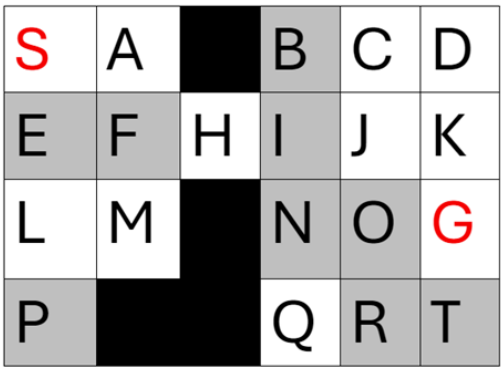
F29AI - Artificial Intelligence and Intelligent Agents

Coursework 1 - Undergraduate

A* Search and Automated Planning

You should complete this coursework **working in pairs**. Coursework 1 has **two parts** (A* search and automated planning using PDDL) and is worth **25%** of your overall F29AI mark. Details of what you should do and hand in, and how you will be assessed, are described below and on Canvas.

Part 1: A* Search



(a) Grid 1

(b) Grid 2

The above grids represent two problem environments where an agent is trying to find a path from the start location (S) to the goal location (G). The agent can move to an adjacent square provided the square is white or grey. The agent cannot move to a black square which represents a wall. No diagonal movement is allowed. Moving into a white square has a cost of 1. Moving into a grey square has a cost of 2. You should assume that ties on the fringe are broken by alphabetical ordering of the nodes.

Part 1A (5 marks)

Provide solutions to the grid problem by applying A*search by hand. If a grid has more than 1 solution, ensure you provide all solutions. You should do the following:

1. Draw a graph (with nodes and edges) to represent each of the grid problems as a search problem. Label each of the nodes in your graph and include appropriate costs on the edges.
2. Use the Manhattan distance heuristic and calculate appropriate heuristic values for each of the nodes in your graph. Ensure you show your heuristic values.
3. Apply A* search to each grid and list the final set of states expanded, the goal path for each grid and its corresponding total cost. You do not need to provide a full derivation for each search (unless you wish to do so), however, you should provide at least the first 5 steps of each derivation with calculations for the f values to demonstrate you understand the application of the A* algorithm in your graph.

1st grid		
States expanded:	Goal path:	Total Cost:
2nd grid		
States expanded:	Goal path:	Total Cost:

Your solution will primarily be marked on its correctness with respect to the two grid problems. We will also check your graph and heuristic values you are using in your solution.

Part 1B (15 marks)

Implement a solution to the grid problem using A* search in Java. A starter code is provided for you to help you start the project. Your Java code will primarily be marked on its correctness with respect to the two grid problems. We will also check the implementation of the heuristic in your solution.

Programming language: Java

The starter code can be found in the package **uk.ac.hw.macs.search**, which is a set of classes that can be used to represent a search problem. To implement a specific search problem, you will need to do the following:

1. Define a class that implements the State interface. This should include the following:
   (a) A method for determining whether a state is a goal state `(isGoal())`
   (b) A method for computing the heuristic value of a state `(getHeuristic())`
2. Define a class that implements the `SearchOrder` interface. This interface has one public method, `addToFringe`, that is used to add a set of nodes to the frontier. You can use the costs and/or heuristic values to determine the order that nodes are added to the frontier

The classes in the **uk.ac.hw.macs.search.example** package show examples of these two interfaces being used to implement depth-first search and breadth-first search, as well as a simple integer-based state. The `Main` class in this package shows an example of how to use the classes.

To solve the problem, you will need to implement the following:

1. An encoding of the state in the grid by implementing the `State` interface appropriately, including methods for detecting a goal state and computing a heuristic value. You should use the **Manhattan distance** heuristic for generating heuristic values in your search.

