Project Report

Wumpus World

Submitted To

Subrina Akter
Assistant Professor
Dept. of Computer Science and Engineering
IIUC

Submitted By - Group 3

Tehsim Fariha (C193207)
Raisa Ahmed (C193220)
Efter Jahan Ema (C193229)

INDEX

PROBLEM ANALYSIS

INTRODUCTION

In this Lab work we are going to solve the Wumpus problem with an algorithm based on Q-Learning. Before getting through the work, first the Wumpus problem is going to be described .Then in the next step the implementation of the above discussed algorithm to find a solution for the problem is going to be described. At last, the result of the implementation will be shown.

WUMPUS WORLD PROBLEM

The Wumpus world is a game, where a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible Wumpus, a beast that eats anyone who enters its room. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms. The only mitigating feature of this bleak environment is the possibility of finding a heap of gold.

The Wumpus World Game is played on a board with N x N fields where each field on the board can have different states. Field number 1 is always the starting point. The fields could have a combination of the following states:

1- Agent
2- Wumpus
3- Gold
4- Pit
5- Breeze
6- Stench

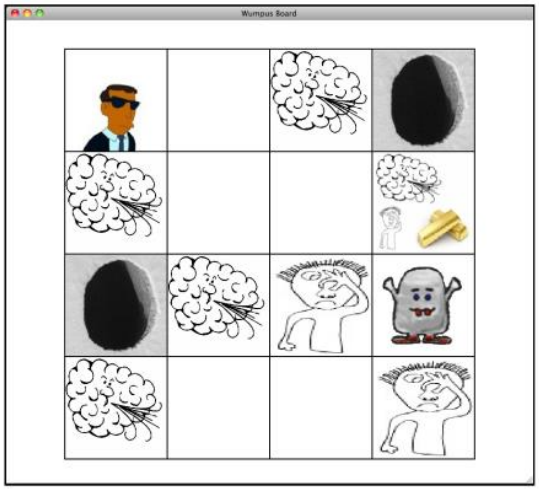


Figure -1

Environment:

An example of a 4x4 game board is shown in figure 1. The start point could only have the Breeze state or Stench state or both or neither of them. If the start point contains a Pit or Wumpus or the Gold, then the game is ended. Therefore, to avoid that, the above assumption has been made. There is only one field which contains the Gold and only one field that contains the Wumpus. For an N x N board, there are maximum N-2 fields that contain a Pit. Pits and Wumpus could be in the same field, whereas the Gold cannot be in the same field where a Pit or Wumpus is present. All of the neighboring fields of a Pit field contain a Breeze and all of the neighboring fields of a Wumpus field contain a Stench.

An agent's precepts, actions and goals are listed below:

• In the square containing the Wumpus and in the directly (not diagonally) adjacent squares the agent will perceive a "Stench".

• In the squares directly adjacent to a pit, the agent will perceive a "Breeze".

• In the square where the gold is, the agent will perceive a "Glitter".

• There are actions to go forward, turn right by 90 degrees, and turn left by 90

degree. In addition, the action Grab can be used to pick up an object that is in the same square as the agent.

• The agent’s goal is to find the gold and exit from the game board quickly as possible, without getting killed.

DESIGN PHASE
&
IMPLEMENTATION

Design Phase:

To solve this Wumpus World problem we choose Q-learning algorithm which is model free reinforcement learning technique. It gives agents learning capability to act optimally by experiencing the consequences of actions, without requiring them to build maps of the domains. We have implemented a learning agent that solves the world through Q-learning.

Pseudo Code for Q-learning algorithm:

1) Set learning parameter γ, and reward matrix R
2) Initialize values of matrix Q to ZERO
3) For each episode:
   - Select random initial state
   - While not reach goal state
     - Select one among all possible actions for the current state
     - Using this possible action, consider to go to next state
     - Get maximum Q value of this next state based on all possible actions.
     - Compute Q value

       $$Q(state, action) = R(state, action) + \gamma \cdot Max[Q(next\ state, all\ actions)]$$
     - Set next state as current state
   - End While
4) End For

Methodology:

Step-1: First we define some set of elements:

- Agent — an "intelligent actor" that can interact with its environment, e.g. a player in a game. It performs 4 actions- right, left, up, down. It sets a start point.
- ROWS & COLUMNS.
- PITS Position
- WUMPUS Position
- GOLD Position

- Reward — incentive (or disincentive) that we give to the agent when it performs desired (undesired) actions at various states. We can reduce future rewards relative to present rewards by using the discount factor {gamma (γ)}.
- Epsilon— enables us to set how much time the agent should spend exploring the environment versus exploiting its existing knowledge about the environment.
- Episode — one complete cycle from the start position to the end position. E.g., in the context of a game, an episode would last from the moment your agent starts a new level until it dies or completes the level.

