Johns Hopkins University

665.645.8VL: Artificial Intelligence for Robotics

Homework and PA #2 [300 points]

PART A: Theory and Algorithms *[100 points]* ** See PART B Programming Assignment on Page 3*

Please - clearly write your **full name** on the first page. Submit a single PDF file as your answer for Part A. Please provide brief but complete explanations, using diagrams where necessary, and suitably using your own words. While presenting calculations and equations, explain the variables and answers in words.

Study Chapter 7 of Russel AI textbook 3rd Edition – selected sections only, plus propositional Logic notes provided. Answer the below:

1. Problem 7.7 [10 points]

2. Design a knowledge-based expert agent for a Street crossing guard robot. Describe [20 points]

- What kinds of objects in the world the robot will need to be aware of
- How these objects can be organized into an ontology (a short description or sketch of a taxonomic tree will suffice)
- What kinds of events in the world the robot needs to model to achieve its goals

Street crossing guard

Goal: Get all pedestrians across the street safely, without them getting hit by vehicles (but without stopping traffic entirely).

Sensors: The robot is aware of objects at either end of the crosswalk and objects that are about to enter the intersection from any direction.

3. Translate the following English sentences into propositional logic [10 points]

i. A and B are both true.

ii. If A is true, then B must be true as well.

iii. If a student studies for a test, they will do well on it. We can also tell that if a student did well on a test, then they must have studied for it.

iv. If a student is completely dry and it is raining outside, it is because they have an umbrella or a hoodie and it is not raining heavily.

v. If a student doesn't hand in the homework late or incomplete, this doesn't necessarily imply that they will not lose points.

4. Simplify and translate this *propositional logic* sentence into English [5 points]

$A \vee (A \wedge B) \Longleftrightarrow \neg(A \wedge B \wedge C)$

5. Translate the following English sentences into *first order logic* [10 points]

vi. Some students pass English but not Math.

vii. Every student is registered in a class and enrolled at a university.

viii. If someone is an aunt or uncle, then someone must be their niece or nephew.
ix. The old that is strong does not wither.

Study Chapter 8 and 10 of Russel AI textbook – selected sections only, plus propositional Logic notes provided. Answer the below:

6. From textbook p. 396-398

A. Problem 10.2 [4 points]
B. Problem 10.9 [6 points]

7. From textbook p. 315-320 [10 points for C to F]

C. Problem 8.1 a, b and c only.
D. Problem 8.6

8. Study this paper below on "CSP-Based Integrated Task & Motion Planning for Assembly Robots" Formulate this type of Robot's task and motion planning as a CSP in your own words. Explain how you would you a CSP solver such as the below to solve this CSP problem. You don't have to write code but pl. explain how you would develop the same (design approach). [25 points]

https://github.com/avshiliaev/constraint-satisfaction-solver
https://github.com/MaxWong03/aima-python/blob/master/csp.ipynb

References:
http://web.cecs.pdx.edu/~bart/cs541-fall2001/homework/1-prop.html
https://www.freecodecamp.org/news/building-an-ai-algorithm-for-the-tic-tac-toe-challenge-29d4d5adee07/

Reference Credits:
Homework problems 2 to 5 are from online resources on AI:
http://www.cs.jhu.edu/~phi
http://web.cse.ohio-state.edu/~newman-griffis.1/cse3521au17/

PART B: Implementation – Programming Assignment (PA #2) *[200 points]*

1. <u>Forward Planning</u>

Planning is an important topic in AI because intelligent agents are expected to automatically plan their own actions in uncertain domains. Planning and scheduling systems are commonly used in automation and logistics operations, robotics and self-driving cars, and for aerospace applications like the Hubble telescope and NASA Mars rovers.

This project is split between implementation and analysis. First you will combine symbolic logic and classical search to implement an agent that performs progression search to solve planning problems. Then you will experiment with different search algorithms and heuristics and use the results to answer questions about designing planning systems.

Read all of the instructions below carefully before starting the project so that you understand the requirements for successfully completing the project. Understanding the project requirements will help you avoid repeating parts of the experiment, some of which can have long runtimes.

NOTE: You should read "Artificial Intelligence: A Modern Approach" 3rd edition chapter 10 *or* 2nd edition Chapter 11 on Planning, available on the AIMA book site before starting this project.

Getting Started (Local Environment)

If you prefer to complete the exercise in your own local environment, then follow the steps below:

NOTE: You are *strongly* encouraged to install pypy 3.5 (download here) for this project. Pypy is an alternative to the standard cPython runtime that tries to optimize and selectively compile your code for improved speed, and it can run 2-10x faster for this project. There are binaries available for Linux, Windows, and OS X. Simply download and run the appropriate pypy binary installer (make sure you get version 3.5) or use the package manager for your OS. When properly installed, any python commands can be run with pypy instead. (You may need to specify pypy3 on some OSes.)

- Activate the aind environment (OS X or Unix/Linux users use the command shown; Windows users only run activate aind)

```
$ source activate aind
```

Completing the Project

1. Make sure that everything is working by running the example problem (based on the cake problem from Fig 10.7 in Chapter 10.3 of AIMA ed3). The script will print information about the problem domain and solve it with several different search algorithms.
   ```
   $ python example_have_cake.py
   ```
2. Complete all TODO sections in my_planning_graph.py. You should refer to the pseudocode\heuristics.md file provided, chapter 10 of AIMA 3rd edition or chapter 11 of AIMA 2nd edition (available on the AIMA book site) and the detailed instructions inline with each TODO statement. Test your code for this module by running:
   ```
   $ python -m unittest -v
   ```
3. Experiment with different search algorithms using the run_search.py script. (See example usage below.) The goal of your experiment is to understand the tradeoffs in speed, optimality, and complexity of progression search as problem size increases.

- Run the search experiment manually (you will be prompted to select problems & search algorithms)

  ```
  $ python run_search.py -m
  ```
- You can also run specific problems & search algorithms - e.g., to run breadth first search and UCS on problems 1 and 2:
  ```
  $ python run_search.py -p 1 2 -s 1 2
  ```

The `run_search.py` script allows you to choose any combination of eleven search algorithms (three uninformed and eight with heuristics) on four air cargo problems. The cargo problem instances have different numbers of airplanes, cargo items, and airports that increase the complexity of the domains.

Use your results to answer the following questions. Make a pdf for the answers to these questions.

- Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?
- Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)
- Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

2. Constraint Satisfaction Problem

A constraint satisfaction problem is a problem composed of *variables* that have possible values (*domains*) and *constraints* on what those values can be. A solver finds a potential solution to that problem by selecting values from the domains of each variable that fit the constraints. For more information you should check out Chapter 6 of Artificial Intelligence: A Modern Approach (Third Edition) by Norvig and Russell.

We are going to look at the Map Coloring problem here. The Map coloring problem is where you are provided with a map and no two adjacent states have the same color. Refer Section 6.1.1 in the book. You are provided with the starter code and need to fill in the solveCSP method according to the instructions provided in the code itself. You only need to submit the map_color.py file for this part.

Submission Checklist:
1. my_planning_graph.py
2. Answers pdf
3. map_color.py