|  |
| --- |
| Artificial and Computational Intelligence (S2-23\_AIMLCZG557) |
| **Assignment 1 - Problem Statement 1** |
| By Group 1, 11th August 2024 |

**Submitted by Group 1 with 100% contribution from all participants:**

*Name:* ***Reddy Balaji .C*** *(BITS ID:* ***2023AC05862****); Email:* [*2023ac05862@wilp.bits-pilani.ac.in*](mailto:2023ac05862@wilp.bits-pilani.ac.in)

*Name:* ***Saurabh Jalendra*** *(BITS ID:* ***2023AC05912****); Email:* [*2023ac05912@wilp.bits-pilani.ac.in*](mailto:2023ac05912@wilp.bits-pilani.ac.in)

*Name:* ***Tushar Shandilya*** *(BITS ID:* ***2023AC05573****); Email:* [*2023ac05573@wilp.bits-pilani.ac.in*](mailto:2023ac05573@wilp.bits-pilani.ac.in)

*Name:* ***Bandana Kumari*** *(BITS ID:* ***2023AC05879****); Email:* [*2023ac05879@wilp.bits-pilani.ac.in*](mailto:2023ac05879@wilp.bits-pilani.ac.in)

*Name:* ***Monica Malik*** *(BITS ID:* ***2023AC05875****); Email:* [*2023ac05875@wilp.bits-pilani.ac.in*](mailto:2023ac05875@wilp.bits-pilani.ac.in)

Contents

[Problem Description 2](#_Toc174274095)

[Problem Representation 2](#_Toc174274096)

[**Grid representation with points** 3](#_Toc174274097)

[**Assumptions for calculations** 3](#_Toc174274098)

[**Points Representation and Handling of Buildings and Roadblocks and Diagonal Movements** 3](#_Toc174274099)

[**Steps** 4](#_Toc174274100)

[Algorithm – UCS 5](#_Toc174274101)

[**Algorithm Tracing:** 5](#_Toc174274102)

[Algorithm – RBFS 7](#_Toc174274103)

[**Algorithm Tracing** 7](#_Toc174274104)

[Problem Statement Solution 10](#_Toc174274105)

[Solution-1 (PEAS Description) 10](#_Toc174274106)

[Solution-2 (RBFS Algorithm reference in Code) 12](#_Toc174274107)

[Solution-3 (UCS Algorithm reference in Code) 12](#_Toc174274108)

[Solution-4 (Path, Cost and Number of Square references in code) 12](#_Toc174274109)

