Journal of Machine Learning Research 1 (2021) Submitted ; Published NA

Wumpus World Project Report

Cooper Strahan COOPERSTRAHAN@GMAIL.COM
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Riley Slater RILEYMAIL6@GMAIL.COM
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Editor:

Abstract

Artificial Intelligence agents need to be able to make accurate inferences about their environment in order to make logically sound choices surrounding their actions. In the Wumpus World problem, an explorer needs to make decisions based on environmental factors that prevent the explorer from dying and help the explorer find its goal state. By codifying first-order logic rules, a Wumpus World explorer can make accurate inferences about it's environment in order to make its current best choice towards finding a goal state. Expanding on this technique can allow more complex agents to make inferences about their own environments. By applying first-order logic, agents are learning without explicitly experiencing. This is the first step towards creating a truly artificially intelligent being

Keywords: Logic, First-Order Logic, Wumpus World, Agent, Inference

1. Introduction

The Wumpus World problem was inspired by the video game **Hunt The Wumpus** created in 1973. The basic premise of the problem is that an explorer is placed in a cave that contains nxn cells, where n is representative of the length and width of the cave. The cave contains Wumpuses, pits, obstacles, and a bar of gold. Wumpuses emit a stench and pits emit a breeze that can be smelled / felt in the cells North, South, East, and West of them. Wumpuses and pits both kill the explorer if the explorer enters the same state as one of them. The explorer is armed with as many arrows as there are Wumpuses, and can use arrows to kill Wumpuses, removing them from the game, or turning them into general obstacles dependent on the games implementation. Obstacles prevent the explorer from moving into the same state. The goal of the explorer is to find the gold without dying. The gold state is thought of as a goal state, and can be assumed to have a ladder to escape the cave after finding the gold. The cave is completely dark and the explorer can only use his senses to move through each cell in the cave.

In order to find a goal state without perishing the explorer needs to make logical inferences about Wumpus and Pit states based on its perception of stenches and breezes. Codifying the information in a Wumpus World board into first-order logic rules allows us to teach an explorer about states without ever having to enter them, helping the explorer find the gold state and avoid states that would kill it.

In our project we designed and implemented a logic based agent that applied propositional logic based resolutions to create accurate inferences about it's discrete static environment. We also implemented an agent that acted randomly on the world attempting to find a gold state.

2. Problem Statement and Hypothesis

The problem addressed in this paper is how to teach a knowledge based agent how to traverse a board in search of a gold state while attempting to avoid certain states. One algorithmic approach was taken, in that an agent created a knowledge base and then utilized a propositional logic based algorithm (PL-Resolution) to determine dangerous and not dangerous states.

A PL-Resoluton algorithm prevents an explorer from dying. The PL-Resolution if implemented without the possibility of sending an explorer into an unsafe state, would create infinite loops that stopped the explorer from leaving already visited "safe" states. With prioritization applied to PL-Resolution algorithms, they can be taught to explore unknown states over known states. This allows prioritized PL-Resolution algorithms to always find the goal state if given enough time to process the board.

2.1 Hypothesis

The knowledge based explorer running the prioritized PL-Resolution algorithm will outperform the simple explorer running the random algorithm in all categories. As the board gets larger both the knowledge based and simple explorer will find the goal state less often, as there are more safe spaces which will slow forced exploration.

3. Algorithm Description and Implementation

Our Wumpus World implementation was broken into two parts. The Wumpus World GameBoard was built as a two dimensional array made up of Cell objects. We passed probabilities to the GameBoard to determine the number of Wumpuses, pits, and obstacles that exist in the board. These probabilities were multiplied by the number of squares in the board to create a concrete number for each item. We then randomly asserted these objects onto the board, making sure that none of these items would be assigned to the same cell. We also randomly inserted a gold state, also making certain it was not in conflict with one of the penalty states.

Our Cell objects contained all of the important information that was accessed by the explorer. They maintained their indices, adjacency lists, if they had been visited, and the state. The state was represented as a dictionary of boolean values that indicated the items located in the state as well as Stenches and Breezes. The state also allowed states to be settable by the explorer in its copy of the board.

Our implementation of the Explorer was in two objects, the primary object being the SimpleExplorer which kept track of position, arrows, gold, and other relevant metrics. The simple explorer used a random function that made the SimpleExplorer turn and move forward. The simple explorer followed this pattern, shooting an arrow (using the shootArrow() function) if it has one with a 2% probability each turn, appending each new visited state to an array. If the explorer reached a pit or a Wumpus the program would terminate with a cost of -1000 and if it reached the Gold state it would terminate with a cost of +1000. This process was used for the exploration of all the boards with the simple explorer.

