          5
   +----------+----------+----------+----------+----------+----------+
0  |  99.901  |  ######  |  94.946  |  93.776  |  92.555  |  93.229  |
   +----------+----------+----------+----------+----------+----------+
1  |  98.294  |  95.784  |  94.446  |  94.298  |  ######  |  90.819  |
   +----------+----------+----------+----------+----------+----------+
2  |  96.849  |  95.487  |  93.195  |  93.077  |  93.003  |  91.696  |
   +----------+----------+----------+----------+----------+----------+
3  |  95.455  |  94.353  |  93.133  |  91.016  |  91.715  |  91.789  |
   +----------+----------+----------+----------+----------+----------+
4  |  94.213  |  ######  |  ######  |  ######  |  89.449  |  90.467  |
   +----------+----------+----------+----------+----------+----------+
5  |  92.838  |  91.629  |  90.436  |  89.257  |  88.470  |  89.198  |
   +----------+----------+----------+----------+----------+----------+
Number of iterations: 688
Convergence Criteria: 0.0010101010101010112
```

```
===========REFERENCE UTILITIES OF STATES===========
> Coordinates are in (col,row) format. Top-left corner is (0,0) <
(0,0) : 99.900685
(0,1) : 98.294047
(0,2) : 96.849185
(0,3) : 95.454524
(0,4) : 94.213205
(0,5) : 92.838160
(1,1) : 95.783703
(1,2) : 95.487113
(1,3) : 94.353179
(1,5) : 91.629463
(2,0) : 94.946142
(2,1) : 94.445684
(2,2) : 93.195113
(2,3) : 93.133231
(2,5) : 90.435837
(3,0) : 93.775686
(3,1) : 94.298400
(3,2) : 93.076958
(3,3) : 91.015942
(3,5) : 89.257095
(4,0) : 92.555300
(4,2) : 93.003054
(4,3) : 91.715092
(4,4) : 89.449098
(4,5) : 88.469784
(5,0) : 93.229188
(5,1) : 90.818608
(5,2) : 91.695556
(5,3) : 91.788770
(5,4) : 90.467451
(5,5) : 89.198376
```

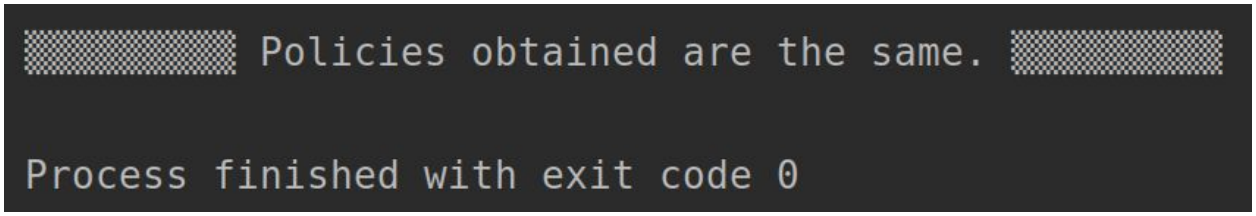
Policy Iteration (Scale=1):

```
▒▒▒▒▒▒▒▒▒▒ [2] Policy Iteration ▒▒▒▒▒▒▒▒▒▒
==========PARAMETERS==========
> Discount : 0.99
> K : 30
(i.e. Simplified Bellman Update is repeated K times per state to produce next utility estimate)
==========ACTION POLICY==========
     0    1    2    3    4    5
   +----+----+----+----+----+----+
0  | ^  | ## | <  | <  | <  | ^  |
   +----+----+----+----+----+----+
1  | ^  | <  | <  | <  | ## | ^  |
   +----+----+----+----+----+----+
2  | ^  | <  | <  | ^  | <  | <  |
   +----+----+----+----+----+----+
3  | ^  | <  | <  | ^  | ^  | ^  |
   +----+----+----+----+----+----+
4  | ^  | ## | ## | ## | ^  | ^  |
   +----+----+----+----+----+----+
5  | ^  | <  | <  | <  | ^  | ^  |
   +----+----+----+----+----+----+
```

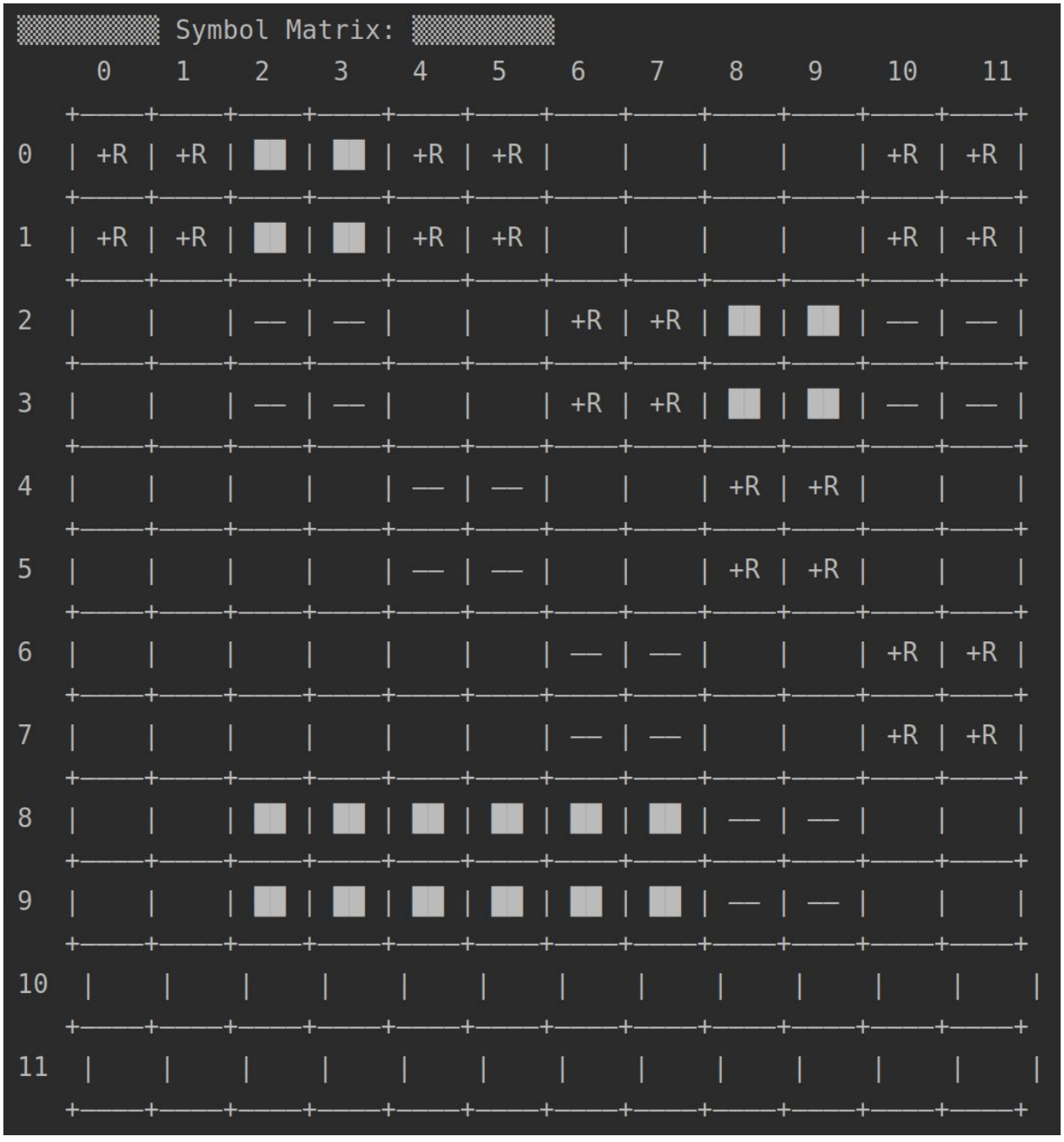
```
==========UTILITIES GRID==========
         0          1          2          3          4          5
    +----------+----------+----------+----------+----------+----------+
0   |  86.895  |  ######  |  81.940  |  80.770  |  79.550  |  80.223  |
    +----------+----------+----------+----------+----------+----------+
1   |  85.288  |  82.778  |  81.440  |  81.293  |  ######  |  77.812  |
    +----------+----------+----------+----------+----------+----------+
2   |  83.844  |  82.481  |  80.189  |  80.071  |  79.997  |  78.690  |
    +----------+----------+----------+----------+----------+----------+
3   |  82.449  |  81.348  |  80.128  |  78.010  |  78.709  |  78.783  |
    +----------+----------+----------+----------+----------+----------+
4   |  81.208  |  ######  |  ######  |  ######  |  76.443  |  77.462  |
    +----------+----------+----------+----------+----------+----------+
5   |  79.833  |  78.624  |  77.430  |  76.251  |  75.464  |  76.193  |
    +----------+----------+----------+----------+----------+----------+
Number of iterations: 7
```

