Artificial Intelligence Practice: Heuristic Search in Radars

Artificial Intelligence

Bachelor in Computer Science and Engineering

2024-205

Content

1. Problem Statement

In the field of aerial navigation and remote sensing, radar (Radio Detection and Ranging) is a fundamental system based on the emission and reception of electromagnetic waves that enables the detection, localization, and tracking of moving objects at a certain distance (see Figure 1). Its operation is based on the emission of radiofrequency pulses that, upon impacting an object, are reflected back to the radar, allowing the estimation of the position, velocity, and direction of movement associated with a particular object.

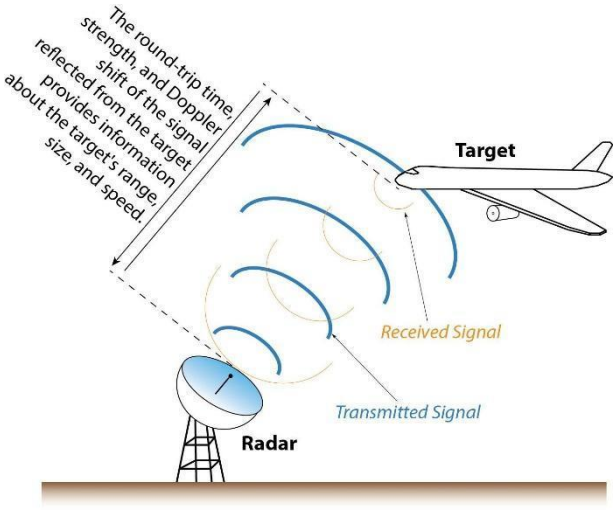

Figure 1: Radar operation.

Radars possess several characteristics that define them, including:

- Range and accuracy
- Capability to operate under adverse conditions
- Object differentiation
- Speed estimation

Among these, range and accuracy are perhaps the most critical fundamental properties of a radar. For this reason, engineers from various fields developed what is known today as the radar equation, a mathematical tool that allows determining the maximum detection distance of a specific radar, based on its physical and operational properties. Mathematically:

$$R_{máx} = \left(\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 P_{mín} L}\right)^{\frac{1}{4}} \quad (1)$$

Where:

- R_{max}: Maximum detection range (maximum distance) [m].
- P_t: Transmitted power [W].
- G: Antenna gain (dimensionless).
- λ: Wavelength of the signal [m].
- σ: Radar cross-section (target reflectivity) [m²].
- $P_{mín}$: Minimum detectable power by the receiver [W].
- L: Additional losses due to weather and impedance (dimensionless: $L \in \mathbb{Z}^+$).

Using the radar equation (1), we can obtain a very good approximation of the distance at which the radar will cease to effectively detect objects, thus implicitly providing information about its effective operating range.

In the proposed problem, a stealth spy plane (stealth aircraft) is being tested by the military. Its design employs what is known as *stealth technology*, primarily allowing it to absorb and deflect radar emissions, thus preventing the "bounce back" of radiofrequency pulses that enables radar systems to identify moving objects. However, it is not completely invisible. The presence of multiple radars and the advanced technology behind them means that the plane cannot fly freely over any area without the risk of being detected. The objective of the spy plane is to fly over and photograph various critical regions from the air without being detected. To achieve this, the military intends to develop a software tool that provides the plane with a probability map of being detected. Using heuristic search algorithms, the plane can then plan a route that allows it to visit the critical zones while minimizing total radar exposure.

To build this detection map, the military has defined the following design guidelines:

1. The map will be a discrete mesh (grid) region consisting of $H \times W$ blocks, where H is the number of blocks in each column of the grid, and W is the number of blocks in each row of the grid.

2. The grid will cover a real-world area bounded by geodetic coordinates; that is, the area covered by the grid will be rectangular in shape, with coordinates (lat_0, lon_0) for the top-left corner and (lat_1, lon_1) for the bottom-right corner. For example: consider an area bounded by {(lat_0, lon_0), (lat_1, lon_1)} ={(40.10°, -3.18°), (40.03°, -3.07°)}, where a grid of H = 20 and W = 30 is to be constructed. The vertical latitude range $\Delta_{lat} = lat_0 - lat_1$ = 40.10° - 40.03° = 0.07° will be divided among H = 20 points, with the first and last points being h_0 = 40.03° y h_{19} = 40.10°, respectively. Similarly, the horizontal longitude range $\Delta_{lon} = lon_1 - lon_0 = -3.07° - (-3.18°) = 0.11°$ will be divided among W = 30 points, with the first and last points being $w_0 = -3.18°$ y $w_{29} = -3.07°$, respectively. For more details, see Figure 2.

