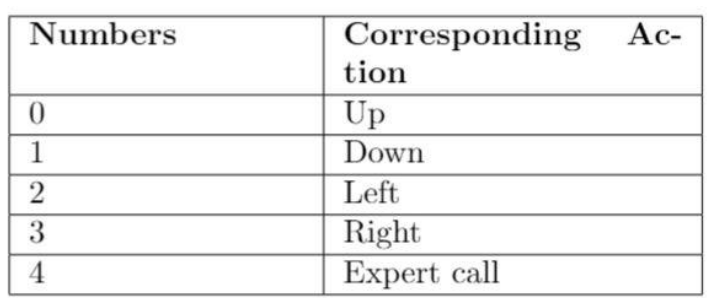
**Human in the loop Reinforcement Learning**

As in the paper, there are 4 implementations here:

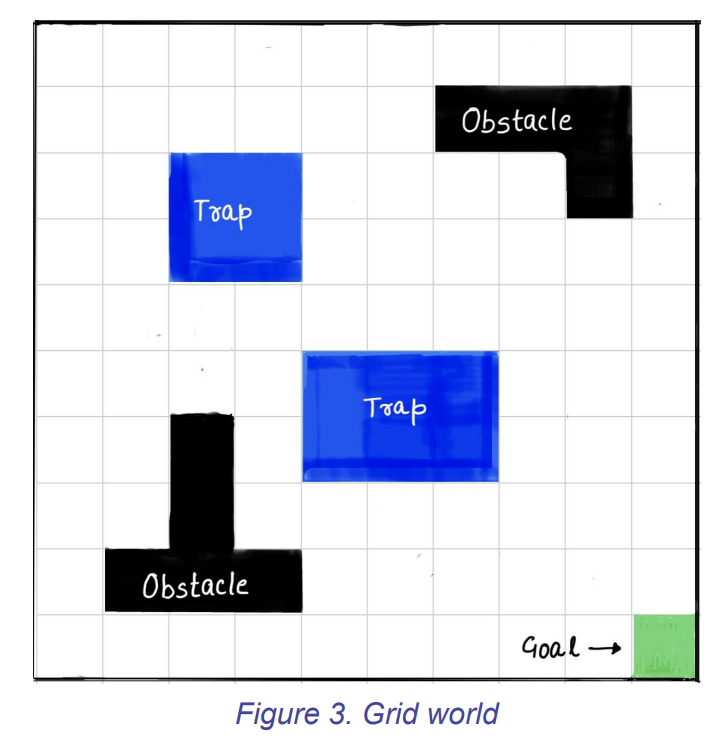
1. **Vanilla:** toy grid world problem
2. **ALG1:** where agent has the option to choose expert action at a cost of -5
3. **ALG2:** we are learning the variance of the rewards using Bellman equations

I have run all methods for 1,000,000 episodes. Note that experience replay was not used anywhere and the learning rate is 0.1.

We will follow this legend for all our plots (Expert call is only present in ALG1):



Grid structure: Notice the placement of trap and obstacles



1. **Vanilla:**
   1. **Training – 1 million episodes**

Chart

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence



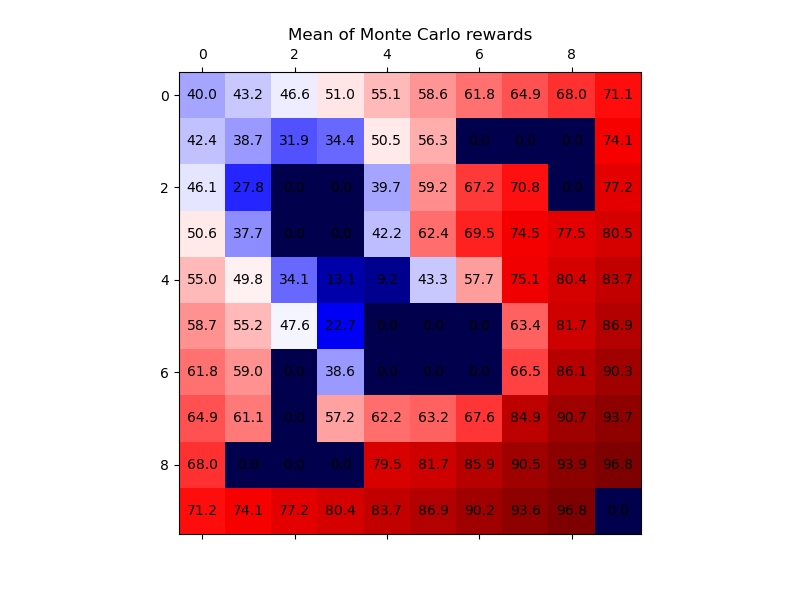
In the converged action space, almost every action near the trap takes the agent away from the trap

