2nd Lt David Crow

ENG/20M

Assignment #1 – Agents and Uninformed Search

Artificial Intelligence - CSCE 523

Turnin: E-mail me a zip file containing your typed solution to questions 1 through 5, and your program and report for question 6.

1. (10 points) What is intelligence? There are several research views that can be taken to understand intelligence. For the purpose of this discussion, let us label them philosophical, physicalism, and psychological. Almost every AI algorithm we will discuss is related to one or more of these views. Read the Stanford Encyclopedia of Philosophy articles on the Theory of Mind topics of Qualia and Physicalism (<http://plato.stanford.edu/entries/qualia/>, <http://plato.stanford.edu/entries/physicalism/>), and the Sweller paper on Human Cognitive Architecture (<http://www.csuchico.edu/~nschwartz/Sweller_2008.pdf>) (note: this is the closest I have found to a good short discussion on cognitive psychology and the modeling of it).
   1. Which view do you feel is more correct? And why?

In my opinion, physicalism is not an apt model because not everything is inherently physical or a property of something physical. Metacognition, for example, is a property of a property of something physical. At some point, adding multiple levels (like in that example) divides the physical from the non-physical. Besides, I don’t think defining everything as physical allows for necessary detail in describing the world.

Furthermore, I don’t think the idea of qualia is completely sound. I am of the opinion that qualia seek to over-simplify some of the complex pieces of our world. Feelings, for example, aren’t necessarily discrete; as Marvin Minsky says, “The big mistake comes from looking for some single, simple, ‘essence’ of hurting, rather than recognizing that this is the word we use for complex rearrangement of our disposition of resources.” Additionally, I think that some things simply *are*. For example, green grass is green, regardless of an individual’s experiences with grass. I don’t think that the individual’s qualia indicate that green grass isn’t green; it just might not *look* green to that individual.

I feel that Sweller’s view – that is, the idea that cognitive psychology is a sufficient explanation for the human psyche – is the most accurate. He explains five tenets of a natural information processing system, and I think these tenets provide the necessary amount of detail to adequately describe our world and how we fit in it. Furthermore, I feel that Sweller’s Cognitive Load Theory provides a sufficient explanation of some of the limits of the human mind.

* 1. Given your choice, is an artificially-generated intelligence possible? Why or why not?

I believe we can create an artificial intelligence using the five tenets of cognitive human architecture as described by Sweller. Current computers already contain long-term and short-term “memories,” and we are certainly able to manipulate the information in and the information that flows between these different pieces of memories. If we assume that Sweller’s perspective is an accurate description of the human mind and of its interactions with the world, I think that constructing an artificial intelligence using cognitive human architecture will eventually prove successful.

1. (15 points) Testing for intelligence: Read Turing’s original paper (available online at: <http://www.abelard.org/turpap/turpap.htm>). In the paper, he discusses several potential objections to his proposed enterprise and his test for intelligence. Also, refer to the discussion on the Chinese Room Argument found in the text, slides, and at <http://plato.stanford.edu/entries/chinese-room/> or video: <http://www.open.edu/openlearn/history-the-arts/culture/philosophy/60-second-adventures-thought?track=04fb8b569>.
   1. Which objections still carry some weight? Are his refutations valid?

The objection that still carries the most weight is, to me, the argument from consciousness. Effectively, this objection says that machines can’t actually think or feel. I think it’s quite clear that machines can’t think or feel; some machines can mimic those feelings (Furby comes to mind), but approximations aren’t always reality.

Turing refutes this argument by claiming that one cannot confirm that a machine is or is not thinking without *being* that machine. I don’t think that’s valid. Open-source software can be analyzed. Proprietary software can be reverse-engineered. We can communicate with some of the more developed machines. In no case do we have clear evidence of a thinking or feeling machine.

Furthermore, Turing provides an example conversation and asks what a proponent of this objection would say if they overheard this conversation between a man and a machine. However, as far as I know, we don’t yet have machines capable of conversations like the example. Chatbots, for example, seek to emulate this behavior, but they’re not perfect.

Overall, I think this objection still holds water.

* 1. Can you think of new objections arising from developments since he wrote the paper?

In Alan Turing’s time, computers and research in the field of artificial intelligence were still new. In 1990, Robert French claimed that the Turing Test is simply too hard to achieve in the foreseeable future. He argued that we are clearly not close to a machine capable of passing the Turing Test, and that no machine will be able to pass the test at any point because no machine can effectively replicate the human consciousness. His claims stem from 40 years of continued development of computers and AI research. The lack of significant progress\*\* towards passing the Turing Test illustrates that, perhaps, we can’t devise any machine capable of succeeding at the Imitation Game.

\*\* I don’t mean to say that we haven’t progressed. I only mean to say that today’s machines (and, especially, those in 1990) aren’t significantly closer to *passing the Turing Test* than those in the ‘50s. AI has come a long way, but we still can’t reasonably approximate thought in such a way that allows a machine to pass the Turing Test.

* 1. Do you think that there is a better test that could be proposed?