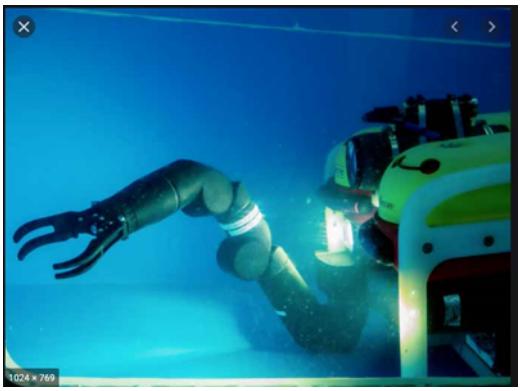
2. A class implementing A* search by implementing the `SearchOrder` interface and implementing `addToFringe` appropriately.

Test your code on the two grid problems provided by running A* search on them and capture the output. Submit the code used for the implementation of the problem. Make sure your code includes appropriate comments in the parts of the code you implemented. Do not change any of the classes in the **uk.ac.hw.macs.search** package: we will test your implementation against the provided versions of these classes.

Part 2: Automated Planning

Heriot-Watt and The National Robotarium have secured a project focused on the underwater robotic inspection of wind farms in the North Sea. Given the challenging environment of the North Sea, these missions must be fully automated. You have been tasked with creating a planning solution for the deployment of an Unmanned Underwater Vehicle (UUV). The UUV will be carried by the ship to the deployment site and then released to perform several tasks: navigating through a series of waypoints/locations, capturing images of underwater structures, conducting sonar scans of the seabed, and collecting samples at various locations. The UUV can transmit image and sonar data back to the ship's local storage. The collected samples, however, can only be delivered back to the ship once the UUV returns to the ship.

(a) An illustration of a ship.



(b) An illustration of a UUV.

A decision has been made to use PDDL and one of the off-the-shelf planners to model the mission problems and define the environment in which the UUV needs to operate. The PDDL

domain must represent the various types of objects that might exist in the North Sea missions, including the UUV, underwater locations, and the ship.

The domain must define predicates that describe the world state, such as the UUV's position, the connections between locations, and whether the UUV has captured data.

A single problem instance needs to define the mission's initial state (e.g., ship location, path between waypoints) and the final goal as described in the given tasks (e.g., the samples or data to be collected).

Part 2A: Modelling the Domain

The first step is to understand the **domain**. For this part of the coursework, you need to implement the domain of the planning problem.

Task 1.1: Describing the World State (3 marks)

Define the types of objects involved in the mission and the predicates representing the system's state. These predicates should indicate whether the UUV is at a specific location, has collected an image, the connection between two waypoints, and other relevant states.

Task 1.2: Defining Actions (3 marks)

Define the actions that can be performed. These actions may include deploying the UUV, moving between locations, taking pictures at specific underwater sites, using sonar to scan the environment, and transmitting data back to the ship. Each action has specific preconditions that must be met for execution. For example, the UUV must be at the correct location to take a picture, or it must have collected sonar data before it can transmit it to the ship. Each action also has certain effects that will modify the overall world state if the preconditions are met. For instance, if there is a path between two locations and the UUV is at one of them, it can move along the path, resulting in it no longer being at the starting point but at the new location.

RESTRICTIONS

- The UUV can be deployed at any location but only once. It cannot return to the ship to navigate through waypoints.
- The UUV has limited memory, capable of storing only one piece of data at a time.
- To collect a sample, the UUV must pick it up from a location and return to the ship to store it.
- Each ship can store only one sample.

Part 2B: Modelling the Problems

For this part of the coursework, you need to implement the **problems**, or the missions. The planner needs these to **generate a final plan** that needs to be included in the final report.

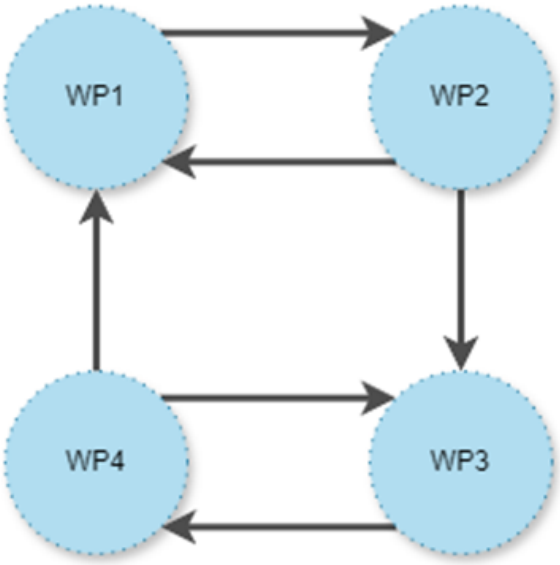


Figure 3: Paths for Problem 1.