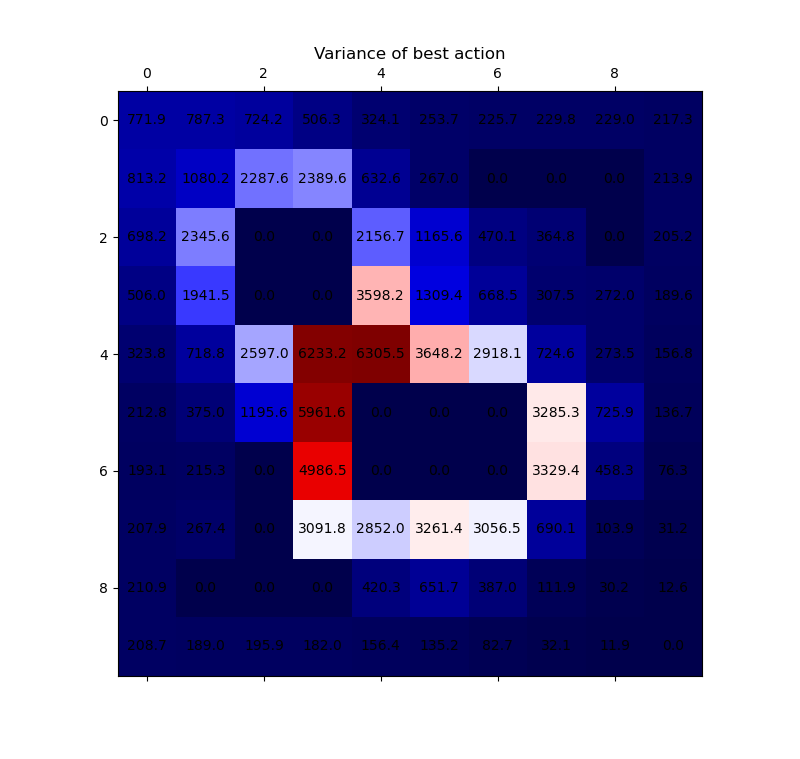


* 1. **Mean and variance of rewards using Monte Carlo – 100000 episodes:**

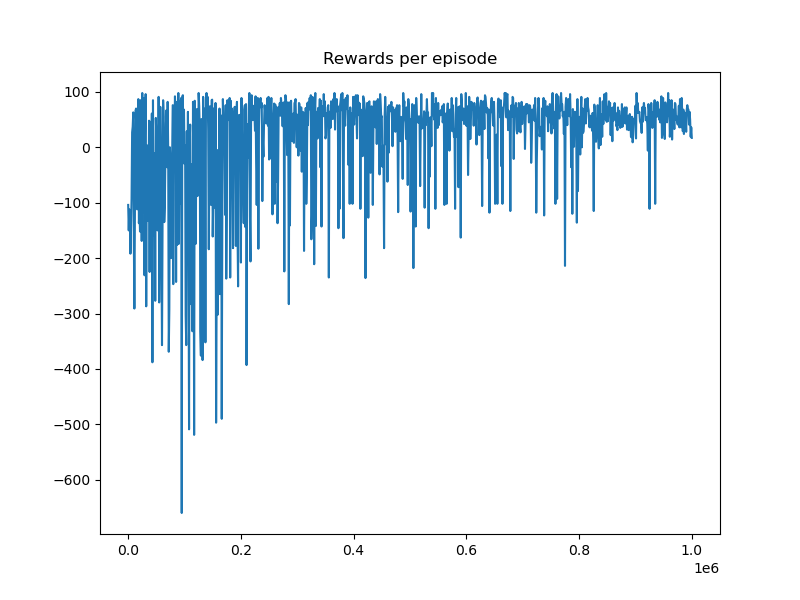
Chart

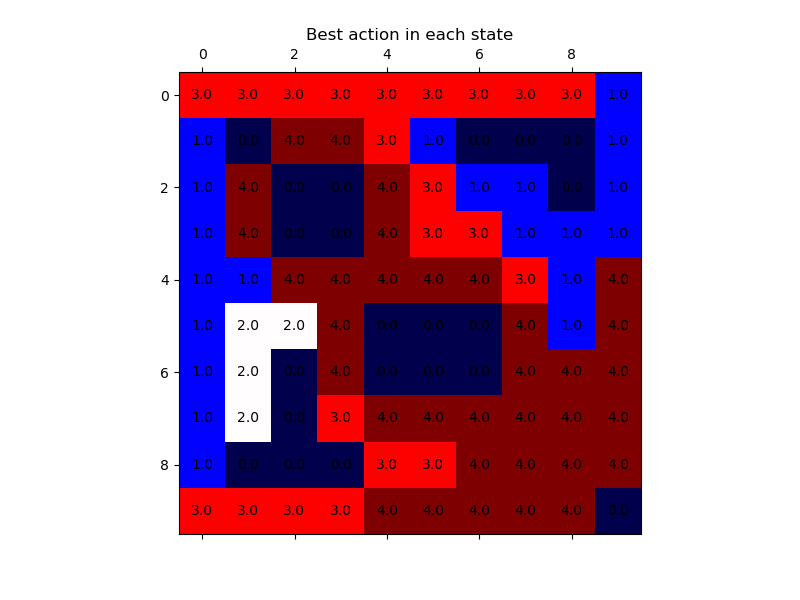
Description automatically generated



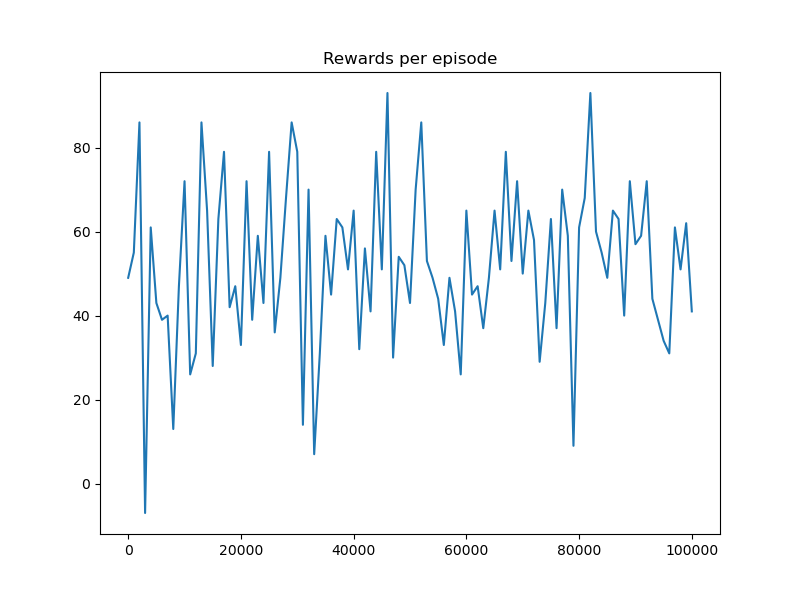
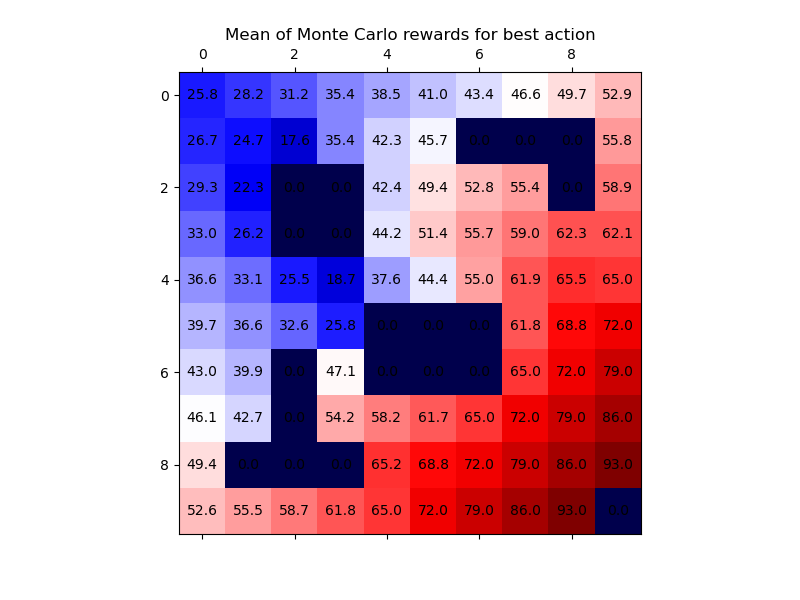
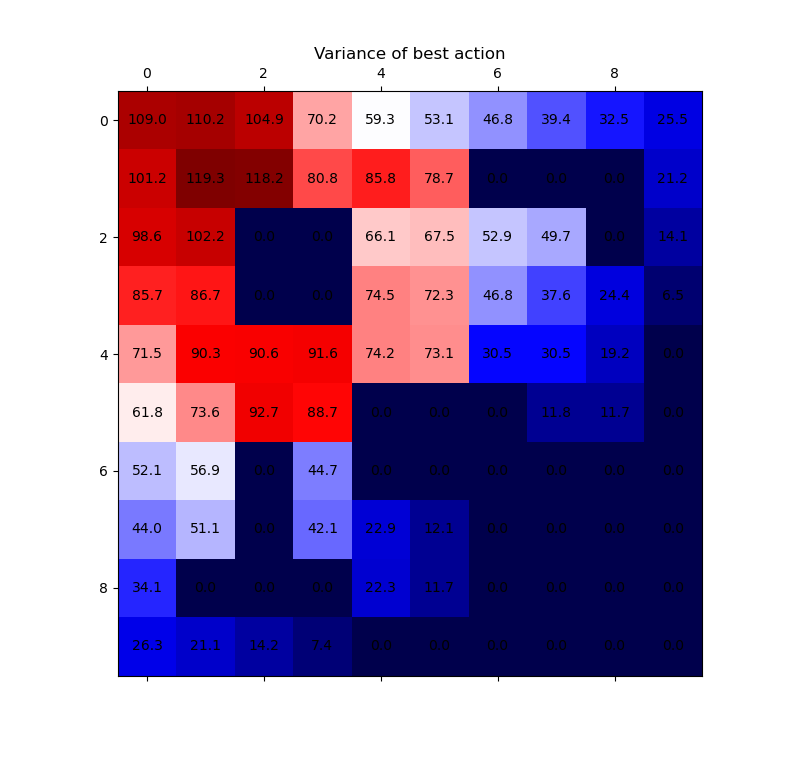


1. **ALG1 –** 
   1. **Trained for 1M episodes:**

****



**Note that action ‘4’ is calling the expert**

* 1. **Monte Carlo Episodes:**

**3) ALG2 – 1 million episodes:**

Things to note: I have calculated variance values using Bellman equations during the training of the agent.

Another possible way I can see is that the q values are found out for each state action using the vanilla training method until they converge. Then mean and variance of rewards are found out by rolling the policy and using Bellman equations instead of Monte Carlo. Please add in the comments if one is better over the other and why?

Chart, bar chart

Description automatically generated

A picture containing table

Description automatically generated

A picture containing treemap chart

Description automatically generated

Chart, treemap chart

Description automatically generated