Step-2: We generate space state & learning parameter. There are 16 states for 4*4 matrix. Each state has a utility.

When it is 0, the agent not learn anything
When it is 1, the agent consider the most recent information.

Step-3: Before learning has started, the value of Q matrix is fixed to 0.

Step-4: Each time the agent picks an action, observes a reward and a new state determined by both the previous state and selected action and Q matrix updated with Q-values.

Step-5: We have to build reward matrix (R) that represent the environment by assigning the values:

If agent move to safe state then the value set to -1.
If agent move to room contains pit then the value set to -1000.
If agent move to room contains Wumpus then the value set to -1000.
If agent move to room contains Gold then the value set to 1000.

Step-6: In Training session phase the agent learn more and more experience through many episodes. After each time the agent reach the goal state a new episode begin.

Step-7: After many episodes the values of Q matrix become convergence from one episode to nest one. When we reach this situation we normalize Q values by dividing it by the highest value in the Matrix. Once Q matrix reach almost convergence value, the agent can find solution in an optimal way.
Step-9: We run the program with learning parameter 0.8 which is best value of learning rate. The range for values in normalized Q matrix give a good indication about which state is best. According this, we achieved the path of destination.

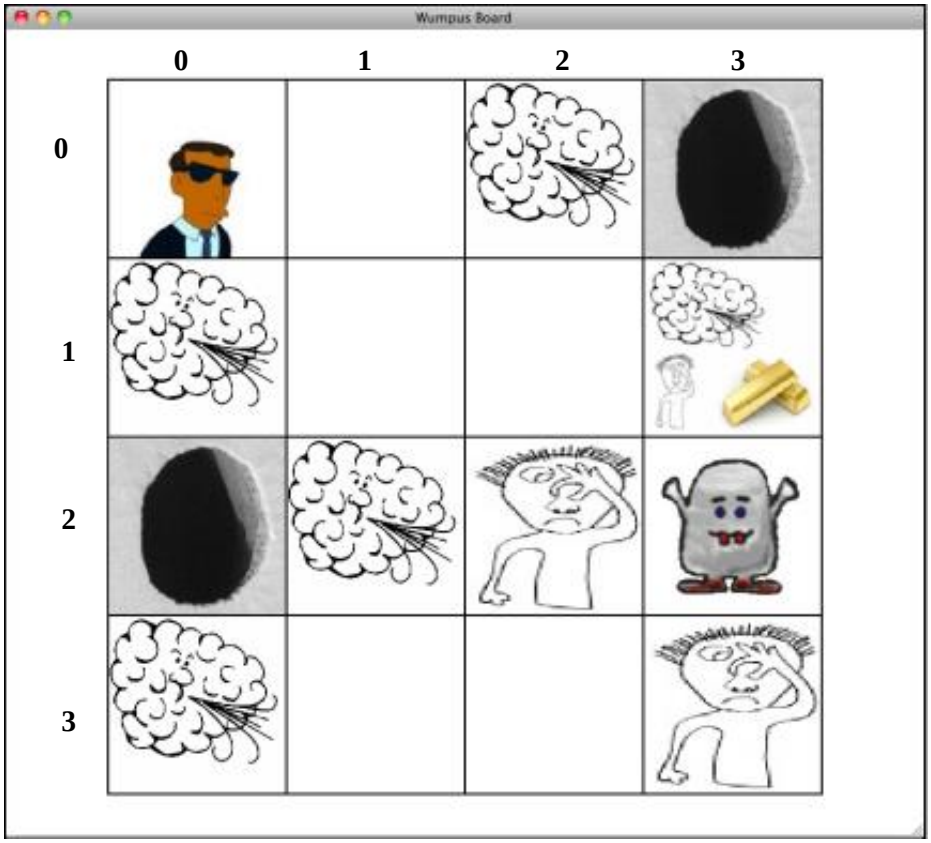
Implementation:

