1. The Gridworld presented for this problem was a simple 5x5 grid with 2 obstacle blocks in the middle denoted as OO. The two main assumptions for this world were the probability transitions matrix and the discount reward factor. The probability transitions were assumed to be [0.8, 0.05, 0.05, 0.1] where the agent had an 80% chance of moving in the desired direction, 5% chance to move +- 90 degrees of the desired direction or 10% chance of not moving at all. The discount factor was set to 1 in this experiment.

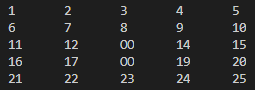


Figure 1: Gridworld States with obstacles

The goal state for this world was state 25 with a reward of +10. State 23 was a water state that had a negative reward of -10. The starting state was state 1 and, along with the rest of the states, had a reward of -0.04. This reward was introduced to help show the agent the optimal path(s) to the goal state from any state by following the highest state-value.

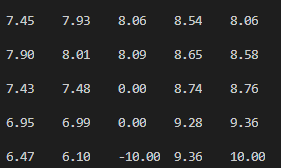


Figure 2: Gridworld State-Value

The state-values were updated following Bellman’s Equation and can be found inside Knowledge.\_\_init\_\_ function. As shown in Figure 2, the optimal policy would be: 1 🡪 2 🡪 3 🡪 4 🡪 9 🡪 14 🡪 19 🡪 20/24 🡪 25. Attempts were made to following a MAPE-K architecture as shown with the 5 different python files. Planning.py is the only one which contains the two different “policies” of random selection vs optimal selection.

Table 1: Simulation Results

|  |  |  |
| --- | --- | --- |
| Statistics | Random Results | Optimal Results |
| Mean | 976.10 | 2616.17 |
| Std Dev. | 824.46 | 1590.66 |
| Max | 9705.39 | 14023.14 |
| Min | -43.85 | 223.10 |
| Steps at Max | 1270 | 1669 |
| Steps at Min | 22 | 27 |

Between the two different planning policies, the optimal results always maintained a positive result and was generally higher. The last two statistics describe how many steps were taken to achieve the maximum and the minimum discounted rewards. Each episode was however many steps it took for the agent to reach the goal state. It is interesting to see how much 20% variation can cause as well as only utilizing state-values rather than action-values. For this model, failing to move also provided a reward. It is difficult to account for the environmental actions that can affect this system. On average, the optimal policy provided higher rewards in the end.

1. A GC’s primary job is to ensure the user or program has memory to continue operating. For this MDP, the primary goals were to reduce CPU usage as well as minimizing memory usage. The MDP state parameters are {Memory Usage, Time Between Collections}. These states should be quantized into buckets (i.e memory usage can be split into low, medium, and high) to minimize the number of states in the program. Generally, the GC wants to have low memory usage and a long time between collections.

This GC also implements three simple actions: {Do Nothing, Mark for Collection, Collection}. The “Do Nothing” state was included to illustrate that the GC may not have enough memory to collect where its worth taking CPU usage for. “Mark for Collection” is simply an action that will not change state but tick a counter. When high enough, it will trigger the “Collections” action which should change the memory usage and time between collections of the system.

The primary rewards for this system reflect the goals: {X – X/[CPU Utilization], Time Between Collections – Y, Z – Memory Usage}. These three reward factors each have parameters that can be set by the user and tuned for the desired output. Each can be measured in units that make sense for the user. For example, X can be a percentage of CPU Utilization. Y can be a time in seconds. Z can be a specific memory threshold. Generally, the rewards want to reduce CPU Utilization, maximize time between collections, and minimize memory usage based on the user defined parameters.

1. The short answer is yes. The total rewards for a system, denoted as Rc, is the total reward with an added constant c.

With two different policies, r1 and r2, and two different lengths K1 and K2, the number of added constants will be different because the second term will be different. The only time where the positive constant does not have an impact is if the MDP was unbounded (continuous) where K1 does equal K2 since they will always have the same length.