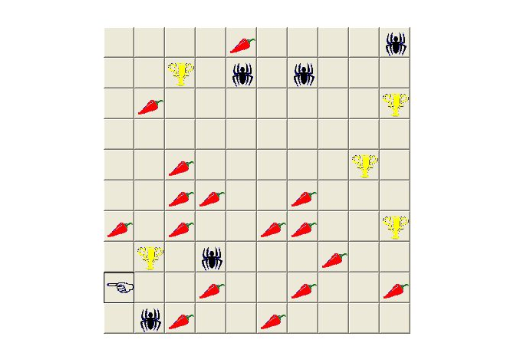
**Ali Hassaan Mughal: Reg#: 173627**

**Advanced Programming: Assignment 1**

**Problem:**

I found it interesting is a basic problem from the reinforcement learning i.e. presently a very active area of research in computer science, and is a step towards getting closer to the functions performed by the brain i.e. action leading to a reward process. The reward may be a positive one or a negative one.

Consider a basic maze of n x n elements where n is considerable to make the computation possible. Some of the blocks/locations have reward consider it to be a treasure and some of them are places where there is something negative consider it be fire/freeze state. Following are the stages I divided this problem into when analyzing it.



**1. Investigate and Understand the Problem**

The problem can be understood as consider each block as a step in the environment, and there is an agent which is assigned a starting position and has to follow a path so that it can acquire most of the reward possible in the maze or environment. It can be considered as getting the best solution out of all. The problem will revolve around getting away from the negative/loss we can get in the environment. We have to traverse all the environment first and then cater for the path leading to that reward or loss in a way that the paths for benefit become clearly visible or differentiable from the paths with loss.

**2. Split into sub-problems**

The problem can be split into subparts of creating an environment, deciding on what type of environment and how to provide rewards and loss in the environment so that it is applicable to real life’s basic solutions. The problem is recursive type basically in nature, unless we use Reinforcement learning algorithms, which has different training schemes for the environment, it uses a reward function as well, we can traverse each path as a subproblem and then the path leading towards the reward is given a positive probability or value and the path to loss is provide with a negative value or is subtracted in value, so that the probability of following that path decreases.

**3. Search for familiarity**

A function for tracing a path can be made and hence the reusability factor can be increased. The function can do the repeated task of finding different paths towards the solution and properly change the value of each step in the environment.

**4. Solve by Analogy**

I take the analogy of this problem from, for example take our real-life example of 100x100 steps. Consider we can only see what is present on the block we are presently in and the next can be tested at maximum, the rest of the path is not visible to us. We need to traverse each path i.e. to visit each step and see if we get a reward following it, when we get a reward we go back on the path we followed to get there and add a sticker on the step with a value for example 0.2 for the closer steps to the reward and 0.1 for those who are far from the reward and 0 to those not included in the path. So that this path is preferred for the next time, and hence the probability of getting the reward increases, and if we reach a state of loss, we decrease the value of each step we followed to get to the loss, decreasing the probability of reaching that state of loss.

**5. Means-Ends Analysis**

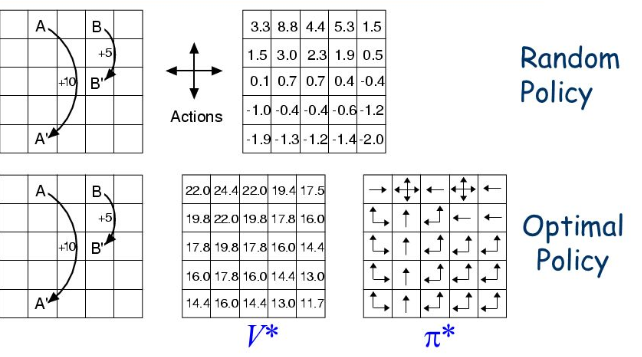
In the means end analysis, our goal is the maximize the action value function, to find an optimal value function or the best possible value action value / value function so that, the agent can take actions in the environment accordingly by following these functions. We know that there maybe some goal to follow by the agent, in the environment, that it has to learn, so learning that task to be performed using the reward and loss functions.

**6. Divide and Conquer**

The traversal can be done by divide and conquer too, if we think of a whole 100x100 grid for example, lets make it 4 50x50 or more as we keep going on so that, each can be solved separately. But the usual approaches don’t use divide and conquer in this type of problem.

**7. Building solutions**

The solution for this problem, was to use the bellman equation to solve this grid world problem.



We have a model which tells us the transition probability, i.e. for example if I move left the probability of moving left in the environment = 0.8 and moving right = 0.2 etc.

In the figure above, we can see V\* i.e. the optimal value function giving the best possible values at each point and pi\* giving the best possible policy i.e. the steps that will be best to be taken in any particular position in the grid world.

The idea iterates over the grid world i.e. n x n grid of steps. We need a policy which tells us the next step at any position, i.e. the transition probability at each step in the grid world, and the value function which tells the value at a specific step, this value function as well as the policy need to be trained, so that the agent knows how to perform in the environment.

The bellman equation used for the value function is:

While for action value function i.e. tells the value after we take a step in the environment:



And the policy is trained using these two types of functions i.e. pi function.