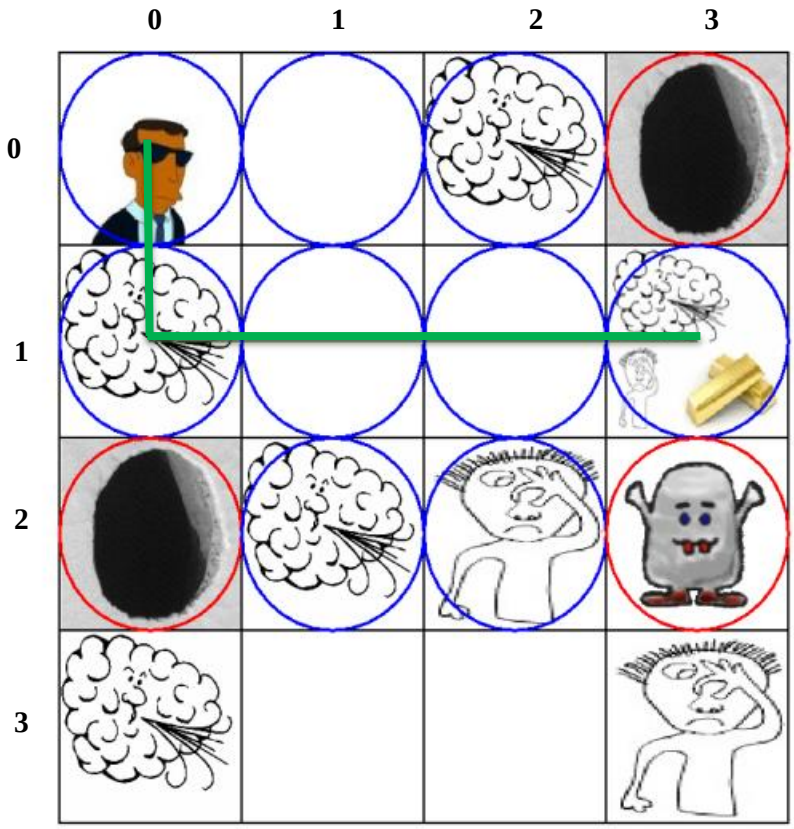
We implemented this problem using python.
https://pastebin.com/xrSkdeGC
https://pastebin.com/hw4jzvNi

RESULT & ANALYSIS

Result:

This is our Wumpus world game board, where agent is located at [0,0] and Gold is located at [1,3]





In our project, the path we got for hunting the Gold is
[0,0]-> [1,0]-> [1,1]-> [1,2]-> [1,3]

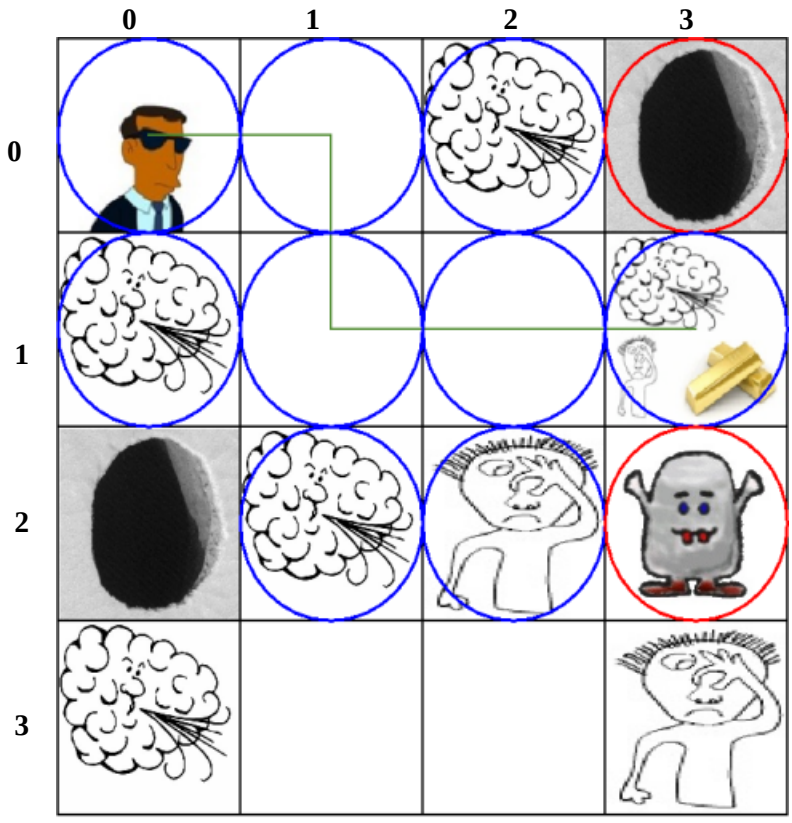
```
main
"C:\Program Files\Python311\python.exe" "D:\6th Semester\AI lab\Project\Wumpus World
[[   -1    -1    -1 -1000]
 [   -1    -1    -1  1000]
 [-1000    -1    -1 -1000]
 [   -1    -1    -1    -1]]
Training Finished!!!
Path to hunt GOLD!!! :
[0, 0] [1, 0] [1, 1] [1, 2] [1, 3]

Process finished with exit code 0
```