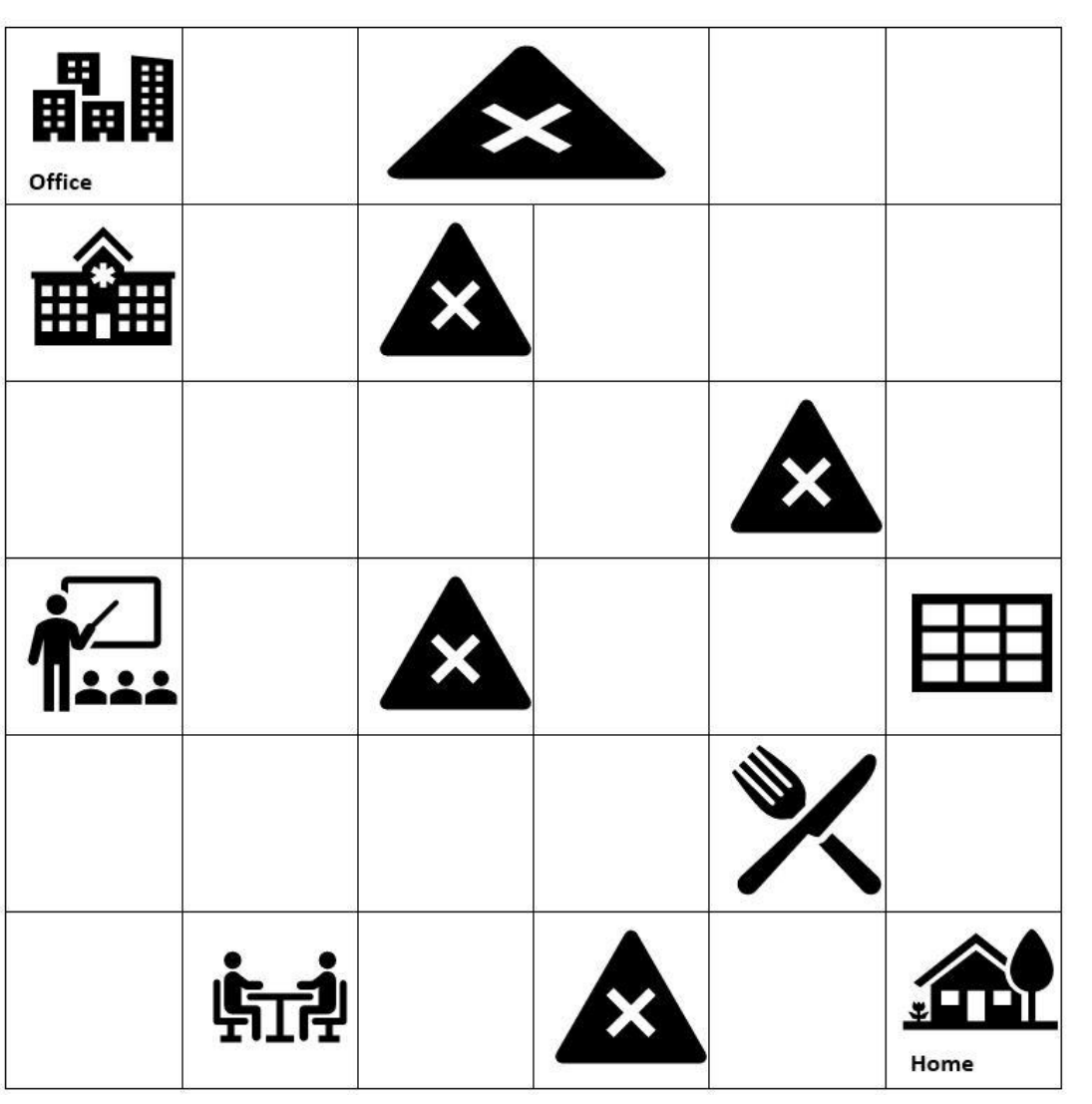
[Solution-5 (Time and Space Complexity reference in code) 12](#_Toc174274110)

[Solution-6 (Comparative Analysis or Findings) 12](#_Toc174274111)

[Conclusion : 13](#_Toc174274112)

## **Problem Description**

**Description**: The agent is an autonomous system designed to navigate a grid-based city map. Its primary function is to determine the shortest, quickest and safest route from the office to home, considering various environmental factors such as roadblocks, buildings and traffic conditions.



## **Problem Representation**

|  |  |
| --- | --- |
| **Input** | Grid: A 2D array representing the layout/map, with the given symbols indicating accessible areas ('**.**'), road blocks ('**X**'), and buildings ('**B**') |
| **Start** | The starting cell - **S** (0,0) - **Office** |
| **Goal** | The destination cell – **G** (5,5) - **Home** |
| **Output** | * Path. The shortest and safest path from Start to Goal, or an indication that no such path exists. * Expanded Nodes * Time Complexity * Space Complexity |

### **Grid representation with points**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S**  (0,0) | .  (0,1) | **X**  (0,2) | **X**  (0,3) | .  (0,4) | .  (0,5) |
| **B**  (1,0) | .  (1,1) | **X**  (1,2) | .  (1,3) | .  (1,4) | .  (1,5) |
| .  (2,0) | .  (2,1) | .  (2,2) | .  (2,3) | **X**  (2,4) | .  (2,5) |
| **B**  (3,0) | .  (3,1) | **X**  (3,2) | .  (3,3) | .  (3,4) | **B**  (3,5) |
| .  (4,0) | .  (4,1) | .  (4,2) | .  (4,3) | **B**  (4,4) | .  (4,5) |
| .  (5,0) | **B**  (5,1) | .  (5,2) | **X**  (5,3) | .  (5,4) | **G**  (5,5) |

### **Assumptions for calculations**

* “Adjacent to” – This is understood as only vertical and horizontal proximity. Diagonal adjacency is not considered.
* If both **B** and **X** are adjacent to the traversing node, then the points will be considered   
  as -5 + 3 = -2.
* If more than one building or road blocks are there then sum of the points will be considered. For example, if there are 2 buildings then the points will be considered as -5 + (-5) = -10 and if two road blocks then point will be 3+3 = 6.
* As the problem statement mentioning maximizing points, the algorithms require a cost function to operate correctly. To align with this, the points have been made negative, effectively turning the maximization problem into a minimization one. This adjustment ensures that the algorithms can find the most optimal path based on the lowest cost, which corresponds to the highest points when interpreted in the context of the original problem statement.

