Course Work for Artificial Intelligence H COMPSCI 4004 (2025-2026)

Debasis Ganguly, Edmond Ho

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in the course documentation, and work which is submitted later than the deadline will be subject to a penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

1. In respect of work submitted not more than five working days after the deadline
   (a) the work will be assessed in the usual way;
   (b) the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.
2. Work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for the late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus. Penalty for non-adherence to Submission Instructions is 2 bands You must complete an "Own Work" form via `https://studentltc.dcs.gla.ac.uk/` for all coursework

Submission and deadline

Your submission will be:

- A report in PDF which contains the answers to all questions including discussions, algorithms, plots and figures.
- The source code of the implementation of the corrupted samples selection method (Either put a **link to Google Colab or a link to a Github repository** containing a Python notebook in your report, **OR** a **zip file of your Python notebook** as a part of the Moodle submission).

Deadline: You must submit this on Moodle by 6.00 PM, Wednesday 26^{th} November, 2025.

Assignment: Escape the Frozen Maze of Wormholes!

Problem Statement

You find yourself trapped in a mysterious **Frozen Maze**. The maze is icy, slippery, and full of surprises. Some paths are blocked by unbreakable **walls**, others hide deadly **holes**.To make things even stranger, magical **wormholes** (one-way portals) are scattered across the land. Step onto a wormhole's entrance, and you will be instantly transported to its paired exit – but beware, these portals are not reversible, and they might help you or take you further from your goal.

Your task is to find a way out of this maze trying to avoid falling into the holes using a combinatorial search algorithm like the **A* algorithm**. You're also free to use any algorithm of your choice (there are no marks on the choice of any particular algorithm).

Environment Description

The environment is represented as an $N \times N$ grid. Each cell can contain:

- **A** : The **initial position** of the agent (always located at a random edge of the maze).
- **G** : The **goal**, a golden exit hidden in the far corner of the maze. Reaching this ends the episode successfully.
- **W** : An unbreakable **wall**. The agent cannot move into this cell.
- **H** : A **hole** on the lake. The agent tries to avoid falling into a hole. However, **it remains alive even after falling into a hole and continues to take actions**.
- **.** : A safe, empty cell of frozen ice.
- **S%d** : The **start** of a wormhole (e.g., S0, S1, ...).
- **E%d** : The **end** of the corresponding wormhole. Stepping onto S0 teleports the agent to E0, stepping onto S1 teleports to E1, etc. Note: wormholes are one-way – there is no return trip!

Random Maze Generation (Code given)

You are given a Colab notebook[1] that contains code to generate a maze on a frozen lake with teleportation links. For your experiments, the maze is configured by the following parameters.

The maze is generated randomly with the following parameters:

- `size` (N): the dimension N of the grid ($N \times N$).
- `wall_prob` ($w \in [0, 1]$): the probability that any given cell is a wall.
- `hole_prob` ($h \in [0, 1]$): the probability that any given cell is a hole.
- `n_portals` (m an integer): the number of wormhole pairs to be inserted.

For simplicity, the start state **I** is the top-left and the goal is the bottom-right cell. Portals are placed at random free cells, with each portal start Si paired uniquely with an end Ei.

[1] `https://colab.research.google.com/drive/1lPmHinN3T3FAqI-TSdKSWycO_aYYVIWY?usp=sharing`

Example Maze Generated (12×12)

```
W  W  .  W  .  W  .  H  .  W  H  H
H  W  .  W  .  W  H  .  .  W  W  .
W  .  .  H  .  .  H  .  H  .  .  .
H  .  .  .  .  .  .  .  .  .  .  W
.  W  E0 .  .  W  H  .  W  .  .  H
W  W  .  W  .  H  .  .  .  .  .  .
.  H  E1 W  .  .  .  W  W  W  H  S0
H  .  .  W  W  .  W  .  W  .  W  .
.  .  H  .  .  H  H  .  .  .  .  .
.  .  H  W  .  W  .  .  W  .  .  .
.  W  S1 .  .  .  .  W  W  .  H  .
A  .  .  H  .  W  .  .  .  W  .  G
```

Actions in the Environment

Some notes on the wormholes:

- An agent **knows if a cell contains a wormhole starting or ending point**.
- An agent **doesn't know where the portal leads to** until it actually takes an action to visit the cell containing the start of a wormhole.
- **Wormholes are not reversible**, i.e., an agent can't move from Ei to Si.

The agent can move in the four cardinal directions:

$$\{0 = \text{LEFT}, \quad 1 = \text{DOWN}, \quad 2 = \text{RIGHT}, \quad 3 = \text{UP}\}.$$

The agent cannot move outside the boundaries of the grid or into a wall cell.