The Stanford Encyclopedia of Philosophy details some potential alternatives to the Turing Test at <https://plato.stanford.edu/entries/turing-test/#ChiRoo>. It also details, however, valid criticisms of each alternative.

I feel that the Lovelace objection in Turing’s original paper is a valid critique of the Turing Test. This objection claims that machines can only do what humans have taught them to do, and I tend to agree with this viewpoint. For this reason, I’m a fan of the Lovelace Test. Essentially, it serves as a counter to the Lovelace objection; a machine passes the Lovelace Test if it can demonstrate creativity (that is, if it can do something it wasn’t taught to do/it didn’t learn to do based on what it *was* taught to do). Furthermore, I don’t feel that the Stanford Encyclopedia’s refutation of this alternative test is completely sound.

Perhaps a blend of the two tests will suffice. Perhaps, instead, a machine is intelligent if it can pass both. I can’t say for sure, but I can’t personally rule out the Lovelace Test as a potential alternative. Then again, I’m not an expert/AI researcher/philosopher.

1. (5 points) Intelligence and computational limits: There are well-known classes of problems that are intractably difficult for computers and other classes that are provably undecideable by any computer. Does this mean that strong (human-level) AI is impossible? Why or why not?

No, I don’t think this necessarily implies strong AI is impossible. Consider, for example, the Traveling Salesman Problem (TSP). On even relatively small TSP instances, humans struggle to find the correct solution, but computers can already approximate (and, in some cases, correctly reach) a solution. In other problems, of course, humans perform better than computers. In still other problems, neither humans nor computers can reach a solution in any reasonable amount of time.

None of this indicates that we can’t develop strong AI. To me, it only indicates that some problems are inherently difficult, and that’s okay. It sometimes seems that people believe a computer must *outperform* humans to be considered intelligent, but that begs the question: are humans unintelligent? Even if neither a human nor a computer can solve a particular problem, the computer could still be as intelligent as the human.

1. (10 points) Consider a modified version of the vacuum-cleaner world (depicted in Figures 2.2 and 2.3 and specified on pages 35-36), in which the geography of the environment – its extent, boundaries, dirt locations, and obstacles is unknown; the agent can go Up and Down as well as Left and Right.
   1. Can a simple reflex agent be perfectly rational for this environment? Explain.

Yes, it can. If the agent’s performance measure awards one point for each clean square at each time step, and if we assume the following:

* The agent correctly detects whether or not its current square is clean, and
* Clean squares stay clean,

Then an agent that cleans a square if the square is dirty and otherwise moves to a random, adjacent square can be rational. Yes, it will, in all likelihood, require an extremely long time to clean the environment. No, this does not make the agent irrational under our current assumptions and performance measure.

* 1. Can you design an environment in which your randomized agent will perform very poorly? If not, explain; if so, provide an example.

Yes; designing such an environment is trivial. Consider an environment of size in which the bottom-left square is dirty, and all other squares are clean. Place the agent in the top-right corner.

Under our performance measure and the above assumptions, the agent will eventually clean the sole dirty square. Thus, under our performance measure, the agent performs well.

In general (that is, compared to other, better agents), though, the agent performs very poorly. The agent will randomly move around the large, open room for a long, *long* time before finally reaching – and cleaning – the bottom-left corner. The extreme time period indicates the agent’s poor performance.

* 1. Can a reflex agent with state outperform a simple reflex agent? Why?

Yes, clearly. If the agent is equipped with enough internal state to remember whether or not it’s been in a given square (note that the internal state might be very large, but we can make it so large as to encompass all realistic environments by representing each square as a unique number), it can avoid unnecessarily backtracking. We can implement this with a simple conditional: if the agent has already been in a given square, don’t go to that square. The agent can thus clean the environment in fewer time steps and thus start earning points sooner.

The stateful agent can also perform worse than a simple reflex agent. If, for example, the agent happens to spiral into a single square, the agent will be unable to move and thus fail to clean the room. In general, however, a minimally-stateful agent will perform better than a simple reflex agent.

1. (20 points) For the following tree, show the lists of open and visited nodes for each cycle of the listed search algorithms. Expand the nodes in a *right* *to left* ordering. The start node is**S**and the goal node is**X**. The numbersnext to the edges indicate the associated cost. Note: follow the format from class.
   1. Breadth-first search

|  |  |
| --- | --- |
| Frontier | Explored |
| S | - |
| ABC | S |
| BCDE | SA |
| CDEIF | SAB |
| DEIFG | SABC |
| EIFGH | SABCD |
| IFGH | SABCDE |
| FGHJ | SABCDEI |
| GHJ | SABCDEIF |
| HJ | SABCDEIFG |
| J | SABCDEIFGH |
| X | SABCDEIFGHJ |
| - | SABCDEIFGHJX |