Output

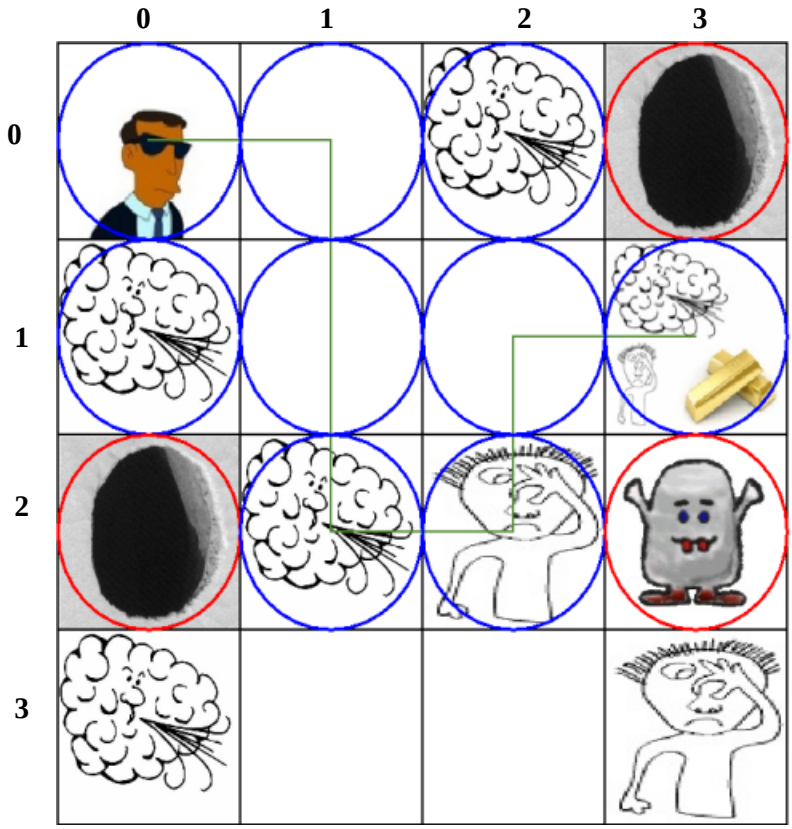
Analysis:

In our project, output generated one path. But according to our Wumpus world, there are other several ways to find the Gold.
Let us analysis those paths.