```
===========REFERENCE UTILITIES OF STATES===========
> Coordinates are in (col,row) format. Top-left corner is (0,0) <
(0,0) : 86.895040
(0,1) : 85.288401
(0,2) : 83.843540
(0,3) : 82.448879
(0,4) : 81.207559
(0,5) : 79.832514
(1,1) : 82.778057
(1,2) : 82.481467
(1,3) : 81.347533
(1,5) : 78.623817
(2,0) : 81.940432
(2,1) : 81.440031
(2,2) : 80.189466
(2,3) : 80.127585
(2,5) : 77.430192
(3,0) : 80.769974
(3,1) : 81.292739
(3,2) : 80.071297
(3,3) : 78.010279
(3,5) : 76.251449
(4,0) : 79.549580
(4,2) : 79.997390
(4,3) : 78.709418
(4,4) : 76.443412
(4,5) : 75.464092
(5,0) : 80.223006
(5,1) : 77.812361
(5,2) : 78.689818
(5,3) : 78.783029
(5,4) : 77.461707
(5,5) : 76.192628
```

Final Results (Scale=1)



ComplexGridWorldExperiment1.java Results (Scale=2)



Value Iteration (Scale=2):

```
▒▒▒▒▒▒▒▒▒▒ [1] Value Iteration ▒▒▒▒▒▒▒▒▒▒
==========PARAMETERS==========
> Discount : 0.99
> Rmax : 1.0
> Constant(c) : 0.1
> Epsilon(c*Rmax) : 0.1
==========ACTION POLICY==========
     0    1    2    3    4    5    6    7    8    9    10    11
   +————+————+————+————+————+————+————+————+————+————+————+————+
0  | ∧  | ∧  | ▒▒ | ▒▒ | ∧  | <  | <  | <  | >  | >  | >  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
1  | ∧  | ∧  | ▒▒ | ▒▒ | ∧  | ∧  | <  | v  | >  | >  | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
2  | ∧  | ∧  | <  | >  | ∧  | ∧  | >  | v  | ▒▒ | ▒▒ | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
3  | ∧  | ∧  | <  | >  | ∧  | >  | >  | ∧  | ▒▒ | ▒▒ | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
4  | ∧  | ∧  | <  | ∧  | ∧  | ∧  | ∧  | ∧  | >  | <  | <  | v  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
5  | ∧  | ∧  | ∧  | ∧  | ∧  | >  | ∧  | >  | ∧  | ∧  | v  | v  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
6  | ∧  | ∧  | ∧  | ∧  | >  | ∧  | ∧  | ∧  | ∧  | ∧  | >  | v  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
7  | ∧  | ∧  | ∧  | ∧  | ∧  | ∧  | ∧  | >  | ∧  | >  | >  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
8  | ∧  | ∧  | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ∧  | ∧  | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
9  | ∧  | ∧  | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ∧  | >  | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
10  | ∧  | ∧  | <  | <  | >  | >  | >  | >  | >  | >  | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
11  | ∧  | ∧  | <  | >  | >  | >  | >  | >  | >  | >  | ∧  | ∧  |
   +————+————+————+————+————+————+————+————+————+————+————+————+
```

==========UTILITIES GRID==========

	0	1	2	3	4	5	6	7	8	9	10	11
0	99.901	99.901			99.755	99.741	98.438	97.208	97.137	98.426	99.735	99.750
1	99.901	99.901			99.741	99.607	98.375	97.599	97.050	98.317	99.596	99.735
2	98.572	98.312	95.686	95.515	98.146	98.346	98.802	98.855			97.120	97.229
3	97.238	96.799	94.452	94.230	96.648	97.260	98.690	98.849			94.671	94.758
4	95.915	95.435	94.093	93.228	94.176	94.974	97.188	97.618	98.475	98.477	96.948	96.032
5	94.606	94.108	92.915	92.097	91.948	93.369	95.763	96.847	98.316	98.348	97.195	97.182
6	93.314	92.813	91.743	90.985	90.942	92.150	93.324	94.525	96.803	97.169	98.332	98.459
7	92.038	91.546	90.582	89.885	89.879	90.861	91.130	93.161	95.431	96.776	98.292	98.457
8	90.791	90.402							93.153	94.449	96.773	97.133
9	89.570	89.262							90.990	92.995	95.295	95.808
10	88.357	87.987	86.719	85.525	86.410	87.593	88.801	90.037	91.305	92.649	93.960	94.496
11	87.155	86.762	85.749	85.030	86.094	87.193	88.296	89.399	90.500	91.594	92.670	93.201

Number of iterations: 688

Convergence Criteria: 0.0010101010101010112

==========REFERENCE UTILITIES OF STATES==========

> Coordinates are in (col,row) format. Top-left corner is (0,0) <

(0,0) : 99.900685

(0,1) : 99.900685

(0,2) : 98.571833

(0,3) : 97.237541

(0,4) : 95.914738

(0,5) : 94.606170

(0,6) : 93.313641

(0,7) : 92.038290

(0,8) : 90.791471

(0,9) : 89.570203

(0,10) : 88.356591

(0,11) : 87.155226

(1,0) : 99.900685

(1,1) : 99.900685

(1,2) : 98.311836

(1,3) : 96.799252

(1,4) : 95.434743

(1,5) : 94.107882

(1,6) : 92.813101

(1,7) : 91.546346

(1,8) : 90.401852

(1,9) : 89.261624

(1,10) : 87.986670
(1,11) : 86.761943
(2,2) : 95.685611
(2,3) : 94.452060
(2,4) : 94.092625
(2,5) : 92.914622
(2,6) : 91.743449
(2,7) : 90.581536
(2,10) : 86.718730
(2,11) : 85.748746
(3,2) : 95.515386
(3,3) : 94.229885
(3,4) : 93.227701
(3,5) : 92.096720
(3,6) : 90.985500
(3,7) : 89.885138
(3,10) : 85.525231
(3,11) : 85.030212
(4,0) : 99.755470
(4,1) : 99.740785
(4,2) : 98.145954
(4,3) : 96.648119
(4,4) : 94.176301
(4,5) : 91.947729
(4,6) : 90.942298
(4,7) : 89.879212
(4,10) : 86.410467
(4,11) : 86.093691
(5,0) : 99.740785
(5,1) : 99.607133
(5,2) : 98.345666
(5,3) : 97.260164
(5,4) : 94.974134
(5,5) : 93.368833
(5,6) : 92.149521
(5,7) : 90.861351
(5,10) : 87.592846
(5,11) : 87.192876
(6,0) : 98.438174
(6,1) : 98.374596
(6,2) : 98.801603
(6,3) : 98.690018
(6,4) : 97.188095

(6,5) : 95.763360
(6,6) : 93.324320
(6,7) : 91.130075
(6,10) : 88.800553
(6,11) : 88.295541
(7,0) : 97.207958
(7,1) : 97.599270
(7,2) : 98.855046
(7,3) : 98.848521
(7,4) : 97.617697
(7,5) : 96.847106
(7,6) : 94.524503
(7,7) : 93.160881
(7,10) : 90.036640
(7,11) : 89.398998
(8,0) : 97.137319
(8,1) : 97.050297
(8,4) : 98.475117
(8,5) : 98.315669
(8,6) : 96.802713
(8,7) : 95.430566
(8,8) : 93.152542
(8,9) : 90.990397
(8,10) : 91.304912
(8,11) : 90.499809
(9,0) : 98.426435
(9,1) : 98.316558
(9,4) : 98.477031
(9,5) : 98.348410
(9,6) : 97.169271
(9,7) : 96.776466
(9,8) : 94.448602
(9,9) : 92.995095
(9,10) : 92.649445
(9,11) : 91.593586
(10,0) : 99.734685
(10,1) : 99.595952
(10,2) : 97.119551
(10,3) : 94.671207
(10,4) : 96.947607
(10,5) : 97.195247
(10,6) : 98.331733
(10,7) : 98.292015

(10,8) : 96.772913
(10,9) : 95.294655
(10,10) : 93.959778
(10,11) : 92.669829
(11,0) : 99.749928
(11,1) : 99.734685
(11,2) : 97.229427
(11,3) : 94.758229
(11,4) : 96.032008
(11,5) : 97.181824
(11,6) : 98.458930
(11,7) : 98.456608
(11,8) : 97.133378
(11,9) : 95.807799
(11,10) : 94.495912
(11,11) : 93.200995

Policy Iteration (Scale=2):