The knowledge based explorer named Explorer, was a child of the SimpleExplorer. The knowledge based explorer contained a GameBoard object of its own that it filled out as it explored. As each new state was reached as did the SimpleExplorer, the Explorer added new states to its visited state array. Explorer would read in state from the GameBoard object, it would first determine if the state it was in was a safe state. It would then check if it was in a winning state. As our implementation did not guarantee that the Explorer was placed in a safe / non winning location, the Explorer could be killed or win the game upon initialization. The explorer reads in information about its current state, attempting to infer any information that it could about the surrounding states by checking for stenches, and breezes. If a stench or breeze was detected, the explorer utilized the prioritized PL-Resolution algorithm based on the (Russell, Norvig 2022) PL-Resolution algorithm to attempt to prove a Wumpus or pit [RN22].

Our PL-Resolution algorithm read through the array of visited states, constructing a knowledge base that contained relevant information pertaining to the state found. The algorithm then asserted the state of potential Wumpus or potential pit onto the adjacent states from which it had found the stench or the breeze. If a cell had previously been determined safe, from the visited states array, it did not get the new designation. Note that the state is being read from the actual game board and written to the explorer's board. A reduction was then attempted on the cells that have been determined as potential Wumpuses and potential pits. The knowledge base was used to attempt to cancel any of the potential pits based on the previous states and their adjacent states. The reduction attempted to reduce the number of potential Wumpuses / pits to 1. If it was able to successfully complete the reduction, we had proved a Wumpus or pit! We then asserted the proved state onto the explorer game board. The resolution algorithms were done in the provePit() and proveWumpus() functions owned by the Explorer class. If we are able to prove a wumpus, we shoot it with an arrow by turning towards it and using the shootArrow() class.

After the explored has inferred what it could through the knowledge base and the PL-Resolution algorithms, the Explorer attempts to determine its best next move in a function named findBestMove(). This is the prioritization piece of our algorithm that pushes unsafe values, allowing for wider exploration of the Wumpus World GameBoard. The function creates an array of potential moves, it does this by first adding "safe" adjacent spaces to the array that have not yet been visited. A safe adjacent space is defined as a space that does not contain a proven pit or Wumpus. If there are no safe spaces, we append previously visited safe spaces to the array. If none of those exist, which is rare, then uncertain safe spaces are added to the array. Uncertain safe spaces are spaces that contain potential pits, or potential Wumpuses. These are generally spaces where we do not yet have enough information to prove a Wumpus or pit. From this array, if there exists a value or values in the array we randomly select a value from the array, turn and move into the space. If no value exists in the array, our explorer randomly turns and moves the same way as the SimpleExplorer did.

After reaching a new state, the explorer repeats the above process. We made a design decision to limit the total number of moves an explorer can make equivalent to the size of the board to prevent infinite looping. This limit potentially prevents the explorer from finding a solution but keeps run-times fast.

In order to determine performance on a board each action performed by an agent would need to have a cost. The agent's cost started at 0. If an agent turned or moved forward, the cost was reduced by 1. If an agent died by pit or Wumpus the cost was reduced by 1000. If an agent shot an arrow, its cost was reduced by 50. If an agent found a gold state its cost was increased by 1000.

4. Experimental Approach

In order to determine the effectiveness of our prioritized PL-Resolution based algorithm, we ran a random based explorer and a knowledge based explorer against 100 boards for each of these sizes; 5x5, 10x10, 15x15, 20x20, 25x25. We kept track of Cost, Moves, Wumpuses Killed, Deaths by Wumpus, Death by Pit, Gold Captured, Number of Visited Cells.

We created Wumpuses, pits, and obstacles all with a probability of 0.05, meaning one of each were created for the smallest board and 25 of each were created for the largest board size.

5. Results

Each table represents the summary statistics surrounding the 100 iterations ran on the boards. The statistics chosen were the best and worst performing explorers, the average score on the board size, and the std deviation of each category. These categories were broken into the metrics that seemed most appropriate to describe explorer performance.

