Artificial Intelligence Fundamentals

Master in Artificial Intelligence

Lab 1: Implementation of search algorithms

1. Description of the problem

Current technological development has unleashed a fever for the mining of new metals and rare earths. Multiple companies and governments around the world are rushing to increase their market share, since it is well known that whoever dominates the mining industry will soon dominate the world, hahahaha!

In an unprecedented turn of events, the Government of Spain has decided to take the lead and proposes an ambitious plan that will involve a multidisciplinary team of engineers organized into three groups. The first group will be in charge of developing a state-of-the-art sonar technology system, which will allow obtaining a map of the subsoil to locate minerals with unprecedented precision. Another team will develop a drilling robot equipped with the latest drilling technology, batteries, and remote control. Finally, a group of AI experts will be in charge of developing the software to optimize the robot's guidance and drilling software for maximizing productivity in the mining task. In this last area, they decide to delegate the mission to the best students of the Master of Artificial Intelligence to develop an IA-based search-algorithm for accomplishing the task.

The beach bar economy is over! Let's help our country to be at the forefront of industrialization and the knowledge-based society!

All right, so the task is the following. To develop our first prototypes, we will start from a matrix coding of the terrain map obtained by the first team with the following format:

```
1 2 2 1 4 3 2 1
3 2 5 1 2 4 2 1
2 3 5 3 3 2 4 4
1 3 1 1 4 1 1 2
3 5 1 2 4 4 1 4
1 3 1 1 4 5 1 1
```

where digits 1 to 9 denote the hardness of the subsoil rock at each point. We are also provided with an additional pair of coordinates (x_0, y_0) and (x_t, y_t) that respectively identify the start and target locations of the drilling robot, the latter where the precious mineral has been localized.

We can interpret the matrix codification with the top left-most position representing the (0,0) coordinate, with the bottom most-right position corresponding to (5,7) in the example above. Of course, we should be able to find the best path across a wide range of terrain maps with different configurations and target locations.

In modelling our problem, we should also take into account the following restrictions:

- At each position we should consider the robot's orientation, which can be 'North', 'Northeast', 'East', 'Southeast', 'South', 'Southwest', 'West' or 'Northwest'

- In addition, at each position only one of the following three options can be performed:
    - Move (drill) in the direction of its current orientation
    - Rotate 45º clockwise to change his orientation
    - Rotate 45º counterclockwise to change his orientation

Each 45º turn action has an associated cost of one unit, and the cost of advancing toward the next position has the cost associated with the hardness of the rock at that position. The drilling robot can only move in the direction according to its current orientation.

2. General objectives

The main objectives of this work are the following:

1. To apply blind and informed AI search strategies in order to find the best path (sequence of actions) to get from (x_0, y_0) to (x_t, y_t) while complying with the above-stated set of constraints

2. To carry out a comparative analysis between the different applied methods to determine the one that maximizes drilling performance (path with minimum cost)

3. To develop heuristics and to evaluate their quality and contribution in the context of the targeted problem

3. Specific requirements

The task requires the implementation of an AI program that allows the application of different algorithms of blind search (depth- and breadth-first) and informed search (A*) for the resolution of the above stated problem. The resulting code will be considered part of the deliverable results.

In addition, a written report document (maximum 10 pages excluding title, index, and possible appendices) will be prepared that MUST be composed of the following sections:

3.1. Front page

It will indicate the title (Artificial Intelligence Fundamentals. Lab 1: *Implementation of search algorithms*. Master in Artificial Intelligence 2025-2026), the authors, a reference to the corresponding practice group, and the date.

3.2. Index of contents

3.3. Methods

3.3.1. Formal characterization of the problem that should include the following aspects:

- Representation of the state
- Specification of the available set of operators (actions of the robot)
- The transition model (possible set of preconditions and the result of each action)
- Goal test
- Cost function

3.3.2. Analysis that justifies the *a priori* suitability for the problem (that is, before its implementation) of one of the two generic blind search methods (depth- and breadth-first). For the analysis, the theoretical requirements of *memory* and *execution time* will be taken into account, as well as a reference to their expected *optimality* and *completeness* in the context of the proposed problem.

3.3.3. Define the concept of heuristic function and propose one heuristic to be applied in the context of the problem, to be used in the context of the A* algorithm. For the heuristic, describe:

1) Its expression in mathematical terms

2) The rationale for its suitability in the context of the stated problem, including an explanation of whether the heuristic satisfies *the monotony property*. Why is it advisable, besides *admissibility*, to require the monotony property in a heuristic?

3.4. Results

3.4.1. Execution trace example

The complete trace of the execution of each one of the implemented search algorithms will be shown for the map provided in the file “exampleMap.txt”. See Section 4 for details on the specific input and output formats. In particular, the execution trace <u>must strictly follow</u> the format expressed in Section 4.2.