```
▒▒▒▒▒▒▒▒▒▒ [2] Policy Iteration ▒▒▒▒▒▒▒▒▒▒
===========PARAMETERS===========
> Discount : 0.99
> K : 30
(i.e. Simplified Bellman Update is repeated K times per state to produce next utility estimate)
===========ACTION POLICY===========
     0    1    2    3    4    5    6    7    8    9    10    11
   +----+----+----+----+----+----+----+----+----+----+----+----+
0  | >  | >  | ▒▒ | ▒▒ | <  | <  | <  | <  | >  | >  | >  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
1  | ^  | ^  | ▒▒ | ▒▒ | ^  | ^  | <  | v  | >  | >  | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
2  | ^  | ^  | <  | >  | ^  | ^  | >  | v  | ▒▒ | ▒▒ | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
3  | ^  | ^  | <  | >  | ^  | >  | >  | ^  | ▒▒ | ▒▒ | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
4  | ^  | ^  | <  | ^  | ^  | ^  | ^  | ^  | >  | <  | <  | v  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
5  | ^  | ^  | ^  | ^  | ^  | >  | ^  | >  | ^  | ^  | <  | v  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
6  | ^  | ^  | ^  | ^  | >  | ^  | ^  | ^  | ^  | ^  | >  | v  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
7  | ^  | ^  | ^  | ^  | ^  | ^  | ^  | >  | ^  | >  | >  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
8  | ^  | ^  | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ^  | ^  | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
9  | ^  | ^  | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ▒▒ | ^  | >  | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
10  | ^  | ^  | <  | <  | >  | >  | >  | >  | >  | >  | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
11  | ^  | ^  | <  | >  | >  | >  | >  | >  | >  | >  | ^  | ^  |
   +----+----+----+----+----+----+----+----+----+----+----+----+
```

==========UTILITIES GRID==========

	0	1	2	3	4	5	6	7	8	9	10	11
0	96.922	96.922			96.833	96.818	95.515	94.283	94.137	95.426	96.734	96.749
1	96.922	96.922			96.818	96.684	95.451	94.658	94.050	95.316	96.595	96.734
2	95.593	95.333	92.707	92.592	95.223	95.422	95.867	95.919			94.119	94.229
3	94.259	93.821	91.473	91.305	93.724	94.326	95.754	95.913			91.671	91.758
4	92.936	92.456	91.114	90.298	91.251	92.040	94.251	94.670	95.420	95.421	93.895	92.874
5	91.627	91.130	89.941	89.162	89.019	90.422	92.815	93.806	95.261	95.291	94.115	94.011
6	90.335	89.836	88.773	88.046	87.997	89.202	90.367	91.490	93.749	94.100	95.160	95.278
7	89.060	88.570	87.613	86.942	86.933	87.912	88.163	90.098	92.365	93.608	95.111	95.275
8	87.813	87.425							90.076	91.288	93.593	93.951
9	86.592	86.285							87.902	89.816	92.114	92.625
10	85.379	85.010	83.742	82.528	83.232	84.416	85.625	86.862	88.132	89.468	90.779	91.313