Simple Explorer Results

5 x 5 board

Metric	Min	Max	Average	Std. Dev
Cost	-1098	1000	-319.5	826.58
Moves	0	26	13.5	9.62
Visited Cells	1	14	6.04	3.47
Wumpuses Killed	0	0	0.0	0.0
Death by Wumpus	0	1	0.25	0.44
Death by Pit	0	1	0.27	0.45
Gold Captured	0	1	0.24	0.43

10 x 10 board

Metric	Min	Max	Average	Std. Dev
Cost	-1262	1000	-905.14	499.0
Moves	0	101	24.07	25.84
Visited Cells	1	31	8.77	6.84
Wumpuses Killed	0	1	0.06	0.24
Death by Wumpus	0	1	0.34	0.48
Death by Pit	0	1	0.56	0.5
Gold Captured	0	1	0.06	0.24

15 x 15 board

Metric	Min	Max	Average	Std. Dev
Cost	-1474	1000	-930.38	529.27
Moves	0	162	24.44	31.06
Visited Cells	1	48	9.6	8.99
Wumpuses Killed	0	2	0.13	0.42
Death by Wumpus	0	1	0.48	0.5
Death by Pit	0	1	0.45	0.5
Gold Captured	0	1	0.07	0.26

20 x 20 board

Metric	Min	Max	Average	Std. Dev
Cost	-1658	998	-1044.84	308.74
Moves	0	204	27.17	31.54
Visited Cells	1	48	10.61	9.26
Wumpuses Killed	0	2	0.17	0.49
Death by Wumpus	0	1	0.47	0.5
Death by Pit	0	1	0.51	0.5
Gold Captured	0	1	0.02	0.14

25 x 25 board

Metric	Min	Max	Average	Std. Dev
Cost	-1598	994	-1063.08	231.7
Moves	0	224	27.54	33.23
Visited Cells	1	42	10.4	8.62
Wumpuses Killed	0	3	0.29	0.73
Death by Wumpus	0	1	0.51	0.5
Death by Pit	0	1	0.48	0.5
Gold Captured	0	1	0.01	0.1

Explorer Results

5 x 5 board

Metric	Min	Max	Average	Std. Dev
Cost	-1012	1000	515.3	647.51
Moves	0	26	13.99	9.69
Visited Cells	1	21	10.12	5.87
Wumpuses Killed	0	1	0.11	0.31
Death by Wumpus	0	1	0.03	0.17
Death by Pit	0	1	0.05	0.22
Gold Captured	0	1	0.63	0.49

10 x 10 board

Metric	Min	Max	Average	Std. Dev
Cost	-1000	1000	79.35	618.39
Moves	0	101	69.78	41.13
Visited Cells	1	70	23.44	16.81
Wumpuses Killed	0	5	0.39	1.07
Death by Wumpus	0	1	0.01	0.1
Death by Pit	0	1	0.07	0.26
Gold Captured	0	1	0.32	0.47

15 x 15 board

Metric	Min	Max	Average	Std. Dev
Cost	-1017	1000	-164.45	670.1
Moves	0	226	155.22	98.02
Visited Cells	1	108	34.2	27.23
Wumpuses Killed	0	12	0.57	1.69
Death by Wumpus	0	1	0.04	0.2
Death by Pit	0	1	0.05	0.22
Gold Captured	0	1	0.26	0.44

20 x 20 board

Metric	Min	Max	Average	Std. Dev
Cost	-1108	944	-798.61	307.79
Moves	0	401	338.34	144.53
Visited Cells	1	137	31.98	29.19
Wumpuses Killed	0	6	0.49	1.2
Death by Wumpus	0	1	0.09	0.29
Death by Pit	0	1	0.04	0.2
Gold Captured	0	1	0.03	0.17

25 x 25 board

Metric	Min	Max	Average	Std. Dev
Cost	-1456	998	-975.39	725.44
Moves	0	626	500.2	245.85
Visited Cells	1	194	43.17	45.32
Wumpuses Killed	0	8	0.76	1.69
Death by Wumpus	0	1	0.05	0.22
Death by Pit	0	1	0.03	0.17
Gold Captured	0	1	0.13	0.34

6. Analysis and Discussion

Consistent with our Hypothesis the knowledge-base explorer outperformed the random explorer in all metrics, for all of the boards. Both scores also got worse with increased board size. The allowance of backtracking to safe cells allowed our explorers to stay within a grouping of safe cells rather than forcing them to venture into unknown cells.

Our inference logic was generally able to keep our knowledge based explorer safe, with some exceptions. Future work could include a more robust optimization function that encouraged our agent to act with a more path finding based approach.

7. Conclusion

In this paper we created an agent that utilized inference knowledge to make accurate predictions about its environment.

References

[RN22] Stuart J. Russell and Peter Norvig. “Chapter 7 Logical Agents”. In: *Artificial Intelligence: A modern approach.* Fourth. Pearson Education Limited, 2022.