Task 2.1: Problem 1 (3 marks)

- The UUV begins its journey from the ship.
- Use the map in Figure 3 to configure connections and paths between waypoints.

Mission Goals:

- Save an image at waypoint 3.
- Save a sonar scan at waypoint 4.

Task 2.2: Problem 2 (3 marks)

- The UUV begins its journey from the ship.
- Use the map in Figure 4 to configure connections and paths between waypoints.

Mission Goals:

- Save an image at waypoint 5.
- Save a sonar scan at waypoint 3.
- Collect a sample from waypoint 1.

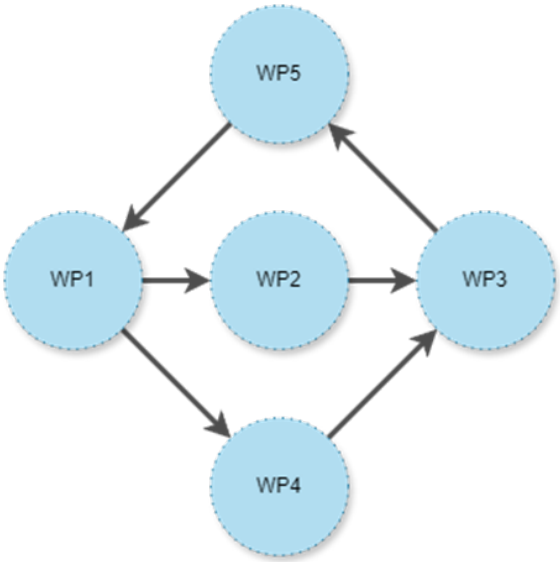


Figure 4: Paths for Problem 2.

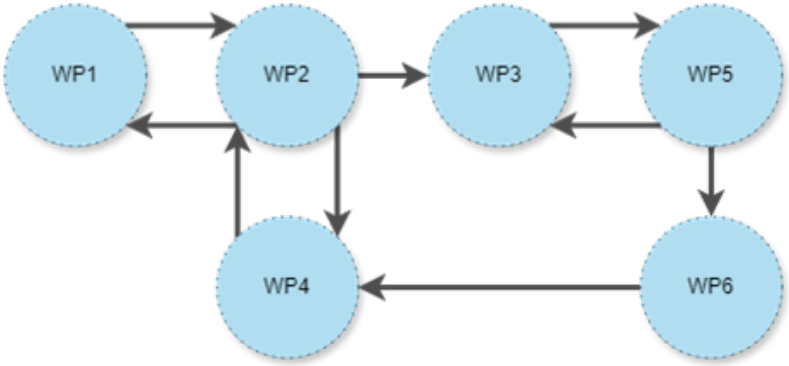


Figure 5: Paths for Problem 3.

Task 2.3: Problem 3 (3 marks)

- Use the map in Figure 5 to configure connections and paths between waypoints.
- Introduce a second UUV and a secondary ship to assist with the mission.
- The first UUV starts deployed at waypoint 2.
- The second UUV starts from the secondary ship.

Mission Goals:

- Save an image at waypoint 3.

- Save a sonar scan at waypoint 4.
- Save an image at waypoint 2.
- Save a sonar scan at waypoint 6.
- Collect a sample from waypoint 5.
- Collect a sample from waypoint 1.

Part 2C: Extension

In this part you will extend the domain by adding some additional features. **This task uses the same environment configuration and mission goals as Task 2.3 with 6 waypoints** (Figure 5).

Task 3.1: Problem 4 (5 marks)

- You have a brave engineer to assist in deploying the robot in the North Sea.
- Add locations on the ship: the bay and the control centre. The engineer can walk between these two locations.
- The UUV can only be deployed or return to the ship if the engineer is at the bay.
- The UUV can transmit images and scan data only if the engineer is in the control centre.