11	84.177	83.785	82.772	81.868	82.912	84.011	85.114	86.217	87.318	88.411	89.488	90.018

Number of iterations: 12

==========REFERENCE UTILITIES OF STATES==========

> Coordinates are in (col,row) format. Top-left corner is (0,0) <

(0,0) : 96.921934
(0,1) : 96.921934
(0,2) : 95.593082
(0,3) : 94.258791
(0,4) : 92.935995
(0,5) : 91.627487
(0,6) : 90.335091
(0,7) : 89.059947
(0,8) : 87.813295
(0,9) : 86.592160
(0,10) : 85.378654
(0,11) : 84.177375
(1,0) : 96.921934
(1,1) : 96.921934
(1,2) : 95.333086
(1,3) : 93.820508
(1,4) : 92.456053
(1,5) : 91.129680
(1,6) : 89.835613
(1,7) : 88.569669
(1,8) : 87.425008
(1,9) : 86.284647

(1,10) : 85.009585
(1,11) : 83.784773
(2,2) : 92.706867
(2,3) : 91.473370
(2,4) : 91.114422
(2,5) : 89.940806
(2,6) : 88.772734
(2,7) : 87.613019
(2,10) : 83.741636
(2,11) : 82.771584
(3,2) : 92.592399
(3,3) : 91.305261
(3,4) : 90.297606
(3,5) : 89.162387
(3,6) : 88.046378
(3,7) : 86.942391
(3,10) : 82.527531
(3,11) : 81.868202
(4,0) : 96.832871
(4,1) : 96.818171
(4,2) : 95.223172
(4,3) : 93.723981
(4,4) : 91.250585
(4,5) : 89.019021
(4,6) : 87.996531
(4,7) : 86.933331
(4,10) : 83.232399
(4,11) : 82.911744
(5,0) : 96.818171
(5,1) : 96.684388
(5,2) : 95.421732
(5,3) : 94.325707
(5,4) : 92.040267
(5,5) : 90.421925
(5,6) : 89.201683
(5,7) : 87.911773
(5,10) : 84.415905
(5,11) : 84.011108
(6,0) : 95.515415
(6,1) : 95.450664
(6,2) : 95.867048
(6,3) : 95.754232
(6,4) : 94.251242

(6,5) : 92.814990
(6,6) : 90.367380
(6,7) : 88.163286
(6,10) : 85.624850
(6,11) : 85.113831
(7,0) : 94.283166
(7,1) : 94.658253
(7,2) : 95.919378
(7,3) : 95.912781
(7,4) : 94.669835
(7,5) : 93.805559
(7,6) : 91.490241
(7,7) : 90.098419
(7,10) : 86.862313
(7,11) : 86.217197
(8,0) : 94.136853
(8,1) : 94.049831
(8,4) : 95.419706
(8,5) : 95.261448
(8,6) : 93.749075
(8,7) : 92.364586
(8,8) : 90.075993
(8,9) : 87.902353
(8,10) : 88.132137
(8,11) : 87.317733
(9,0) : 95.425969
(9,1) : 95.316093
(9,4) : 95.421404
(9,5) : 95.290513
(9,6) : 94.100452
(9,7) : 93.608337
(9,8) : 91.288070
(9,9) : 89.816178
(9,10) : 89.467847
(9,11) : 88.411007
(10,0) : 96.734219
(10,1) : 96.595486
(10,2) : 94.119085
(10,3) : 91.670741
(10,4) : 93.895080
(10,5) : 94.114909
(10,6) : 95.160289
(10,7) : 95.111094

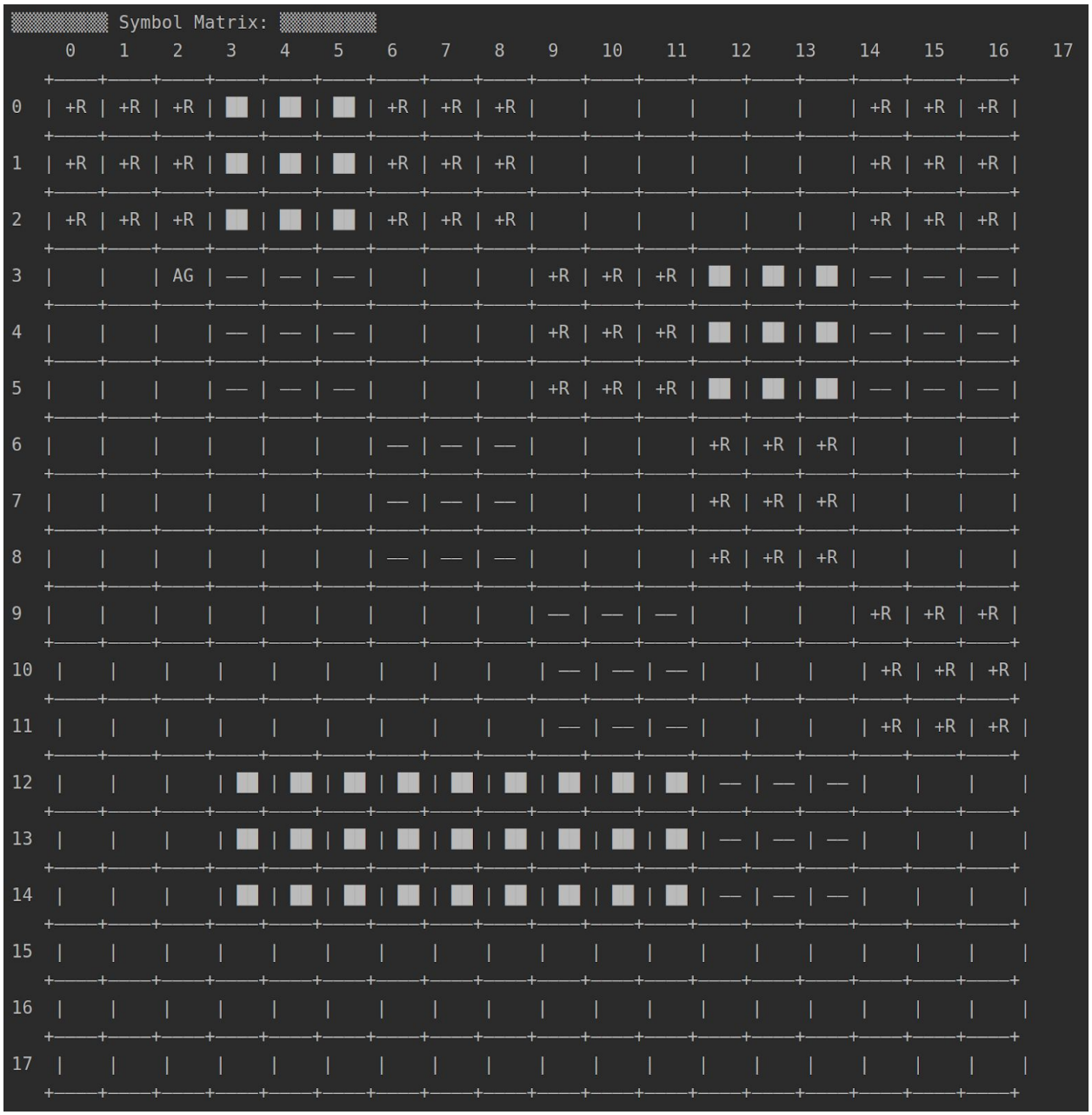
(10,8) : 93.593415
(10,9) : 92.114413
(10,10) : 90.778622
(10,11) : 89.487788
(11,0) : 96.749463
(11,1) : 96.734219
(11,2) : 94.228962
(11,3) : 91.757763
(11,4) : 92.874280
(11,5) : 94.011440
(11,6) : 95.277851
(11,7) : 95.274969
(11,8) : 93.951396
(11,9) : 92.625428
(11,10) : 91.313089
(11,11) : 90.017669

Final Results (Scale=2):

