CPSC 383: Explorations in Artificial Intelligence and Machine Learning

Assignment 3: Multi-Agent Systems, Planning, Re-Planning, Cooperation

Weight: 15%

Collaboration

Discussing the assignment requirements with others is a reasonable thing to do, and an excellent way to learn. However, the work you hand-in must ultimately be your **OWN GROUPS work**. This is essential for **YOUR GROUP** to benefit from the learning experience, and for the instructors and TAs to grade you fairly. Handing in work that is not **YOUR GROUPS** original work, but is represented as such, is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar. ***For purposes of this assignment the below should be read as applying to your group instead of you singular.***

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code that you hand-in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example:

   ```
   # the following code is from
   https://www.quackit.com/python/tutorial/python_hello_world.cfm.
   ```

   Use the complete URL so that the marker can check the source.

2. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. **However, you may still get a low grade if you submit code that is not primarily developed by yourself. Cited material should never be used to complete core assignment specifications. You can and should verify and code you are concerned with your instructor/TA before submission.**
3. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's screen, then this code is not yours.
4. **Collaborative coding is strictly prohibited**. **Your assignment submission must be strictly your code**. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing code itself, or modelling code after another student's algorithm. **You can not use (even with citation) another student's code.**
5. Making your code available, even passively, for others to copy, or potentially copy, is also plagiarism.
6. We will be looking for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).
7. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize. **The most common penalty is an F on a plagiarized assignment.**

8. For assignments **limited use of generative AI in writing assistance is acceptable. For example, grammar suggestion, or code suggestion tools for programming**. **Programming or text that is largely generative AI produced is not allowed**. Learners are ultimately accountable for the work they submit. Use of AI tools must be documented in an appendix for the assignment. The documentation should include what tool(s) were used, how they were used, and how the results from the AI were incorporated into the submitted work. Failure to cite the use of AI generated content in an assignment will be considered a breach of academic integrity and subject to Academic Misconduct procedures.

Late Penalty

For late individual assignments, those submitted within 24 hours of the initial deadline will receive 10% off, and within 48 hours will receive 20% off. After 48 hours, no late assignments will be accepted.

Goal

Within the provided **AEGIS** system modify the provided **Python agent** code to allow the agent to work together with other agents to save **SURVIVOR(s)**. Each simulation will begin with 7 identical agents all started from your one codebase. These agents will have unique ID numbers given to them. We will be forming groups of 3-4 students for this assignment. You are free to use code from assignments 1/2 for any of your groups members submissions. Assignment 3 does not need either advanced maze path-finding or image prediction to be completed successfully. The solution must be written by you the student group and not use existing libraries beyond those in aegis, those used for tensorflow in assignment2, or the base python installation.

Technology

AEGIS, Python 3.12

Submission Instructions

You must submit your assignment electronically using **D2L**. **We only need one submission per group as long as you communicated clearly which other students are in your group.** Use the Assignment 3 dropbox in **D2L** for a final codebase electronic submission. In **D2L**, you can submit multiple times over the top of a previous submission. Do not wait until the last minute to attempt to submit. You are responsible if you attempt this, and time runs out. Your assignment must be completed in **Python 3**.

Description

What is AEGIS? *(This is repeated from assignment 1)*

The Goobs have sent an elite space force to occupy AEGIS, the galaxy's central hub dedicated to saving lives across the galaxy. This futuristic space station floats in a serene nebula, and it's the last beacon of hope in the vast and dangerous expanse of space. Equipped with powerful scanners and teleportation gates, AEGIS connects distant worlds and provides a lifeline in the galaxy's most perilous regions. From here, they embark on their journeys, navigating the galaxy's dangers to rescue those in distress and tackle the most formidable dangers.

AEGIS is a simulated agent disaster rescue scenario environment written in Python 3 with an optional Electron-based GUI. AEGIS consists of a simulation controller that manages all communication with agents, executes the simulation, and maintains the current state of the world. It updates the world as events happen. If the optional GUI is used, the GUI displays real-time updates of the world's state during the simulation. Additionally, the GUI allows users to create their own worlds.

Full documentation is available here and students are expected to read this themselves to complete the assignment: https://cpsc-383.github.io/aegis/

Tutorials will also introduce students to the AEGIS system.

The AEGIS simulation is turn-based. When agents first **CONNECT** they are given a simplified rectangular grid view of the world which will include if a grid is safe to move on, or if the grid is **FIRE**/**KILLER** which will immediately kill an agent moving on top of it.

A simulation consists of multiple rounds, with each round being a single time step in the simulated world. During each round each agent is given one second to decide what action to take. AEGIS processes each agent one by one and waits for their command. **The last action command received from an agent is the one that will be executed for that round.** Agents have energy that is expended each time they use commands. If their energy expiries the agent becomes non-functional.

What is new in AEGIS for Assignment 3?

The world is no longer limited to **ONE SURVIVOR.** There now may be more than one survivor. These survivors may also be buried under **RUBBLE.** This **RUBBLE** requires agents to use a **DIG** command to remove it (sometimes it will require two agents to use **DIG** at the same time to remove it, but never more).

There are now **CHARGING** grids where an agent can stop and if they **SLEEP** on their turn they will **recover five energy**.

The agents also have two new commands.