To make your search through the stochastic environment, you need to define a parameter $p \in [0, 1]$, which gives you the probability of ending up in the desired direction; the other direction that you can end up with is the direction at a 90 degrees angle to the intended direction. The state transitions are explicitly stated as follows:

$$\begin{aligned}
P(s_{n+1} = (i+1, j) \mid a = \text{DOWN},\ s_n = (i, j)) &= p, \\
P(s_{n+1} = (i, j+1) \mid a = \text{DOWN},\ s_n = (i, j)) &= 1 - p, \\
P(s_{n+1} = (i, j+1) \mid a = \text{RIGHT},\ s_n = (i, j)) &= p, \\
P(s_{n+1} = (i-1, j) \mid a = \text{RIGHT},\ s_n = (i, j)) &= 1 - p, \\
P(s_{n+1} = (i-1, j) \mid a = \text{UP},\ s_n = (i, j)) &= p, \\
P(s_{n+1} = (i, j-1) \mid a = \text{UP},\ s_n = (i, j)) &= 1 - p, \\
P(s_{n+1} = (i, j-1) \mid a = \text{LEFT},\ s_n = (i, j)) &= p, \\
P(s_{n+1} = (i+1, j) \mid a = \text{LEFT},\ s_n = (i, j)) &= 1 - p.
\end{aligned}$$

The first set of equations means that the agent goes DOWN with probability p else it goes RIGHT with probability $1 - p$. Similarly, the second set means that when you intend to go RIGHT you may end up going UP with probability $1 - p$, and so on.

Tasks

Implement your solution (15 marks)

You need to **implement** and **explain** an algorithm of your choice. You can choose to approach this problem with any discrete state-space search algorithm. Your report should clearly explain how the state-space search algorithm, that you decided to implement, works. You should ideally illustrate this on a small example maze.
For this exercise, you are free to use any software library, e.g., you are free to use AIMA book's Python (`https://github.com/aimacode/aima-python` or Java `https://github.com/aimacode/aima-java`) libraries. You may also write your own code to implement a combinatorial search algorithm.

Task 2: What happens if there're more portals? (15 marks)

Now that the implementation is done, it's now time to run some experiments and report observations. Let's define the performance measure of the environment as:

$$P = Minimize \quad \#\text{steps (time) to get out} \times \#\text{times agent falls into a hole} \tag{1}$$

For this set of experiments, you are going to use the following values for the static parameters.

- N (maze grid size) is 50 (or set to a higher grid size to increase the likelihood of solvability).
- w (`wall_prob`) is 0.15.
- h (`hole_prob`) is 0.15
- $p = 0.8$

Now vary the number of wormholes `n_portals` (m) from 5 to 30 in steps of 5.

- Report a table of the performance measure (Equation 1) for these 6 different values of m.
- Based on your observations, comment on the effects of the number of wormholes on the performance of the algorithm.
- Try to explain the reason behind your observations.

Task 3: What happens if there's more melted ice? (15 marks).

For this set of experiments, you are going to use the following values for the static parameters.

- N (maze grid size) is 50 (or set to a higher grid size to increase the likelihood of solvability).
- w (`wall_prob`) is 0.15.
- m (`n_portals`) is 15.
- $p = 0.8$

Now vary the parameter `hole_prob` (h) from 0.05 to 0.3 in steps of 5, i.e., $h \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$.

- Report a table of the performance measure (Equation 1) for these 6 different values of h.
- Based on your observations, comment on the effects of the number of holes on the performance of the algorithm.
- Try to explain the reason behind your observations.

Task 4: What happens if the lake becomes more slippery? (15 marks).

For this set of experiments, you are going to use the following values for the static parameters.

- N (maze grid size) is 50 (or set to a higher grid size to increase the likelihood of solvability).
- w (`wall_prob`) is 0.15.
- m (`n_portals`) is 15.

Now vary the parameter p from 0.9 to 0.2 in steps of 0.1, i.e., $p \in \{0.9, 0.8, \ldots, 0.2\}$.

- Report a table of the performance measure (Equation 1) for these different values of p.
- Based on your observations, comment on the effects of the uncertainty of the actions (slippery nature of the surface) on the performance of the algorithm.
- Try to explain the reason behind your observations.

Marking Criteria

Marks will be given to: 1) **clarity of the discussion**, and 2) **reasoning behind your observations**. You may consider supporting your discussions by referencing to other articles/books as well. Note that the performance measure of your solutions themselves will NOT be affecting the marks for this task.