Instructions

1. **Download the Starter Files:** Download the **PDDL_UG.zip** file containing the starter PDDL code folders and files.
2. **Folder Structure:** The zip file contains two subfolders:
   - **PartA&B:** For Part 2A and Part 2B.
   - **PartC:** For Part 2C.
3. **Getting Started:** Begin by modifying the provided domain and problem files. The domain and problem names are already set up for you.

PDDL Implementation

Do not use any features of PDDL that we have not covered in the course. Include comments in your PDDL files to describe important sections of your code. **You do not need PDDL features that we have not covered in the course.** You should clearly indicate in your report what planner you have used to test your solution. PDDL files will only be tested on editor.planning.domains, FF, or Fast Downward. Correctness will be assessed not only against the coursework requirements but also with respect to the specific implemented solution. (I.e., non-obvious or incorrect/missing action preconditions or effects may lead to strange plan output and mark deductions.) Usual program quality criteria (e.g., use of whitespace, comments, naming

conventions, etc.) will apply here to assess the readability of the code. The marker will be looking to see if you understand how to write PDDL domains and problems and have made good use of the language features that are available.

We recommend using Visual Studio Code (`https://code.visualstudio.com/`) as code editor for PDDL, and the PDDL support extension: `https://marketplace.visualstudio.com/items?itemName=jan-dolejsi.pddl`.

Once the extension is installed you can run the planner by right clicking in the domain or problem file and click on "PDDL: Run the planner and display the plan"

You can select the online plan-as-service operation mode offered by planning.domains, with one of this Best First Search planners: "BFWS –FF-parser version", "LAMA-first"

IMPORTANT: if you have multiple domain files make sure the name of your domain matches the one in the problem you want to run the planner with as sometimes there could be errors.

Example:

Domain:

```
(define (domain windfarm)
...
)
```

Problem:

```
(define (problem windfarm-mission-1)
    (:domain windfarm)
...
)
```

Alternatively, if you are having problem setting up the development environment you can use the online editor: `https://editor.planning.domains/`. If you are using Visual Studio Code and you want to run planners offline you can always download one and use it on your local machine, here are some of the most important classical planners:

- Fast Downward (`https://www.fast-downward.org/HomePage`)
- FF (`https://fai.cs.uni-saarland.de/hoffmann/ff.html`) (look for FF-v2.3)
- Pyperplan (`https://github.com/aibasel/pyperplan`)

What to hand in

Submit a single zipped folder (**accepted formats: .zip or .rar**) that includes the following:

1: Report

A report detailing Part 1A and Part1B A* search showing a clear step-by-step approach to solve the problem. The report should contain screenshots of your artefact, code snippets of the added classes as well as the output and justification of your approach using appropriate references where necessary (maximum 2-3 pages).

The same report should contain Part 2 PDDL (maximum 2-3 pages) where you briefly discuss how you structured your domain. Show the different types of locations in your domain and the location of the command centre. Also list in your report the problem instances you have included in your code and what planner you have used to test your domain/problems. The report will be used by the markers to help understand your code.

At the beginning of your report, clearly state the group you have signed up for on Canvas (e.g., Edinburgh CW 1 UG 42), along with the names and student ID numbers of all group members. At the end of your report, clearly provide the link to your video demo.

2: Code (Java and PDDL files)

Submit the code for the implementation of the problem in Part 1B. Make sure your code includes appropriate comments in the parts of the code you implemented. Do not change any of the classes in the **uk.ac.hw.macs.search** package. Your Java code will primarily be marked on its correctness with respect to the two grid problems. We will also check the implementation of the heuristic in your solution.

Submit your PDDL source files consisting of those included in the starter package **PDDL_UG.zip**. Make sure your source files have comments describing the properties and actions you've defined. Your source files will be checked for plagiarism and tested to see if they are operational.

3: Video Demo (link)

A video walk-through of the implementation (both students need to be present in the recorded video) explaining the code implementation from a suitable Java IDE and design decision made for the A*search algorithm. In the same video, you should also include a demo of the PDDL domain, predicates, output etc by opening the planner, generating the plans from the given tasks. The video should not be more than 8 minutes long.