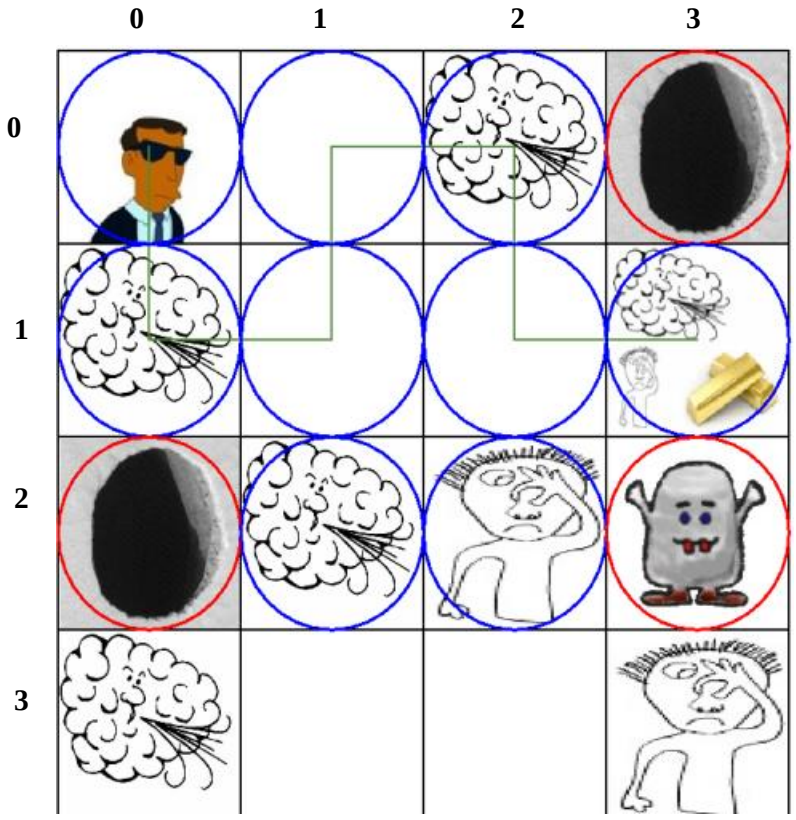
Alternative Possible **Path-1**
[0,0]-> [0,1]-> [1,1]-> [1,2]-> [1,3]



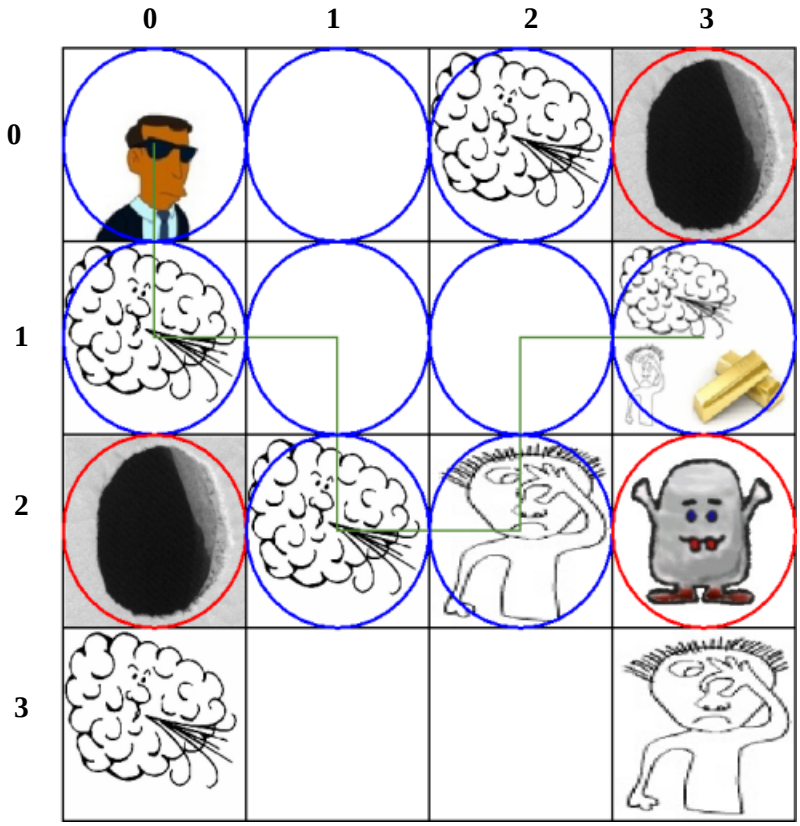
Alternative Possible **Path-2**
[0,0]-> [0,1]-> [1,1]-> [2,1]-> [2,2] -> [1,2]-> [1,3]



Alternative Possible **Path-3**
[0,0]-> [1,0]-> [1,1]-> [0,1]-> [0,2] -> [1,2]-> [1,3]



Alternative Possible **Path-4**
[0,0]-> [1,0]-> [1,1]-> [2,1]-> [2,2] -> [1,2]-> [1,3]



FUTURE SCOPE

In Our Project

- We can add multiple Agents, Wumpuses, Golds.
- Agent can throw arrows.
- Agent can hunt the Wumpus.
- Agent can backtrack its way after finding the Gold.
- We can show multiple possible paths.