1. They can **SENDMESSAGE** for free on their turn (and still do one action command like **DIG, MOVE, OBSERVE, SAVESURV, SLEEP**). This allows agents to send strings of information to each other, that will arrive on the next round of their group members.
2. Agents can call an **OBSERVE** command which allows agents to view a **LifeSignals** list for one grid location anywhere on the map. This can be useful to compare two grid locations far away to get a judgement how close to the top of the layer stack the

survivors at those locations are. The agent could then choose to go to the location in which survivors can be saved in less turns. (this is mostly only useful in the **For Fun** part of assignment as this semester we've chosen to expose all move costs from the start)

The configuration option **Move_Cost** for AEGIS will always be true, meaning the complete map of grid energy move costs will be communicated on connection success at start of the simulation.

MOVE_RESULT, TEAMDIG_RESULT, SAVESURV_RESULT all return surrounding info containing an update on move costs of the 9 local grids (already known) and an integer array of life signals which can be used to identify how deep a **SURVIVOR** is in the stack of **RUBBLE** at a location. This is the same life signal integer array that can be retrieved from **OBSERVE_RESULT** for one grid location anywhere on the map.

You will not know how many agents are required to remove a piece of rubble until it is the top layer on a grid location.

For assignment 3 you should familiarize yourself with the full list of commands for agents in the documentation.

A simulation will end when all survivors (could be more than one) have been saved using the **SAVE_SURV** command, all your agents expire from lack of energy of **FIRE/KILLER** grids, or when a time limit passes.

Assignment Challenge

Your agents in AEGIS will begin on a grid location that does not contain the **SURVIVOR**. They could all appear at the same location, or at different locations. There will be 7 agents. There will be one or more survivors to be saved.

It will be your job to move your agents each round until it is at the survivors' locations and then use **SAVE_SURV** to end the simulation. However, there will be an array of challenges that will require you to consider your system as a multi-agent systems of cooperative agents who will need to plan and re-plan to solve problems. There are 5 categories of these challenges that will be tested (3 tests each):

1. Decide you are by yourself to work by yourself to solve goal
    a. Remember there could be more than one agent, but there may be no way for them to reach each other
2. Decide you need to work together with a pair to solve a goal
    a. Some rubble needs more than one agent there to DIG at the same time to remove it
3. Plan to use a charging grid to survive a movement path to a goal.
    a. You will need to assess the costs to reach the survivor and if it is too costly you will need to find and stop at a charging grid to recover

4. Decide how to split to solve a goal with multiple parts
    a. There may be multiple locations with survivors at them and with uncertainty in time, your group should split to save all 3 at the same time instead of one at a time
5. Decide how to replan after solving a first goal (how to chain together goals)
    a. You maybe be able to save more than one survivor if you can re-plan to move on to other survivors after saving the first one

You will not need assignment 1 path-finding to do these. The maps will not be mazes, but you may find reasonable path-finding helpful, but again, not necessary.

Evaluation

At least one variant map for each of the five categories will be shared with the class. These are will be a starting point for testing your system. Your group is expected to make their own testing maps as well.

For grading we will have 3 maps for each of the five categories. Full credit will be given for saving the survivors in the maps. Some maps will have a limited time frame for full credit. For example, scenario 4 where the group should split to save multiple survivors at the same time.

Bonus

SAVESURV_RESULT returns an image (28 x 28 b/w) like assignment 2. You can return a prediction of label for this image to Aegis. Correct labels will be tracked by Aegis. A discussion of the awarding of one bonus point will be made with TAs. Groups with well-performing accuracy, up to the 10% of the class will receive 1 bonus point grade.

We are hoping to use the dataset collected from students this semester as the training/test dataset. You'll be provided the training data, and we'll keep the test data for aegis to use in the assessment.

For Fun

Completely secondary from your assignment grading, as long as time permits, we will run your system on randomly generated maps with scattered, rubble, killer, fire, charging, and survivor grid spots. We spawn 7 agents from your code and within a time limit keep track of how many survivors you save for points.

We will examine the points achieved by different student groups and the top groups will be invited on the final day of classes to show a couple of slides and tell the class what they incorporated into their design to achieve that performance.

Reminder that this has no consequence on your assignment 3 grading. This is completely for fun and is an opportunity for students to show off their unique ideas to the class.

Additional Specification

- You must comment your code and provide citations for the source of algorithmic designs and note any co-pilot like code suggestion usage.
- Put your **name, date, course, semester, and tutorial** into the comments of the example_agent.py (and other) files modified or created by you to complete your agent.
- **Do not rename or modify the provided common files.** You can create new .py files used by example_agent.py.
- **You should only use your A1 code for A* pathfinding and only use A2 code for the label predictions.**
- Do not change provided code without discussion with instructor. If there is a bug, or something is broken, the instructor should be informed to fix this issue.

Grading

The total grade is out of 20.

Five categories (out of 15)
    3 maps for each
Style/Commenting (out of 5)
    Name/Date/Course/Semester/Tutorial, don't change files, etc.
Bonus (1 bonus)
    Top teams at classifying the SAVE_SURV result images (up to 10% of class)

Submit the following using the Assignment 3 Dropbox in D2L

1. **example_agent.py**
    a. **Also any other .py files created and used by example_agent.py (however it is easier for us if your submission is just one file)**