Please prepare the video recording and submit the **link** to us. One convenient way is to use Microsoft Stream or Teams to record your video (note: you may use other tools as you wish) and then provide the Sharepoint / OneDrive / Google Drive link in your report. It is your responsibility to ensure the link is accessible, otherwise no marks will be awarded to this part.

4: Declaration of Authorship

Students are also required to sign and include a standard declaration of authorship with each submission (note: one declaration per student). This is a mandatory part of the assessment submission. Links to the standard declaration can be found on the Canvas site for F29AI.

Deadlines

The deadline for submitting Coursework 1 (all parts) is Thursday, 24 October 2024. Submissions are due by 15:30 (Edinburgh time) for the Edinburgh Campus, 23:59 (Dubai time) for the Dubai Campus, and 23:59 (Malaysia time) for the Malaysia Campus. Details on how to submit your coursework will be posted on Canvas.

Feedback

Individual written feedback will be provided to students approximately three working weeks after the submission of Coursework 1.

Additional notes

This is a group coursework assignment. All submitted files will be checked for plagiarism. You must conform to the naming conventions described in this document. If files are unreadable or code does not run or video does not play, you will receive 0 marks on that part of the assessment. You are not permitted to use generative AI tools for any part of this coursework.

Assesment

This coursework will count towards 25% of your overall mark for F29AI and will be marked out of 45 marks.

A* Search: Non-coding solution (5 marks).

0 (None)	1 (Poor)	2 (Fair)	3 (Good)	4 (Very Good)	5 (Excellent)
No solution submitted.	Major problems with solution. Solution is incorrect and/or many parts of the solution are wrong or missing.	Major problems with solution. Some parts of the solution are partially incorrect or missing. Description of the solution could be improved in many places.	Good solution. Solution is mainly correct but there are some minor problems. Description of the solution could be improved.	Very good solution. All parts of the solution are described, and the solution is correct but there are some small problems with the description that could be improved.	Exceptional solution. All parts of the solution are well described, and the solution is perfect.

A* Search: Java code (15 marks)

0 (None)	1-3 (Poor)	4-6 (Fair)	7-9 (Good)	10-12 (Very Good)	13-15 (Excellent)
No code submitted. Poor demonstration.	Major problems with code. Solution is incorrect and/or code does not run. Demonstration shows that the artefact is poor, lacks functionality or does not work, and falls well below expectation. Demonstration shows that the artefact is poor, lacks functionality or does not work, and falls well below expectation.	Code partially works but there are major problems with code structure, solution, and/or heuristic. Demonstration shows that the artefact has limited functionality. Demonstration shows that the artefact with limited functionality.	Good code and solution. Code runs almost perfectly but there are problems with code structure, solution, or heuristic. Demonstration shows that the artefact is equal to expectation.	Very good code and solution. Code runs perfectly. Small problems with code structure, solution, or heuristic. Good demonstration that proves the artefact goes beyond expectation.	Exceptional code and solution. Code runs perfectly. Heuristic is well defined. Exceptional demonstration that proves the artefact goes far beyond expectation.

Automated Planning (20 marks)

The automated planning mark will primarily be based on the PDDL code you have supplied and how correctly it implements the tasks. The points breakdown is as follows:

- Task 1.1: 3 marks
- Task 1.2: 3 marks
- Task 2.1: 3 marks
- Task 2.2: 3 marks
- Task 2.3: 3 marks
- Task 3.1: 5 marks

Overall Presentation of work and Communication (5 marks)

0 (None)	1 (Poor)	2 (Fair)	3 (Good)	4 (Very Good)	5 (Excellent)
Elements of disorganisation/poor presentation/poor communication or expression or Communications too brief or rambling, inappropriate to context or purpose, with many errors/omissions, inadequately expressed/presented. Little or no clear introduction to the topic.	Organisation and presentation of work and communications adequate in most contexts, with some mistakes/irrelevancies. Outlines a basic introduction to the topic with limited resources from the literature.	Satisfactory organisation and presentation of work, communications mostly appropriate to the context/purpose. Identifies relevant key themes using appropriately cited sources.	Presentation and organisation of work appropriate to context and purpose, communication clear. Detailed overview of the topic supported by a range of appropriate sources.	Excellent presentation and organisation of work and fluent communication in most contexts. Good overall report supported by a wide range of literature from good resources.	Exceptional presentation and organisation of work and fluent communication in all contexts. Evidence of independence of thought and ability to build upon experience, supported by literature from high quality resources.

