CS 511 Project 2:

Utilities-Based Agent for Wumpus World

By Wumpus Slasher

Classes:

WumpusCell: The building block of the model. It records states of a traveled wumpus world coordinate. Each Cell corresponds to a unique coordinate of the wumpus world.

WumpusWorld: It translates percepts into states and records it for later use by modifying the flags of cells and some internal variables. At each step the model is updated with given percepts.
The essential methods of WumpusWorld class are:

1. updateWumpusWorld: Update the wumpus world state based on the current percepts.
2. tryFrontierConfiguration: Try a given frontier configuration and check if this configuration is valid while searching. Where frontier cells are cells that are unvisited and connected to at least one visited cell.
3. killWumpus/reviveWumpus: Kill the wumpus temporarily to calculate the utilities of worlds without living wumpus, and revive the wumpus once the utilities have been evaluated. This is used for deciding whether to kill the wumpus when the agent has an arrow.
4. reviveWumpus:resolveWumpus/resolvePit: Find out where wumpus/pit is whenever the world state is updated.

Detailed explanations are given as comments in WumpusWorld.java.

ForwardSearch: It searches every possible future outcome from the current state and calculates utilities of travali.
The essential methods of WumpusWorld class are:

1. simulate: Try every frontier configuration and calculate utilities for each frontier cell with backtrack algorithm. Frontier cell with the highest utility is a good choice to visit first.
2. calcUtility: The mastermind of a utility-based agent. This method calculates the utility of a frontier cell. Utility is designed to be close to the sum of the highest scores the agent can get from every observable wumpus world with that configuration. (ex. 1 wumpus no pit, 1 wumpus 1pit etc..)
3. findShortestPath: Find the shortest path to frontier cells by using breadth first search algorithm. A* algorithms can be adopted but it doesn't give noticeable improvement, so I keep BFS for easy understanding.
4. fwdSearch: Wrap up the results obtained from simulate and findShortestPath then return the best destination. This allows the agent function to decide what to do next.

Detailed explanations are given as comments in ForwardSearch.java.

Agent Function: This class takes precepts as input and updates the world state based on percept at each step.
The essential methods of Agent Function class are:

1. nextDestination: Determine the most viable cell destination, (destX, destY), including current position based on the state of the world. If shooting at a wumpus is a better choice, this function will return (-1, -1).
2. nextAction: Output the corresponding action based on the destination cell chosen/

StateSnapshot: Backup a state before it is changed. This class and wumpusWorld.backTrack are used in ForwardSearch.simulate for backtracking.

Setup:

I commented out some of the print functions to make the simulations run faster.
The following lines in WorldApplication.java have been modified to meet the environment requirements.

line 36: int numTrials = 10000;
line 39: boolean nonDeterministicMode = false;
line 40: boolean randomAgentLoc = false;

Results:

The average score for a 100000 trials run is 528. (see wumpus_out.txt). For 10000 trials runs, the scores fluctuate from 520 to 538.