3. Each radar will have coordinates $R_i = (lat_i, lon_i)$ located within the bounding area defined above.

4. Each point (block or cell) of the grid will encode the probability of being detected by any of the radars. The possibility (note: not the probability yet) of being detected by a particular radar is modeled using a multivariate Gaussian distribution, whose mathematical formulation is provided below:

$$\Psi(x_1, x_2, \ldots, x_n) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \qquad (2)$$

Where $\Psi(x_1, x_2, \ldots, x_n)$ represents the possibility (or likelihood, not the probability), Σ represents the covariance matrix, and μ represents the mean vector $\mu = \{\mu_x 1, \mu_x 2, \ldots, \mu_x n\}$. In our problem, the mean μ will correspond to the radar's coordinates, and the covariance matrix Σ will be generated automatically. This implies that our multivariate Gaussian distribution will be two-dimensional, which simplifies $\Psi(x_1, x_2, \ldots, x_n)$ to $\Psi(x_1, x_2) = \Psi(lat, lon)$. Since there may be more than one radar, to calculate the detection possibility at a specific point, we will use the following expression:

$$\Psi^*(lat, lon) = max_{\Psi_i^*}\{\Psi_i^*(lat, lon)\}_{i=1}^{N_r} \qquad (3)$$

Where:

$$\Psi_i^*(lat, lon) = \{\Psi_i(lat, lon) \; if \; d \leq R_{max}, 0 \; if \; d > R_{max} \qquad (4)$$

Thus, Ψ_i^* (lat, lon) represents the maximum possibility of being detected by each of the existing i-radars. This can be calculated by computing the possibility for each of the i-radars (denoted as Ψ_i*, where N_r represents the number of radars), storing only the Ψ_i* value that maximizes the detection possibility. Finally, it should be noted that Ψ* will be assigned a value of 0 if the distance between the point where we want to evaluate the detection possibility and the radar that maximizes the possibility is greater than the maximum detection range R_{max} determined by the radar equation (1). To calculate the distance d between the point and the radar, the two-dimensional Euclidean distance will be used:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (5)$$

It is important to remember that the Euclidean distance is being calculated between geodetic coordinates, meaning the units are in decimal degrees. However, the radar equation uses meters as the unit of distance. Therefore, an approximation of the distance between two points given their geodetic coordinates would be calculated as follows:

$$d = K\sqrt{(lat_{point} - lat_{radar})^2 + (lon_{point} - lon_{radar})^2} \quad (6)$$

Where K = 111,000 meters, which is a reasonable approximation for converting one geodetic degree into meters. This approximation works better for degrees of latitude than for longitude, since longitude lines (meridians) converge as they approach the poles, meaning the approximation provided by K is not constant. For the purposes of this problem, it will be sufficient to apply formula (6) to estimate the distance in meters.

5. To calculate the probability from the possibility, each value $\Psi^* \in [0, +\infty]$ will be scaled using the MinMax technique, which operates as follows:

$$\Psi^*_{scaled} = \frac{\Psi^* - \Psi^*_{mín}}{\Psi^*_{max} - \Psi^*_{mín}} \quad (7)$$

Thus achieving that Ψ^* scaled $\in [0, 1]$. After scaling in this way, there will be cells that have a detection value equal to 0. This complicates the design of admissible heuristics, as there could be paths with a total cost equal to zero, significantly increasing the difficulty of estimating costs without overestimating the actual value. To prevent cells from having a probability equal to zero, a very small value ϵ is established, allowing the addition of a residual cost to any cell with zero cost. Starting from equation (7), the following modification is made:

$$\Psi^*_{scaled} = \frac{\Psi^* - \Psi^*_{mín}}{\Psi^*_{max} - \Psi^*_{mín}}(1 - \epsilon) + \epsilon \quad (8)$$

In this way, we ensure that $\Psi^* \in [\epsilon, 1]$, avoiding cells with zero values and, therefore, preventing transitions in the search space without an associated cost.