Learning Objectives

This coursework is meant to contribute to the following high-level aims for F29AI:

- To introduce the fundamental concepts and techniques of AI, including planning, search, and knowledge representation.
- To introduce the scope, subfields, and applications of AI, including autonomous agents.
- To develop skills in AI programming in an appropriate language.

It is also meant to contribute to the following specific learning objectives for the course:

- Critical understanding of traditional AI problem-solving and knowledge representation methods.
- Use of knowledge representation techniques (such as predicate logic).
- Critical understanding of different systematic and heuristic search techniques.
- Practice in expressing problems in terms of state-space search.
- Broad knowledge and understanding of the subfields and applications of AI.
- Detailed knowledge of one subfield of AI (e.g., planning) and ability to apply its formalisms and representations to small problems.
- Detailed understanding of different approaches to autonomous agent and robot architectures, and the ability to critically evaluate their advantages and disadvantages in different contexts.
- Practice in the implementation of simple AI systems using a suitable language.
- Identification, representation, and solution of problems.
- Research skills and report writing.
- Practice in the use of information and communication technology (ICT), numeracy, and presentation skills.

Late submission of coursework

Coursework deadlines are fixed and individual coursework extensions will not be granted. Penalties for the late submission of coursework follow the university's policy on late submissions:

- The mark for coursework submitted late, but within 5 working days of the coursework deadline, will be reduced by 30%.
- Coursework submitted more than 5 working days after the deadline will not be marked.
- In a case where a student submits coursework up to 5 working days late, and the student has valid mitigating circumstances, the Mitigating Circumstances policy will apply. Students should submit a Mitigating Circumstances application for consideration by the Mitigating Circumstances Committee.

The MACS School policy on coursework submission is that the deadline for coursework submissions, whether hard-copy or online, is 15:30 (Edinburgh time) for the Edinburgh Campus, 23:59 (Dubai time) for the Dubai Campus, and 17:00 (Malaysia time) for the Malaysia Campus. The Submission of Coursework policy can be found here: `https://www.hw.ac.uk/services/docs/learning-teaching/policies/submissionofcoursework-policy.pdf`

Mitigating Circumstances (MC)

There are circumstances which, through no fault of your own, may have affected your performance in an assessment (exams or other assessment), meaning that the assessment has not accurately measured your ability. These circumstances are described as mitigating circumstances. You can submit an application to have mitigating circumstances taken into account. Full details on the university's policies on mitigating circumstances and how to submit an application can be found here: `https://www.hw.ac.uk/students/studies/examinations/mitigating-circumstances.htm`

Plagiarism

"Plagiarism is the act of taking the ideas, writings or inventions of another person and using these as if they were your own, whether intentionally or not. Plagiarism occurs where there is no acknowledgement that the writings, or ideas, belong to or have come from another source." (Heriot-Watt University Plagiarism Policy). This coursework must be completed independently:

- Coursework reports must be written in a student's own words and any submitted code (e.g., PDDL) in the coursework must be your own code. Short sections of text or code taken from approved sources like the lecture examples may be included in the coursework provided these sources are properly referenced.
- Failure to reference work that has been obtained from other sources or to copy the words and/or code of another student is plagiarism and, if detected, this will be reported to the School's Discipline Committee. If a student is found guilty of plagiarism, the penalty could involve voiding the course.

- Students must never give hard or soft copies of their coursework reports or code to another student. Students must always refuse any request from another student for a copy of their report and/or code.
- Sharing a coursework report and/or code with another student is collusion, and if detected, this will be reported to the School's Discipline Committee. If found guilty of collusion, the penalty could involve voiding the course.

Plagiarism will be treated extremely seriously as an act of academic misconduct which will result in appropriate student discipline. All students should familiarise themselves with the university policies around plagiarism which can be found here: `https://www.hw.ac.uk/students/studies/examinations/plagiarism.htm`