A4 – CS4300 Lab Report – 10/26/17

Hybrid Agent using both RTP and Ad Hoc Logic

1. Introduction

The hybrid agent demonstrated in *Artificial Intelligence: A Modern Approach* is one that makes use of propositional logic in a large knowledge base to make informed decisions. Such an agent is able to collect percepts from its environment, *Tell* the knowledge base of these percepts, and then *Ask* the knowledge base for information about its world before making a decision to act. The *Ask* part of this cycle is where an RTP (Resolution Theorem Prover) runs through the knowledge base to make inferences based on existing logical statements and create new logical statements. From there, the agent is able to make decisions based on its collected knowledge using traditional if/else logic. The agent attempts to create a plan to proceed for each potential option, only moving on to the next option if the first option was unable to create a plan. The options are as follows:

1. If glitter is perceived, pick up gold then return to [1,1] using A* search and climb out
2. If there are unexplored known safe locations, go explore one of them using A* search
3. If you have an arrow and a Wumpus might still exist, attempt to kill it using A* search to line up the shot
4. If there are unexplored locations not known to be unsafe, go explore one of them using A* search
5. If there are no other options, return to [1,1] and climb out, cutting your losses.

In theory, the RTP should be able to prove things like “is location [x,y] safe, unsafe, or unknown?” and so our agent can at best know for certain if a location is safe, or at worst, attempt to explore an location of unknown danger. This makes it a much more appealing option than an agent using random exploration techniques as seen in previous assignments. The downside of using an RTP is that it can be absurdly slow for large knowledge bases, and it only gets slower as more knowledge is added. This brings up a few important questions, namely:

1. In what situations does the RTP act more slowly than others?
2. In similar situations, how does the size of the knowledge base affect the running time?
3. Are there better ways of achieving similar results?

2. Method

To answer these questions, I created two hybrid agents. The first used the *Ask* method to inquire of the knowledge base whether or not neighboring locations were free of pits and Wumpi. To keep the knowledge base as small as possible, new percepts were only added to the knowledge base when a new square was visited. To keep calls to *Ask* at a minimum, the agent would only call *Ask* when it had taken the move action last cycle and only *Ask* about locations that were currently of unknown safety. Lastly, I placed code in the RTP that would limit the amount of clauses to be tested each use of RTP. The second agent used what I refer to as Ad Hoc logic and never made calls to *Ask*. Its knowledge base did not use propositional logic, but instead consisted of maps of percepts. When encountering a new square, the ‘Pits’ and ‘Wumpus’ maps would be updated according to the ‘Breeze’ and ‘Stench’ percepts respectively. A percept of these types would decrement a counter in neighboring locations on the

respective maps. A lack of percept of these types would increment a counter in neighboring locations on the respective maps. The agent would conclude that a location was safe if the values for a location [x,y] in both maps were both at least 0. Again, repeat visits to locations would not update the percept maps. Though, the agent would attempt to reconsider the safety of unknown neighboring locations upon revisiting them since new knowledge might be available.

3. Verification

In order to verify that the RTP was indeed working properly, I decided to test against knowledge outside of the use of the agent to see that it was giving me proper results. In order to verify that my *Ask* function was working, I asked these questions:

1. A statement known to be true: [B11 v -P21] / [1, -34].
    a. The *Ask* returned true, indicating a correct answer.
2. A statement known to be indeterminate: [S22] / [54].
    a. The *Ask* returned false, indicating a correct answer.
3. A statement known to be false [B11 v P21] / [1, 34].
    a. The *Ask* returned false, indicating a correct answer.
4. Various boundary cases:
    a. []
        i. The *Ask* returned false, indicating a correct answer

In order to verify that my *Tell* function was working, I *Tell*ed the knowledge base these sentences:

1. A sentence which would allow an inference: [P12] / [34] should allow the RTP to infer that [B11] / [1] is true.
    a. After *Tell*ing the knowledge base [34], it was able to infer that [1] was true as expected.
2. A sentence with many clauses to confirm that they were all added to the knowledge base.
    a. After *Tell*ing the knowledge base a sentence with 80 unique clauses, the size of the knowledge base changed from 402 clauses to 482 clauses, each added clause matching the *i*-th clause of the new sentence.
3. A sentence which would make the knowledge base inconsistent: [B11] , [-B11] / [1] , [-1].
    a. After *Tell*ing the knowledge base the 2nd sentence, it accepted it, showing undefined behavior for adding inconsistent knowledge.

4. Data

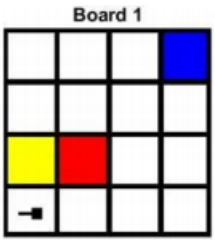

Figure 1

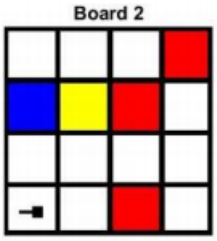

Figure 2

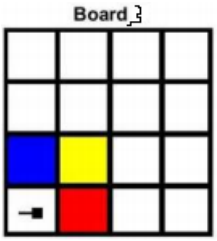

Figure 3

Board	Time to complete in seconds. RTP	Time to complete in seconds. Ad Hoc
1	233.5268	0.1665
2	315.9479	0.0681

3	94.7210	0.0409

Board 1

Location	Time to *Ask* in seconds
[1,1]	55.278
[1,2]	135.3634

Board 2

Location	Time to *Ask* in seconds
[1,1]	57.0538
[1,2]	124.5151

5. Analysis

It seems that having to determine whether or not a location is safe based on inferences is a large cause of long running time when using the RTP. The time it took for the various boards to complete were all very large. Though it could be said Board 3 ran "fast" in comparison to Board 2 when looking at the RTP data, it in fact ran very slowly compared to the Ad Hoc agent.

6. Interpretation

The RTP, by itself does have the advantage of being very safe for the agent, however, the computational complexity seems to be the barrier that holds it back. It more than just a few cases, the ability for the agent to confidently determine a safe location is impossible. IE, in Board 2, it is impossible for the agent to determine that location [4,2] is safe if it is at location [3,2]. Were the gold to be located in location [4,1], the agent would have to take a risk in order to get it. In addition to this, the Ad Hoc version of the hybrid agent uses is able to complete the boards leagues faster than the RTP heavy hybrid agent. The downside of the Ad Hoc agent is that it requires many more lines of manual coding, and could potentially involve many more lines of code if more percepts were added. The RTP does a lot of work by itself on the other hand, just very slowly in comparison.

7. Critique

My biggest critique of this experiment is the large amount of time it took to run each experiment. I knew going in that RTP wasn't the fastest algorithm in the world so I made adjustments to it to try and optimize the number of times it would have to be called in addition to limiting the number of clauses it would test. It still ran too slowly to efficiently test. In the future, researching more optimizations or just allotting a vast quantity of time for testing would be necessary for further research on RTPs.

8. Log

I spent approximately 3-4 hours writing the code for the agents, including the amount of time writing the original code and adapting it to pure Ad Hoc code. I spent approximately 8 hours debugging the code. I spent approximately 3 hours testing.