6. The spy plane will not be able to move through any cell of the grid if its detection probability exceeds a given threshold/tolerance, which will be provided through the program's arguments. If the threshold is set to 0.5, it will mean that the plane cannot transition into cells with a probability value higher than 0.5. This is intended to model the risk taken by the spy plane when flying closer or further from the radar detection fields.
7. The spy plane must visit a sequence of POIs (Points Of Interest). Initially, a "high-level plan" must be established, deciding the order in which the points will be visited. Then, the A* algorithm and heuristic search will be used as algorithmic tools to find the minimum-cost path between each pair of points. To better understand this, consider the following example: POI = {p1, p2, p3, p4, p5}. Suppose the chosen visiting order is {p1, p3, p5, p2, p4}. To construct the spy plane's plan, it would be necessary to perform A*(p1, p3), A*(p3, p5), A*(p5, p2), and A*(p2, p4). The concatenation of each of these paths will form the final solution to the problem (see Figure 3).

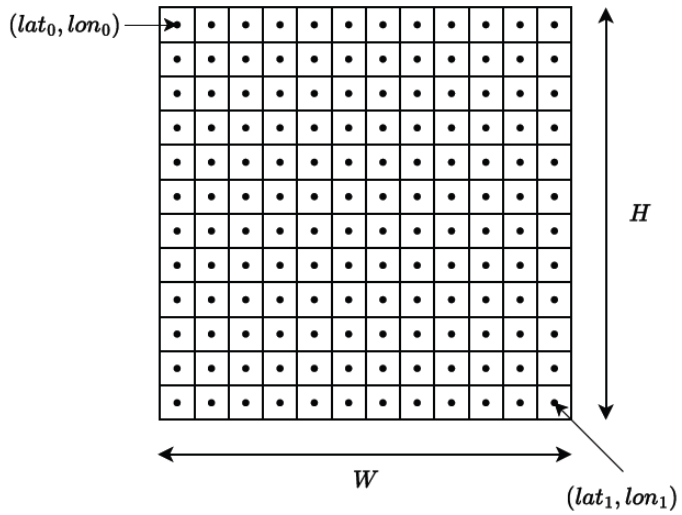

Figure 2: Example of a grid covering an area bounded by (lat0, lon0) and (lat1, lon1), with dimensions H × W.

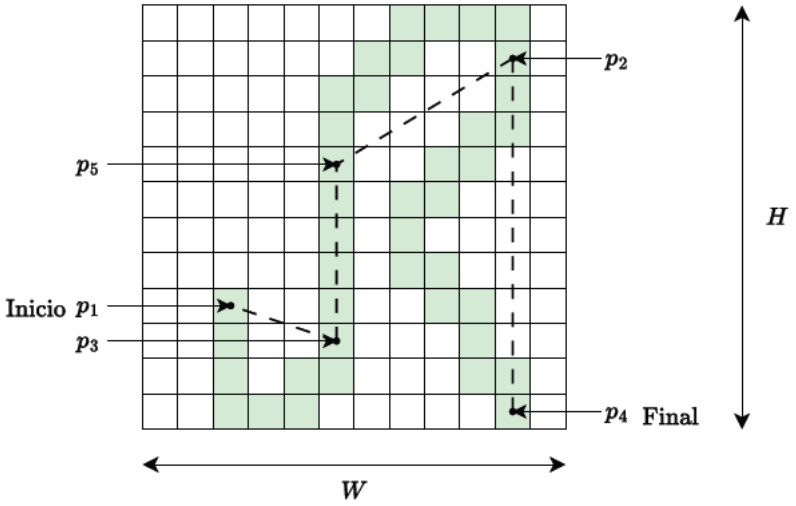

Figure 3: Example of high-level planning and local path planning using A. The green cells represent the selected steps that the plane must take. It is important to remember that each cell in the grid has a specific detection probability, which is why the green path follows the route that minimizes detection.*

2. Tasks

To achieve the proposed objectives, the practice is broken down into the following sequence of tasks, which must be completed as separate but interrelated exercises.

2.1 Modelling the Search Problem

For this section, the following is required:

1. Mathematically model the states and operators of the problem. This must be documented in the project report and should cover all aspects relevant to solving the problem, using rigor and mathematical formality. In the models and formal definitions, there should be no ambiguity or excessive explanatory text without contributions. Using symbols, equations, sets, etc., will ensure a clear and concise model of the problem.

2. Define the initial state and the characteristics of the final state of the search problem, using the state model developed in the previous section.

3. Design and formally formulate at least two admissible heuristics for the search problem. It is very important that students ensure the heuristics are admissible, meaning that the estimated cost by the heuristics never overestimates the actual cost of the problem. There is no standard procedure to guarantee admissibility, so students must demonstrate/justify the admissibility of their heuristics. Designing non-admissible heuristics that work for the problem will be accepted but will not receive full credit.