```
▒▒▒▒▒▒▒▒▒▒▒▒▒ Policies obtained are different. ▒▒▒▒▒▒▒▒▒▒▒▒▒
> Observed difference @(0,0)
> Observed difference @(1,0)
> Observed difference @(4,0)
> Observed difference @(10,5)
> [TOTAL NUMBER OF DIFFERENCES] = 4

Process finished with exit code 0
```

ComplexGridWorldExperiment2.java Results (Scale=3)



Value Iteration (Scale=3):

```
[1] Value Iteration
==========PARAMETERS==========
> Discount : 0.99
> Rmax : 1.0
> Constant(c) : 0.1
> Epsilon(c*Rmax) : 0.1
```

```
==========ACTION POLICY==========
    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
0 | ^ | ^ | ^ |   |   |   | ^ | < | < | < | < | < | > | > | > | > | > | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
1 | ^ | ^ | ^ |   |   |   | ^ | ^ | < | < | < | v | > | > | > | > | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
2 | ^ | ^ | ^ |   |   |   | ^ | ^ | ^ | < | v | v | < | > | > | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
3 | ^ | ^ | ^ | < | < | > | ^ | ^ | ^ | > | v | v |   |   |   | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
4 | ^ | ^ | ^ | < | > | > | ^ | ^ | > | > | > | > |   |   |   | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
5 | ^ | ^ | ^ | < | < | > | ^ | > | > | > | ^ | ^ |   |   |   | v | v | v |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
6 | ^ | ^ | ^ | < | < | ^ | ^ | ^ | ^ | ^ | ^ | ^ | > | ^ | < | < | < | v |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
7 | ^ | ^ | ^ | < | < | ^ | ^ | ^ | > | ^ | ^ | > | > | ^ | < | < | v | v |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
8 | ^ | ^ | ^ | ^ | ^ | ^ | ^ | > | > | ^ | > | > | ^ | ^ | ^ | v | v | v |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
9 | ^ | ^ | ^ | ^ | ^ | ^ | > | > | ^ | ^ | ^ | ^ | ^ | ^ | ^ | > | v | v |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
10 | ^ | ^ | ^ | ^ | ^ | ^ | > | ^ | ^ | ^ | > | > | ^ | ^ | > | > | > | > |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
11 | ^ | ^ | ^ | ^ | ^ | ^ | > | ^ | ^ | > | > | > | ^ | > | > | > | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
12 | ^ | ^ | ^ |   |   |   |   |   |   |   |   |   | ^ | ^ | ^ | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
13 | ^ | ^ | ^ |   |   |   |   |   |   |   |   |   | ^ | ^ | > | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
14 | ^ | ^ | ^ |   |   |   |   |   |   |   |   |   | > | > | > | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
15 | ^ | ^ | ^ | < | < | > | > | > | > | > | > | > | > | > | > | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
16 | ^ | ^ | ^ | < | > | > | > | > | > | > | > | > | > | > | > | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