### **Points Representation and Handling of Buildings and Roadblocks and Diagonal Movements**

**Building (B):**

*Reward (Cost):* Being adjacent to a building provides a negative cost of -5. This represents increased safety, which is beneficial for the agent.

*Explanation:* The presence of buildings offers a strategic advantage in the pathfinding problem, as they provide safety. This is reflected in the negative cost, encouraging the agent to move near buildings when possible.

**Roadblock (X):**

*Penalty (Cost):* Being adjacent to a roadblock incurs an additional cost of +3. This represents the difficulty or danger of navigating near obstacles.

*Explanation:* Roadblocks are obstacles that hinder movement and pose risks, so the agent is penalized for being near them. The positive cost reflects this disadvantage in the pathfinding process.

**Diagonal Movement:**

*Cost:* When diagonal movement is allowed, it has a higher cost of 3 compared to straight movement, which typically has a lower or zero cost.

*Explanation:* Diagonal movement is considered more challenging or resource-intensive, hence the increased cost. This rule ensures that diagonal moves are only chosen when they provide a clear advantage in reducing the overall path cost.

These points can be integrated into the documentation to clarify how the different elements of the grid contribute to the agent's decision-making process within the algorithms.

* **1** cost for each block movement
* **-5** cost for each adjacent building cell (**B**).
* **+3** cost for each adjacent roadblock cell (**X**).
* **-2** cost if both a building (**B**) and a roadblock (**X**) are adjacent.
* **-10** cost if there are two buildings (**B**) adjacent.
* **+6** cost if there are 2 roadblocks adjacent to a cell

### **Steps**

1. Setting the initial state
2. Representation of the problem as a grid
3. Defining the total points for various adjacency scenarios
4. Function to fetch the neighbours
5. Logic for the valid moves – straight and the diagonal move in case of absence of straight path
6. Goal state to check whether goal is reached or not
7. Defining heuristic function based on Manhattan distance
8. Expanding nodes based on RBFS / UCS criteria
9. **RBFS** algorithm
10. **UCS** algorithm
11. Calculate space and time complexity

## **Algorithm – UCS**

**g(n)** – cost to move from one block to another + cost for encountering adjacent building + cost for encountering adjacent roadblock.

**Cost Function- f(n) = g(n)**

### **Algorithm Tracing:**

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | Open List – Frontier | Closed List / Visited Nodes | Goal Test |
| 1 | (0,0) | - | Fail (0,0) |
| 2 | (0,1) | (0,0) | Fail (0,1) |
| 3 | (1,1) | (0,1) | Fail (1,1) |
| 4 | (2,1) | (1,1) | Fail (2,1) |
| 5 | (2,0) (3,1) (2,2) | (2,1) | Fail (2,0) |
| 6 | (3,1) (2,2) | (2,0) | Fail (3,1) |
| 7 | (4,1) (2,2) | (3,1) | Fail (4,1) |
| 8 | (4,2) (4,0) (2,2) | (4,1) | Fail (4,2) |
| 9 | (4,3) (5,2) (4,0) (2,2) | (4,2) | Fail (4,3) |
| 10 | (3,3) (5,2) (4,0) (2,2) | (4,3) | Fail (3,3) |
| 11 | (3,4) (2,3) (5,2) (4,0) (2,2) | (3,3) | Fail (3,4) |
| 12 | (4,5) (2,3) (5,2) (4,0) (2,2) | (3,4) | Fail (4,5) |
| 13 | (5,5) (2,3) (5,2) (4,0) (2,2) | (4,5) | Pass(5,5) |
| 14 | (2,3) (5,2) (4,0) (2,2) | (5,5) |  |

**Tree representation (UCS):**

## **Algorithm – RBFS**

**g(n)** - cost to move from one block to another + cost for encountering adjacent building + cost for encountering adjacent roadblock.