2.2 Map and Search Space Generation

For this section, the following are required:

1. Implement the generation of the map (grid) that defines the search space. To do this, you will need to develop the calculation of the detection probability for each cell, using the radar equation (1), the detection possibility equation (3), and the scaling equation (8), allowing the construction of a map stored in memory with each cell's detection value.
2. Implement the generation of a graph that defines the search space from the detection map. Since the implementation of the A* algorithm operates directly on graphs, students must build a graph based on the detection map. Since the possible movements the spy plane can make are toward adjacent cells (up, down, left, and right), the objective is to generate as many vertices as the grid has cells, and as many edges as there are possible connections between neighboring cells for each cell in particular. Since during the map generation the cells hold a detection probability value, the edges connecting the graph vertices will take as weights (costs) the value of the vertex (cell) being transitioned to. That is, if vertex v_i represents a cell with a detection probability of 0.88 and vertex v_j represents a cell with a detection probability of 0.27, the edge from i to j will have a weight $e_{ij} = 0.27$, and the edge from j to i will have an associated weight $e_{ji} = 0.88$. For more information, see Figure 4.

2.3 Integration of the Search Algorithm

Using the implementation of the A* algorithm from the Python **networkx** library, called **astar_path**, the student must be able to integrate the algorithm into the search system to solve the pathfinding problem. It is important to remember that the heuristics designed in Section 2.1 must be implemented to be applied by the A* algorithm.

2.4 Experimentation with Different Scenarios

Along with the project description, various scenarios will be provided. These scenarios feature grids of different dimensions with different sets of radars distributed across the surface. In these scenarios, the sequence of POIs that the spy plane must fly over will also be provided, intentionally selected to test the correct implementation of the system.

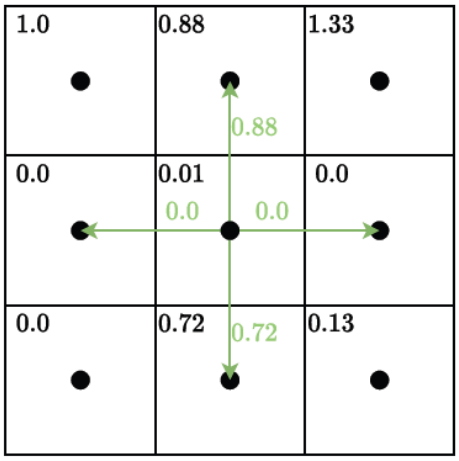

Figure 4: Example of transitions from a node in the graph representing the search space. The weight associated with the edges corresponds to the value of the destination cell.

Along with conducting the experiments corresponding to the scenarios provided with the project description, the student must also carry out an additional set of tests, using scenarios and POIs of their choice, to validate the correct functioning of the system.

3. Provided code

A base code written in Python (the language in which the system implementation will be developed) is provided. Students must work with this code to implement the proposed search system. The student's objective is to complete the implementation of certain key functions that are not yet implemented but are critical to the system. Basic functionalities for generating graphs to visualize the solution, as well as functionalities for the random generation of radars within an area, and the basic structure of some of the problem classes, are already provided by the base code.

Students are strongly encouraged to use Python's NumPy library for array operations, as it offers significantly better performance than native Python lists and simplifies the implementation of functions that operate on matrices and/or vectors. NumPy is a very popular and widely used library that every Python developer should know and be comfortable with.

In addition to completing the required functions, students are allowed (and encouraged) to implement any additional auxiliary functions they deem necessary, either to refactor code or to improve system understanding.

It is reminded that good developer practice includes commenting on the code as much as possible (without being excessive), to facilitate the understanding of the code by both the students and the instructor.