17 | ^ | ^ | ^ | ^ | > | > | > | > | > | > | > | > | > | > | > | ^ | ^ | ^ |
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

==========UTILITIES GRID==========

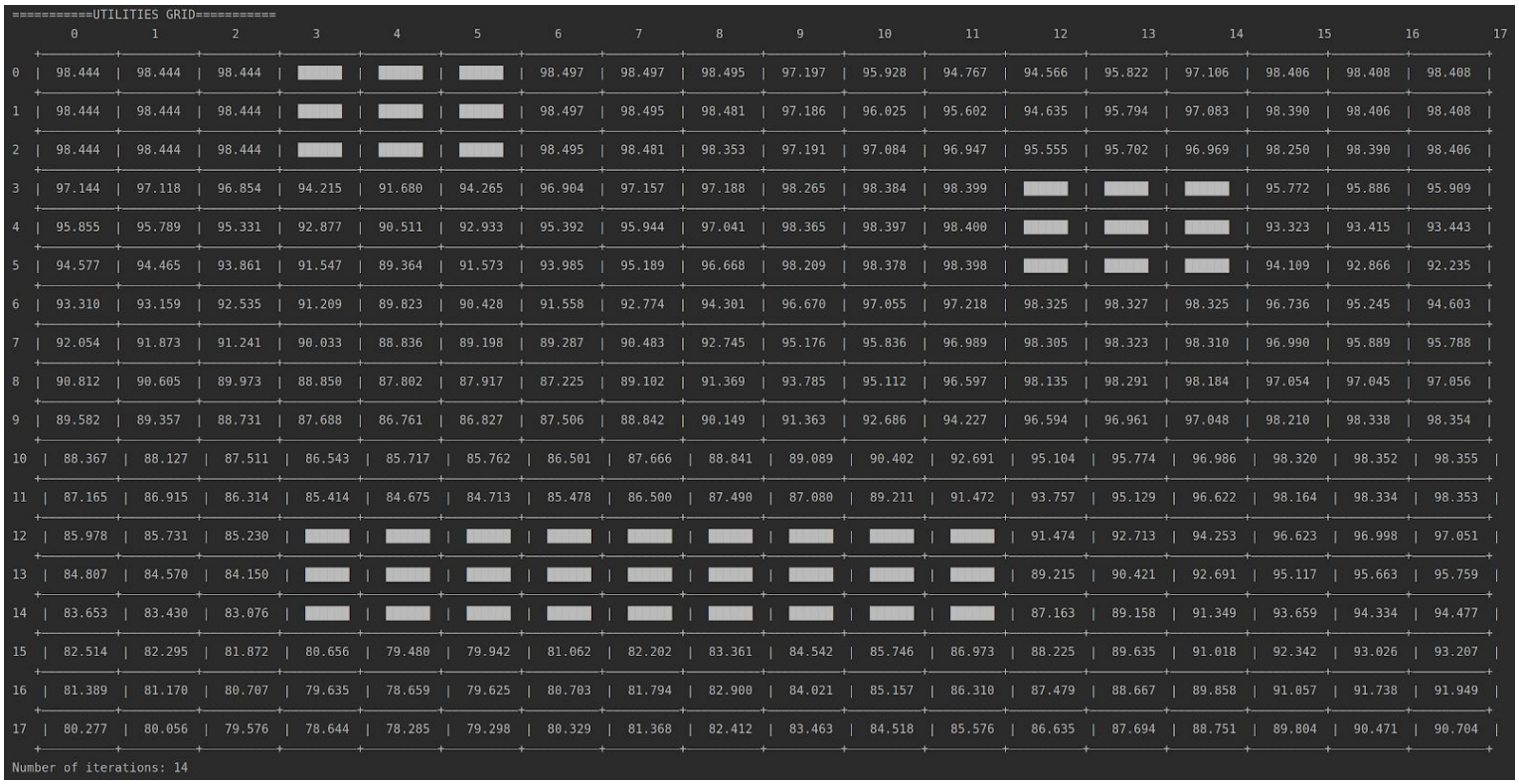
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	99.901	99.901	99.901	[redacted]	[redacted]	[redacted]	99.899	99.899	99.897	98.599	97.331	96.176	96.057	97.313	98.597	99.897	99.899	99.899
1	99.901	99.901	99.901	[redacted]	[redacted]	[redacted]	99.899	99.897	99.883	98.589	97.433	97.065	96.124	97.286	98.574	99.882	99.897	99.899
2	99.901	99.901	99.901	[redacted]	[redacted]	[redacted]	99.897	99.883	99.756	98.599	98.545	98.414	97.024	97.193	98.460	99.741	99.882	99.897
3	98.601	98.575	98.311	95.672	93.086	95.668	98.306	98.560	98.596	99.725	99.850	99.865	[redacted]	[redacted]	[redacted]	97.263	97.377	97.400
4	97.312	97.246	96.787	94.334	91.918	94.337	96.795	97.353	98.502	99.831	99.863	99.867	[redacted]	[redacted]	[redacted]	94.814	94.907	94.934
5	96.034	95.922	95.318	93.004	90.795	92.983	95.394	96.648	98.134	99.675	99.845	99.865	[redacted]	[redacted]	[redacted]	95.648	94.401	93.751
6	94.767	94.616	93.992	92.666	91.278	91.842	92.972	94.229	95.766	98.136	98.522	98.692	99.864	99.866	99.865	98.275	96.782	96.117
7	93.511	93.330	92.698	91.490	90.292	90.617	90.706	91.935	94.213	96.644	97.310	98.522	99.845	99.863	99.849	98.527	97.402	97.300
8	92.269	92.062	91.430	90.307	89.255	89.340	88.649	90.573	92.843	95.260	96.643	98.136	99.675	99.831	99.721	98.568	98.556	98.567
9	91.039	90.814	90.188	89.144	88.212	88.258	88.975	90.316	91.623	92.843	94.213	95.765	98.133	98.498	98.562	99.721	99.849	99.865
10	89.824	89.584	88.968	87.998	87.167	87.198	87.975	89.140	90.316	90.573	91.934	94.225	96.640	97.288	98.498	99.831	99.863	99.866
11	88.622	88.372	87.771	86.869	86.125	86.155	86.953	87.974	88.970	88.594	90.729	92.988	95.271	96.640	98.133	99.675	99.845	99.864
12	87.435	87.188	86.687	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	92.988	94.225	95.764	98.134	98.509	98.562
13	86.264	86.027	85.607	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	90.729	91.933	94.202	96.628	97.174	97.270
14	85.110	84.887	84.532	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	88.675	90.669	92.860	95.170	95.845	95.988
15	83.971	83.752	83.329	82.113	80.938	81.453	82.573	83.713	84.872	86.053	87.257	88.484	89.736	91.146	92.529	93.853	94.537	94.718
16	82.846	82.627	82.164	81.092	80.131	81.136	82.214	83.305	84.411	85.532	86.668	87.821	88.990	90.178	91.369	92.568	93.249	93.460
17	81.734	81.513	81.033	80.106	79.792	80.809	81.840	82.878	83.923	84.974	86.029	87.087	88.146	89.205	90.262	91.315	91.982	92.215