Path: SBIJX, path-cost: 28

* 1. Depth-first search

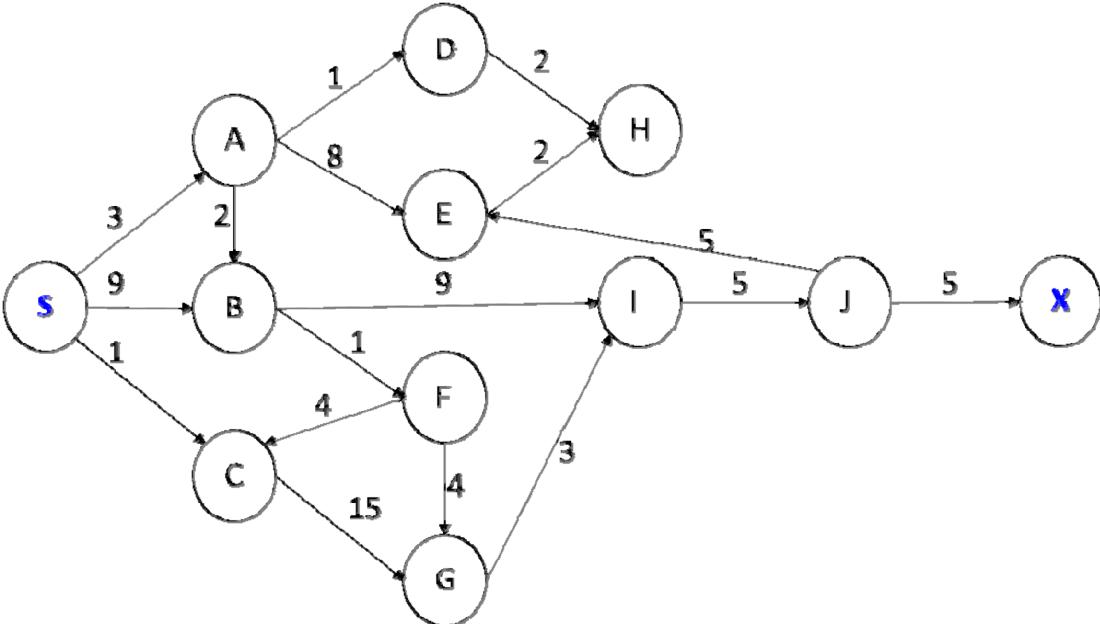
|  |  |
| --- | --- |
| Frontier | Explored |
| S | - |
| CBA | S |
| GBA | SC |
| IBA | SCG |
| JBA | SCGI |
| XBA | SCGIJ |
| BA | SCGIJX |

Path: SCGIJX, path-cost: 29

* 1. Uniform cost search

|  |  |
| --- | --- |
| Frontier | Explored |
| S0 | - |
| C1A3B9 | S |
| A3B9G16 | SC |
| D4B5­E11G16 | SCA |
| B5H6­E11G16 | SCAD |
| F6H6E11I14G­16 | SCADB |
| H6G10E11I14 | SCADBF |
| G10E11I14 | SCADBFH |
| E11I13 | SCADBFHG |
| I13 | SCADBFHGE |
| J18 | SCADBFHGEI |
| X23 | SCADBFHGEIJ |

Path: SBFGIJX, path-cost: 23



1. (40 points) Ah, Christmas is over. Now, that horrible mall traffic should let up… what is this, a traffic jam?

It seems that everyone is stuck, and you can’t get through. You must direct the trapped shoppers out of your way and maneuver your trusty vehicle through the maze to get back home in one piece.

For this problem, implement a solver for the Rush Hour puzzle. Rush Hour is a sliding block puzzle containing 4 trucks, 11 cars, and your vehicle placed on a 6x6 grid, with impenetrable walls surrounding its perimeter except for a one-cell exit edge. The trucks occupy an area of one by three cells, and the cars occupy an area of one by two adjacent grid cells. All vehicles can only move forwards or backwards, never overlapping. The goal is to move the vehicles one at a time so that your car may exit.

The implementation should read a set of puzzles from a text file (format shown in the example below) where A1 to K1 are cars, O1 to R1 are trucks, and X0 is your vehicle. The exit is always at the third position down on the right, and each empty grid location is (..). The file begins with the number of puzzles found in the file.

Expected turn-in is the 1) source code, 2) compilation and execution instructions, and 3) a short paper describing your solution, results and any search enhancements. A set of 5 problems are provided for testing your search algorithm during development (simple.txt), and 5 that must be tested against in the report (hard.txt); be sure to include your timing and best path results for each of the problems your search can solve. Your implementation must be your work only. As for what enhancements to try – take a look at the publications of Andreas Junghanns on Sokoban (<http://www.cs.ualberta.ca/~games/Sokoban/papers.html>).

To get you started, I have provided Java and MATLAB code that will read the files and handle the state updating (both are in the class directory). For the Java program, just implement the search interface in each of your own search(es). For MATLAB, fill in the search function. Note on implementation language – you may find that MATLAB is quicker to write than Java; however, you will take a significant performance hit. Using a BFS, the runtime comparison between the implementations on the simple.txt problems are:

Problem Java Matlab

1 0.003s 0.03s

2 0.08s 0.13s

3 0.50s 4.30s

4 0.44s 755.89s (13 min)

5 0.92s (7hr 36min)

1

O1..Q1Q1Q1B1

O1....C1..B1

O1X0X0C1.... <= Exit is here; this comment and arrow

......P1.... do not appear in the file.

......P1....

......P1....