AI 1 report

160020220

October 2019

Contents

1 Introduction

In this practical, I was asked to Implement AI agents to play the game tornado sweeper. These agents operate on a NxN map with hexagonal cells. The goal here is to uncover all cells on the board but those containing a Tornado. If the agent probes a cell that contains a Tornado, the game is over. Otherwise, a clue appears on the cell indicating the number of Tornadoes in the 6 adjacent neighbours of the probed cell.
3 Logical agents were to be implemented in this practical. However, due to time constraints cause by a foot injury. I completed the random agent and the single point agent. I will try to show my understanding of the SAT agent in this report though!
(I have sent and email to the masters coordinator explaining my situation regarding my situation)

2 Design

2.1 Running Environment

Main method and Argument

To run the program, First compile it in the source folder with the following command:

javac Main.java
Then run it using :
java Main followed by the arguments according to this usage:
Stategy — Map

The Main method takes these arguments to first create the map where the Agent's game will take place. We then run the appropriate Agent algorithm according to the first argument args[0].

We use a try catch block in case we have wrong inputs.

Program's running logic

After getting all the info we need from the arguments. We start by creating the knowledge base where we store a map of the right dimensions populated with hidden cells. We use this knowledge base for the agent's processing. We then create the appropriate agent using this knowledge base.
Then, we make the agent play the game, printing the probed cells and the game state.
All agents start by making a first move. As we know, the top-left cell and the middle cell are safe cells. Thus, the agent starts by probing those. Afterwards, the agent starts applying the specific chosen strategy.

2.2 Strategy

Random Search Strategy

In this strategy, the agent probes the cells randomly. If the agent find a tornado, the game is lost and another run is attempted. after X amount of runs, the agent will succeed and return the discovered map. Remember, the winning condition is:
If the number of remaining cells is the number of tornados, then the agent wins.

Single Point Strategy

The strategy looks at a probed point. If the number of tornados near the point equals the number of marks near the point, the strategy infers that near points whose status is unknown do not contain tornados. Similarly, if the number of tornados near the point equals the number of marks near the point plus the number of unknowns near, the strategy infers that the near points whose status is unknown contain tornados.
I extended the single point strategy to make decisions also based on equations when pairing 2 cells a and b where a is a strict subset of b and vice versa. We then determine the number of tornados that must be in the set difference of the neighbours.

SAT Strategy

The core mechanism of the SAT strategy is to convert current Tornado sweeper board into a CNF, and resort to SAT4J, a SAT solver seen in the lectures, to get the solution. Due to lack of assumptions, or due to the limits on the number of CNF clauses, it often occurs that a number of possible solutions are returned. The goal now is to find the symbol that maintain its set/clear state the most throughout all solutions. If that symbol is always having Tornado underneath or otherwise, then it's definitely that state. If that symbol is mostly one state but sometimes the other, then the former state is a better guess. Although it's the best guess one can make, still it sometimes loses.
How to list CNF clauses is straight forward. Let x1, x2, x3 be three cells surrounding a cell labeled 2. Then x1+x2+x3=2. Since there is either tornado under a cell or no tornado under the cell, the three variables are either 0 or 1. Thus we may enumerate all possible assignments to x1, x2, x3, and write them as DNF clauses. After that, we may convert DNF to CNF. Although it's NP-complete, we can precompute DNF-to-CNF templates.
An Idea to reduce the number of CNF clauses would be to use a form of SAT encoding of cardinality constraints. Thus, removing the need to precompute DNF-to-CNF conversions at the expense of introducing some auxiliary boolean variables.

3 Testing

The first thing we test in the Map Generation. We use the printing method in Board, to check this.

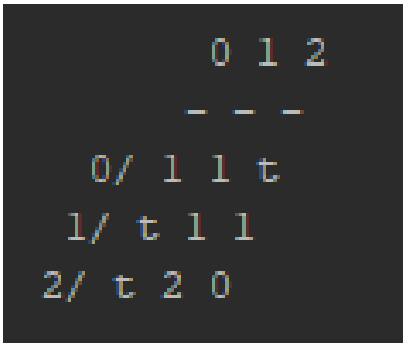

Figure 1: Map generation

Then, We have to check if the 2 starting moves are done appropriately. As we can see, the agent probes the top-left node as well as the middle node

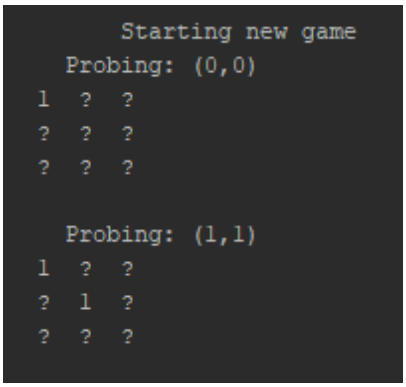

Figure 2: Start Move

This is the testing of the Random Strategy. the agent probes cells randomly. When discovering a tornado, the game is lost and the agent starts a new game!

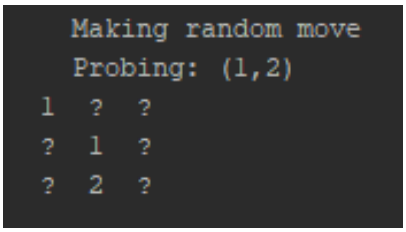

Figure 3: Random Strategy

After Many runs, The game is finally beaten.

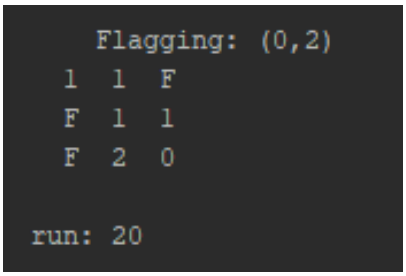

Figure 4: Random Win

Now testing the Single point strategy, we see that the agent is checking cells then checking the cells pairing. Afterwards, it chooses the appropriate move.

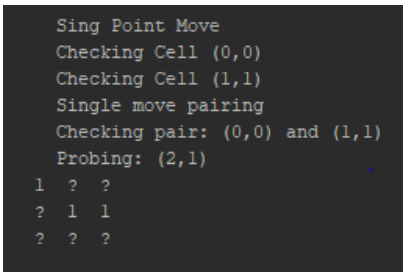

Figure 5: Single point move

In 1 run, the agent beats the game as expected.

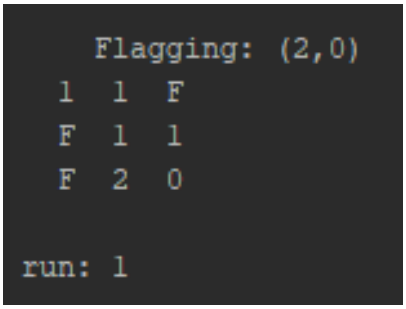

Figure 6: Single point win

The same kind of tests have been run with other maps with various difficulties! The Single point strategy without the equation extension has also been tested and works perfectly!

4 Conclusion

In conclusion, I've been able to implement some features into my AI agent.
I am very happy with the code quality and the implementations I managed to attempt! However, I am a bit disappointed as I could not attempts the other features proposed in the specification sheet.