Number of iterations: 688
Convergence Criteria: 0.0010101010101010112

Policy Iteration (Scale=3):

```
▒▒▒▒▒▒▒▒▒▒ [2] Policy Iteration ▒▒▒▒▒▒▒▒▒▒
===========PARAMETERS===========
> Discount : 0.99
> K : 30
(i.e. Simplified Bellman Update is repeated K times per state to produce next utility estimate)
```

```
===========ACTION POLICY===========
     0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
0  | ^ | ^ | ^ | ▒ | ▒ | ▒ | < | < | < | < | < | < | > | > | > | > | > | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
1  | ^ | ^ | ^ | ▒ | ▒ | ▒ | ^ | < | < | < | < | v | > | > | > | > | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
2  | ^ | ^ | ^ | ▒ | ▒ | ▒ | ^ | ^ | ^ | < | v | v | < | > | > | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
3  | ^ | ^ | ^ | < | > | > | ^ | ^ | ^ | > | v | v | ▒ | ▒ | ▒ | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
4  | ^ | ^ | ^ | < | > | > | ^ | ^ | > | > | > | > | ▒ | ▒ | ▒ | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
5  | ^ | ^ | ^ | < | > | > | ^ | > | > | > | ^ | ^ | ▒ | ▒ | ▒ | v | v | v |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
6  | ^ | ^ | ^ | < | < | ^ | ^ | ^ | ^ | ^ | ^ | ^ | > | ^ | < | < | < | v |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
7  | ^ | ^ | ^ | < | < | ^ | ^ | ^ | > | ^ | ^ | > | > | ^ | < | < | v | v |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
8  | ^ | ^ | ^ | ^ | ^ | ^ | ^ | > | > | ^ | > | > | ^ | ^ | ^ | v | v | v |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
9  | ^ | ^ | ^ | ^ | ^ | ^ | > | > | ^ | ^ | ^ | ^ | ^ | ^ | > | > | v | v |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
10 | ^ | ^ | ^ | ^ | ^ | ^ | > | ^ | ^ | ^ | > | > | ^ | > | > | > | > | > |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
11 | ^ | ^ | ^ | ^ | ^ | ^ | > | ^ | ^ | > | > | > | > | > | > | > | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
12 | ^ | ^ | ^ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ^ | ^ | ^ | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
13 | ^ | ^ | ^ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ^ | ^ | > | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
14 | ^ | ^ | ^ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | ▒ | > | > | > | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
15 | ^ | ^ | ^ | < | < | > | > | > | > | > | > | > | > | > | > | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
16 | ^ | ^ | ^ | < | ^ | > | > | > | > | > | > | > | > | > | > | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
17 | ^ | ^ | ^ | ^ | > | > | > | > | > | > | > | > | > | > | > | ^ | ^ | ^ |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

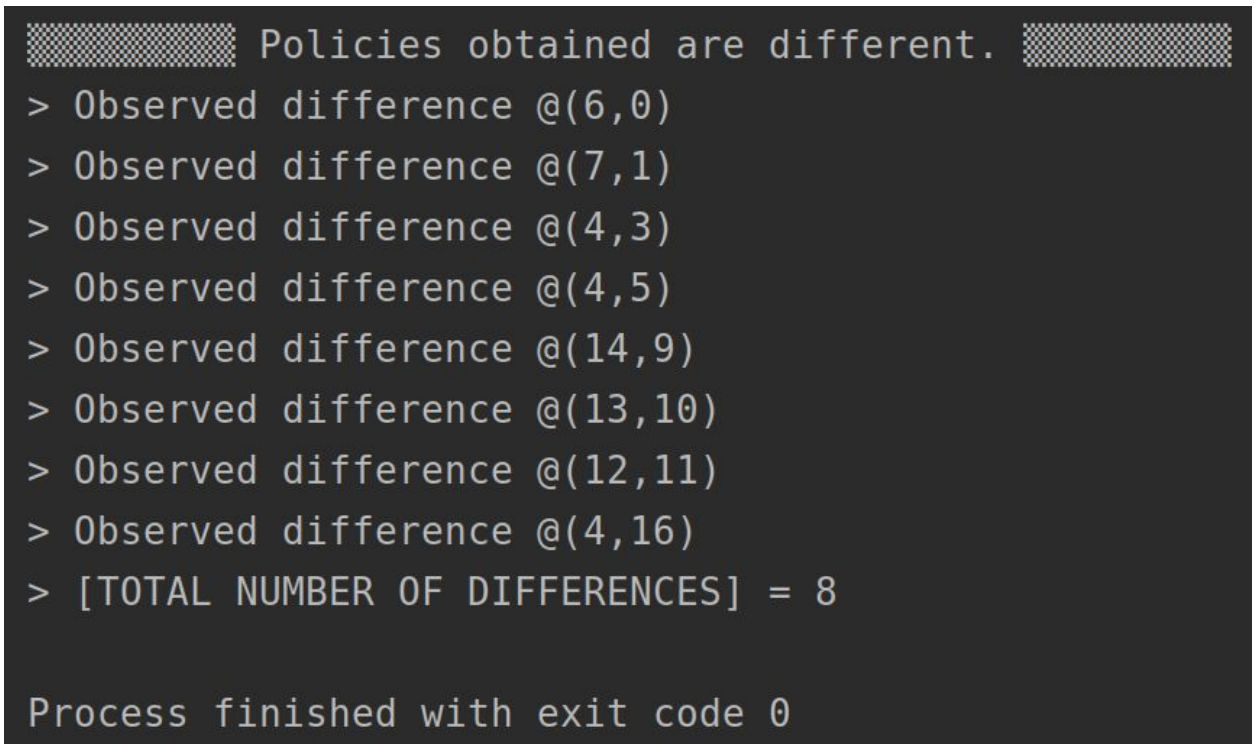


==========UTILITIES GRID==========