**h(n) –** The sum of absolute distance from the current state to the final state (Manhattan Distance as heuristic function).

**h(n)=∣x2​−x1​∣+∣y2​−y1​∣**

**Cost Function-f(n) = g(n) + h(n)**

### **Algorithm Tracing**

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | Open List | Closed List | Goal Test |
| 1 | (0,0) | - | Fail (0,0) |
| 2 | (0,1) | (0,0) | Fail (0,1) |
| 3 | (1,1) | (0,1) | Fail (1,1) |
| 4 | (2,1) | (1,1) | Fail (2,1) |
| 5 | (3,1) (2,0) (2,2) | (2,1) | Fail (3,1) |
| 6 | (4,1) (2,0) (2,2) | (3,1) | Fail (4,1) |
| 7. | (4,2) (4,0) (2,0) (2,2) | (4,1) | Fail (4,2) |
| 8. | (5,2) (4,3) (4,0) (2,0) (2,2) | (4,2) | Fail (5,2) |
| 9. | (4,3) (4,0) (2,0) (2,2) | (5,2) | Fail (4,3) |
| 10. | (3,3) (4,0) (2,0) (2,2) | (4,3) | Fail (3,3) |
| 11. | (2,3) (3,4) (4,0) (2,0) (2,2) | (3,3) | Fail (2,3) |
| 12. | (1,3) (2,2) (3,4) (4,0) (2,0) | (2,3) | Fail (1,3) |
| 13. | (1,4) (2,2) (3,4) (4,0) (2,0) | (1,3) | Fail (1,4) |
| 14. | (0,4) (1,5) (2,2) (3,4) (4,0) (2,0) | (1,4) | Fail (0,4) |
| 15. | (0,5) (1,5) (2,2) (3,4) (4,0) (2,0) | (0,4) | Fail (0,5) |
| 16. | (1,5) (1,5) (2,2) (3,4) (4,0) (2,0) | (0,5) | Fail (1,5) |
| 17. | (2,5) (1,5) (2,2) (3,4) (4,0) (2,0) | (1,5) | Fail (2,5) |
| 18. | (3,4) (1,5) (2,2) (3,4) (4,0) (2,0) | (2,5) | Fail (3,4) |
| 19. | (4,5) (1,5) (2,2) (3,4) (4,0) (2,0) | (3,4) | Fail (4,5) |
| 20. | (5,5) (1,5) (2,2) (3,4) (4,0) (2,0) | (4,5) | Pass (5,5) |
| 21. | (1,5) (2,2) (3,4) (4,0) (2,0) | (5,5) |  |

**Tree representation (RBFS)**

*Continuation of the tree from (0,5) node*

## **Problem Statement Solution**

Solution-1 (PEAS Description)**-** The PEAS model provides a understanding of the GPS Navigation Agent's functioning, highlighting its objectives, the environment it operates, the actions it can perform and the information it depends on to make decisions. This model is essential for developing an efficient and effective pathfinding solution which will meet the specified performance criteria.

1. **Performance Measure:**

The performance of the GPS Navigation Agent is evaluated based on the following criteria:

1. **Safely Reaching the Destination**: The agent must ensure that it successfully navigates from the starting point (office) to the destination (home) without encountering any obstacles that could impede its progress, such as buildings or blocked paths.
2. **Shortest Path**: The agent must minimize the number of grid squares traversed. The goal is to find the most direct route that avoids unnecessary detours.
3. **Quickest Route**: The agent must consider time as a factor, seeking to minimize the overall time spent traveling. This could involve selecting paths with fewer penalties or avoiding heavily blocked areas.
4. **Safest Path**: The agent prioritizes safety by avoiding proximity to roadblocks or other hazardous elements in the grid. It aims to minimize risk by choosing routes that steer clear of dangerous areas.
5. **Minimized Cost**: Cost in this context includes the sum of penalties incurred from diagonal movements, proximity to roadblocks and any additional time or distance travelled. The agent must choose a path that keeps these costs as low as possible while achieving the other objectives.
6. **Environment:**

The environment in which the GPS Navigation Agent operates is a structured grid-based representation of a city including the following elements:

1. **Grid-Based City Map**: The city is represented as a grid where each cell or square represents a specific location in the city.
2. **Buildings**: Certain grid squares are occupied by buildings. These squares are impassable, and the agent must find a way around them.
3. **Blocked Paths**: Some grid squares are marked as blocked due to road maintenance or other reasons. These are temporary obstacles that the agent cannot pass through.
4. **Roadblocks**: Adjacent to some grid squares are roadblocks, which impose additional risks or penalties when nearby. The agent must factor these into its pathfinding to ensure the route is safe.
5. **Traffic Conditions**: The environment may also include varying traffic conditions, which can affect the speed and safety of different routes. The agent needs to account for these when planning the path.
6. **Actuator:**

The actuator defines the actions that the GPS Navigation Agent can perform within the environment:

1. **Movement Directions**: The agent can move in four primary directions: up, down, left, and right. Diagonal movement is also possible but comes with a penalty.

Each move transitions the agent from one grid square to another, and the decision to move in a particular direction is determined by the algorithms in use (e.g., Recursive Best First Search or Uniform Cost Search).

1. **Path Execution**: After determining the optimal path, the agent must execute this path, moving step by step from the starting point to the destination.
2. **Decision Making**: The agent continuously evaluates the environment and updates its decisions based on new information or changing conditions, ensuring that it adapts to dynamic situations such as newly blocked paths or changes in traffic.
3. **Sensor:**

The sensor system provides the GPS Navigation Agent with real-time information about its environment allowing it to make informed decisions:

1. **City Map Information**:
   * The sensor system provides a comprehensive view of the grid-based city map, including the locations of buildings, blocked paths, and roadblocks.
   * It ensures that the agent has accurate data on the static elements of the environment.
2. **Environmental Sensors**: These sensors detect changes in the environment. For example: new obstacles or roadblocks, enabling the agent to adapt its pathfinding strategy in real-time.
3. **Proximity Sensors**: These sensors identify nearby buildings or roadblocks providing data that helps the agent calculate the safety and risk factors associated with different paths.
4. **Real-Time Grid Status**: The sensors continuously update the agent on the current state of the grid including traffic conditions and the status of various paths to ensure the agent's decisions are based on the latest information.

Solution-2 (RBFS Algorithm reference in Code)**:**

Implementation of RBFS algorithm has been done in Section- “**2**. **Definition of Algorithm 1 (Recursive Best First Search - RBFS)”** in the python file (**PS1\_Group1\_ACI\_Assignment.ipynb**) attached with the submission.

Solution-3 (UCS Algorithm reference in Code)**:**

Implementation of UCS algorithm has been done in Section- “**3. Definition of Algorithm 2 (Uniform Cost Search-UCS)”** in the python file (**PS1\_Group1\_ACI\_Assignment.ipynb**) attached with the submission.

Solution-4 (Path, Cost and Number of Square references in code)**:**

Path followed, Total cost g(n) in case of UCS and g(n)+h(n) in case of RBFS and number of squares printing has been implemented in Section **“4. Calling the search algorithms”** in the python file (**PS1\_Group1\_ACI\_Assignment.ipynb**) attached with the submission.

Solution-5 (Time and Space Complexity reference in code)**:**

Calculation of Time and Space Complexity for both UCS and RBFS has been implemented in Section – **“5. Comparative Analysis”** in the python file (**PS1\_Group1\_ACI\_Assignment.ipynb**) attached with the submission.

Solution-6 (Comparative Analysis or Findings)**:**

Comparative analysis is provided for both UCS and RBFS in Section – **“6. Provide your comparative analysis or findings in no more than 3 lines in below section”** in the python file (**PS1\_Group1\_ACI\_Assignment.ipynb**) attached with the submission.

## **Conclusion :**

In the context of the grid-based pathfinding problem, the Recursive Best First Search (RBFS) algorithm efficiently manages memory by using recursion, allowing it to handle smaller grids without significant overhead.

However, in more complex city layouts, RBFS may expand a higher number of nodes compared to Uniform Cost Search (UCS), potentially resulting in longer search times due to its heuristic reliance.

In contrast, UCS consistently provides the shortest path by evaluating all possible paths based on cumulative costs, but it incurs higher memory usage to store all expanded nodes in its priority queue, particularly in densely populated grids with numerous buildings and roadblocks.

*Reference : Python File (****PS1\_Group1\_ACI\_Assignment.ipynb****) with all steps & details*