As previously mentioned in this document, along with the base code, a JSON file is also provided. This file contains the configuration of different scenarios that students must test to verify the correct functioning of their systems. Students must also extend this battery of tests by including additional scenarios that evaluate the system's robustness. Each of these scenarios follows the following structure:

```
{
    "scenario_0": {                          //Scenario identifier
        "max_lat": 37.29139325161781,        // Maximum latitude on the map
        "min_lat": 37.21979775354181,        // Minimum latitude on the map
        "max_lon": -115.78524417824534,      // Maximum longitude on the map
        "min_lon": -115.8885843284312,       // Minimum longitude on the map
        "W": 16,                        // Number of horizontal points in the grid
        "H": 16,                        // Number of vertical points in the grid
        "n_radars": 1,                  // Number of radars randomly distributed
        "POIs": [[37.21979775, -115.88858433], // Points of interest to visit
                 [37.21979775, -115.81969089]]
    }
},
```

Thus, the student must replicate this structure to add any additional scenarios they consider relevant. Finally, it is important to highlight that the development IDE to be used during the practice will be Visual Studio Code, since the base code includes a launch.json file that allows the system to be run in debug mode, automating the passing of arguments to the program and facilitating error debugging. Additionally, the use of PyCharm as an alternative development IDE will also be permitted.

4. Evaluation

The following sections detail the different scores associated with each of the parts that the student must complete. In total, the practical block accounts for **7 points** of the project, while the final report represents the remaining **3 points**.

4.1 Problem Modelling

This section is worth a total of **2 points**, distributed as follows:

- Mathematically model the states and operators of the problem **(0.7 points).**
- Define the initial state and the characteristics of an acceptance (goal) state, using the mathematical model developed previously **(0.3 points).**
- Design and formulate at least two admissible heuristics for the search problem **(1 point).**

4.2 Map Implementation

This section is worth a total of **2 points**, distributed as follows:

- Implement the generation of the map (grid) that defines the search space **(1 point).**
- Implement the generation of the graph from the grid and the possible transitions between different cells **(1 point).**

4.3 Search System Implementation

This section is worth a total of **2 points**, distributed as follows:

- Integrate the A* search algorithm from the **networkx** library into the system, using the admissible heuristics previously designed **(2 points)**.

4.4 Experimentation

This section is worth a total of **1 point**, distributed as follows:

- Test different scenarios and document the results in the report. The use of graphs and tables will be positively evaluated **(1 point)**.

4.5 Report

The project report accounts for **3 points** out of the total 10 points. The objective of the report is to thoroughly document the solution that was designed and implemented, highlighting the most relevant technical aspects. The breakdown is:

- Content and explanation of the system (1.5 points).
- Structure and presentation of the document (0.5 points).
- Description, justification, and reflection on the use of generative AI (1 point).

Special emphasis is placed on the need for students to carefully ensure the quality of their report, with a particular focus on clear and formal explanations, as well as style, aesthetics, and presentation. A good report should be capable of explaining the developed system without requiring significant reference to the code itself. This does not mean that code should be included in the report; students must avoid adding code to the document.

The reports must include, at a minimum:

- A cover page
- An index (table of contents)
- An introduction
- Main content
- Experiments
- Conclusions

Page numbers must be included on all pages except for the cover page. If the student refers to scientific literature, sources must be cited, and the citation format is at the student's choosing.

5. Rules

This project must be completed in groups of 3 students. The project must be submitted as a compressed ZIP file by each group through the link provided on Aula Global. The name of the ZIP file must follow the format: "AI Practice 2025 <st-code1><st-code2><st-code3>.zip", where <st-codex> are the last six digits of the NIA of each group member.

- Example: AI Practice 2025 123456 345678 456789.zip

The ZIP file must include:

- A folder containing the corresponding code, separated into .PY and .JSON files.
- The technical report of the project compiled in PDF format.

- (Optional) A README text file (.TXT) with any instructions or notes necessary to facilitate the execution and correction of the system, if deemed appropriate by the students.

A submission link will be enabled on Aula Global, and students must upload the described ZIP file through it. The deadline for submitting the project is May 15th at 23:59. Submissions via email or after the deadline will not be accepted.

6. Use of Generative AI

The use of generative AI tools is permitted, provided that their use is transparent and properly documented. In particular, AI may be used for:

- Searching for relevant information or examples related to the project.
- Obtaining explanations about methods, libraries, or specific concepts.
- Generating ideas or possible approaches to solving the problem, as long as the student critically analyzes and adapts them to their own development.

Restrictions:

- It is not allowed to use generative AI to create code, algorithms, or text fragments that are incorporated directly into the system or the report without modifications or without understanding by the student.
- Students are expected to design, implement, and document the solution in a reasoned and justified manner.

Students must include an additional section at the end of the project report, where they transparently describe their use of generative AI. In this section, they must specify:

1. For which aspects of the project they used AI.
2. How AI facilitated the completion of the project.
3. How they verified and adapted the information obtained.

This statement will be positively evaluated in the final grading of the project, depending on the quality and depth of the analysis provided.