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	98.444	98.444	98.444	▓▓▓▓	▓▓▓▓	▓▓▓▓	98.497	98.497	98.495	97.197	95.928	94.767	94.566	95.822	97.106	98.406	98.408	98.408
1	98.444	98.444	98.444	▓▓▓▓	▓▓▓▓	▓▓▓▓	98.497	98.495	98.481	97.186	96.025	95.602	94.635	95.794	97.083	98.390	98.406	98.408
2	98.444	98.444	98.444	▓▓▓▓	▓▓▓▓	▓▓▓▓	98.495	98.481	98.353	97.191	97.084	96.947	95.555	95.702	96.969	98.250	98.390	98.406
3	97.144	97.118	96.854	94.215	91.680	94.265	96.904	97.157	97.188	98.265	98.384	98.399	▓▓▓▓	▓▓▓▓	▓▓▓▓	95.772	95.886	95.909
4	95.855	95.789	95.331	92.877	90.511	92.933	95.392	95.944	97.041	98.365	98.397	98.400	▓▓▓▓	▓▓▓▓	▓▓▓▓	93.323	93.415	93.443
5	94.577	94.465	93.861	91.547	89.364	91.573	93.985	95.189	96.668	98.209	98.378	98.398	▓▓▓▓	▓▓▓▓	▓▓▓▓	94.109	92.866	92.235
6	93.310	93.159	92.535	91.209	89.823	90.428	91.558	92.774	94.301	96.670	97.055	97.218	98.325	98.327	98.325	96.736	95.245	94.603
7	92.054	91.873	91.241	90.033	88.836	89.198	89.287	90.483	92.745	95.176	95.836	96.989	98.305	98.323	98.310	96.990	95.889	95.788
8	90.812	90.605	89.973	88.850	87.802	87.917	87.225	89.102	91.369	93.785	95.112	96.597	98.135	98.291	98.184	97.054	97.045	97.056
9	89.582	89.357	88.731	87.688	86.761	86.827	87.506	88.842	90.149	91.363	92.686	94.227	96.594	96.961	97.048	98.210	98.338	98.354
10	88.367	88.127	87.511	86.543	85.717	85.762	86.501	87.666	88.841	89.089	90.402	92.691	95.104	95.774	96.986	98.320	98.352	98.355
11	87.165	86.915	86.314	85.414	84.675	84.713	85.478	86.500	87.490	87.080	89.211	91.472	93.757	95.129	96.622	98.164	98.334	98.353
12	85.978	85.731	85.230	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	91.474	92.713	94.253	96.623	96.998	97.051
13	84.807	84.570	84.150	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	89.215	90.421	92.691	95.117	95.663	95.759
14	83.653	83.430	83.076	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	87.163	89.158	91.349	93.659	94.334	94.477
15	82.514	82.295	81.872	80.656	79.480	79.942	81.062	82.202	83.361	84.542	85.746	86.973	88.225	89.635	91.018	92.342	93.026	93.207
16	81.389	81.170	80.707	79.635	78.659	79.625	80.703	81.794	82.900	84.021	85.157	86.310	87.479	88.667	89.858	91.057	91.738	91.949
17	80.277	80.056	79.576	78.644	78.285	79.298	80.329	81.368	82.412	83.463	84.518	85.576	86.635	87.694	88.751	89.804	90.471	90.704

Number of iterations: 14

Final Results (Scale=3):



```
▓▓▓▓▓▓▓▓▓▓▓▓▓▓ Policies obtained are different. ▓▓▓▓▓▓▓▓▓▓▓▓▓▓
> Observed difference @(6,0)
> Observed difference @(7,1)
> Observed difference @(4,3)
> Observed difference @(4,5)
> Observed difference @(14,9)
> Observed difference @(13,10)
> Observed difference @(12,11)
> Observed difference @(4,16)
> [TOTAL NUMBER OF DIFFERENCES] = 8

Process finished with exit code 0
```