3.4.2. Comparison of the performance of the different implemented methods

For this purpose, different simulations will be carried out generating 5 different random maps with respective sizes of N = 3x3, 5x5, 7x7 and 9x9, with fixed start and target positions (0,0) and (N-1, N -1). Each of the resulting problems will be subsequently solved using each of the implemented algorithms, generating a table like the following:

Table X: *Comparative table of performance of search methods in the map of dimension N*

	d	g	#E	#F
Breadth-first				
Depth-first				
A* (h)				

where:

- *d*: is the depth of the exploration tree-graph in which the solution has been found
- *g*: is the cost of the solution path obtained
- *#E*: is the total number of explored (expanded) nodes throughout the corresponding algorithm's execution
- #F: is the final number of nodes stored in the frontier (open list) at the end of the execution

As result, a total of 4 tables will be generated (one for each map size). On each table, row values will correspond to the averaged 5 results obtained for each random simulation

3.5. Discussion

3.5.1. Based on the results obtained in section 3.4, discuss the advantages and disadvantages of each of the different search methods in the context of our problem, and *why* this is the case

3.5.2. For the A* algorithm, and also in view of the obtained results, elaborate on the performance of implemented heuristic, and why

3.5.3. Let us now consider a slight variation over the original problem, which is the possibility that we find some rocks of extreme hardness that our drilling robot would not be able to perforate.

1) How could we represent this situation in the geological map? How would this affect the set of available actions and problem complexity?

2) Would all the heuristics still be valid? If not, how could we adapt them?

3) Which of the proposed algorithms, and under which conditions, would still achieve a solution to the problem? Would the solution still be optimal?

3.6. Appendices

Source code of the main units of the practice (algorithms, generation of successors, verification of repetitions, etc.) and anything else that might be considered of interest[1]

4. Implementation notes

Implementation can take place using any programming language as long as the choice is justified. The use of support libraries (e.g. AIMA https://github.com/aimacode) will be allowed at the discretion of the student.

The code must be delivered together with an execution script that allows compilation (if necessary) and execution by the teaching staff, including configuration of all possible dependencies with used libraries and/or programming environments (if applies). In addition, during the face-to-face evaluation process, the student will be responsible for having prepared a ready-to-show version, fully functional, and installed either in their own personal laptop, or in the computers available at the corresponding group lab (if this is the case).

4.1. Input specification

The program must allow reading a text file that represents a map of the terrain, as well as initial and target positions. The map will be rectangular of possible different sizes N x M.

Among the materials attached to these instructions, an example file "exampleMap.txt" is provided with the input specification. The format of this file must be maintained to enable the correction process, although as indicated above, the size and configuration of the map, as well as the initial and final positions, can be modifiable.

In the problem's specification file, the first line indicates the size of the map (rows, columns). Next, the following lines represent the map configuration. Finally, the last two lines respectively indicate the initial and target positions and orientations *(x, y, o)*, $x = 0 \ldots N - 1$, $y = 0 \ldots M - 1$, o = {0:'North', 1:'Northeast', 2:'East', 3:'Southeast', 4:'South', 5:'Southwest', 6:'West' or 7:'Northwest'}. For easiness of implementation, it will be assumed that the robot's initial orientation is always "North". Also, for the target position, the special orientation code '8' can be used to indicate that target orientation is irrelevant.

[1]The complete code will still be delivered, separately, through the mechanisms enabled for this purpose

In the corresponding "main" file or execution script of the developed source code, it should also be possible to select the chosen search method, and the heuristic function to be used (in the case of the A* algorithm). Optionally, the program could have a small graphical user interface for configuring the inputs and/or show the output trace, although this is NOT RECOMMENDED. The design of the user interface is not the objective of the work, and therefore developments exceeding the minimum requirements will not be valued for the final grade.

4.2. Output specifications

The program's output must allow following the execution trace of the selected algorithm. At the very minimum it should display the final achieved solution showing the path from the initial state to the target state indicating all the followed steps. If it is not possible to find a solution to the problem, an informative message will be displayed, together with the path from the initial state to the last examined node.

The use of the command line is highly recommended. Recall that the development of more elaborated user interfaces is discouraged to avoid deviating from the real objectives of the practice. The following notation is expected for a path involving *N* + 1 nodes, where index *0* corresponds to the initial (root) node:

Node 0 (starting node)
Operator 1
Node 1
…
Node i
Operator i
Node i+1
Operator i+1
…
Node N (final node)

Total number of items in explored list: xx
Total number of items in frontier: xx

Each node will be represented as:

- Blind search algorithms: (*d*, *g(n)*, *op*, *S*)
- Algorithm A*: (*d*, *g(n)*, *op*, *h(n)*, *S*)

respectively, where *d* stands for the corresponding depth of the node in the generated search tree, *g(n)* is the accumulated path cost from the initial node, *op* is the applied operator that led to the creation of that node, *h(n)* is the value of the heuristic function (for A*), and *S* represents its state[2].

Notice that, in addition to the final path, the total number of nodes/states contained at the final explored and frontier lists at the end of the execution should be printed as well, as an indication of the effective execution cost.

Additional debug information might be recommended to follow up the search process at the implementation stage. This might involve, for example, the iteration number, the list of generated successor nodes, messages in case of backtracking, status of explored and/or frontier, counter of explored and/or created nodes, etc. This information might as well be included under Section 3.4.1. of the final report document, but it is not mandatory (e.g. if incurring in the imposed limit of maximum 10 pages).

5. Evaluation and deliverables

- The final memory report, as well as all the associated source code (duly documented) will be delivered through the mechanisms enabled through the virtual teaching platform enabled for each lab's group. A GitHub repo for the source code is also permitted
- The work will be carried out compulsorily in groups of **3 people**.
- The deadline for the delivery of this lab is on **October 9th 2025 (16:59 p.m.)** i.e. one week after the last dedicated lab session, right before starting of theory session 4.
- Finally, it is recalled that the total or partial copying of the materials between different groups will automatically lead to qualification of 0 in the course.

[2]Intentionally left abstract here, as this will depend on the specific representation chosen on Section 3.3.1