CSE518 : Artificial Intelligence Project Progress Report

Member1 : Nitant Jain AU2340198

Member2 : Even Patel AU2340241

1. Implementation of the reflex module :

- The reflex module continuously monitors real-time conditions such as battery level, terrain cost, and nearby obstacles during path execution.
- Each reflex is triggered instantly when its condition (e.g., Battery < 20%, or ROCK tile ahead) is detected, ensuring immediate corrective action.
- The module supports five reflex types - Obstacle, Low Battery, Terrain, Obstacle-Aversion Heuristic, and Recharge - each modifying movement or heuristic dynamically.
- Reflex actions include rerouting, skipping unsafe cells, penalizing risky areas, or initiating recharging sequences to maintain safe navigation.
- The reflex logic operates independently of A*, allowing real-time adaptability without restarting the full path planning process.

Based upon our Analysis, the Reflex mechanisms implemented are as follows:

Reflex Type	Condition	Actions
Obstacle Reflex	Rock tile in next step	Avoid move
Low Battery Reflex	Battery < 20%	Stop expansion
Terrain Reflex	High cost	Costly move
Obstacle-Aversion Heuristic Reflex	Near Rock obstacle	Add penalty to heuristic
Recharge Reflex	Battery insufficient	Move to nearest recharge station or abort

2. Implementation of Path planning module :

The path planning of the rover is based on a Hierarchical Strategy to generate battery-conscious and safe route planning based upon the A* algorithm.

I. Base-Level Planning (Battery-aware A* Function):

Function objective : The cost of the path segment between two nodes is minimized to find a safe path (battery-conscious).

Constraints :

1. Rock Avoidance: Completely excludes Rock tiles from the search space.
2. Battery Safety: Prunes any path branch if the projected cost would cause the remaining battery to drop below a 20% threshold.

II. High level Planning (Plan_with_Recharges module) :

Initial Check: Always first attempts a Direct Path to the goal using A* with the current battery level.

Iterative Chaining (If Direct one Fails):

1. Identify Reachable sites : Finds all Recharge Sites (RC's) accessible with the current battery (considering 20% threshold safety check).
2. Selection: Chooses the reachable RC that is closest to the Final Goal (greedy algorithm using the heuristic).
3. Segment Execution: Appends the found A* segment to the RC, simulates Recharge (100% battery), and updates the current position.
4. Repeat: It loops back to attempt a path from the newly charged RC to the Goal.

Goal: Returns a full, cost-optimized, and battery-feasible path, which may consist of multiple segments between Start, RC's, and Goal.

III. Integration of both modules :

Condition	Action Taken	Planning Outcome
B < 20%	Immediate Stop and reroute to find nearby RC.	Priority for battery over goal.
20% ≤ B ≤ 25% & RC nearby (≤ 2 tiles)	Reroute to nearby RC : Plans ahead to stop at the adjacent recharge for efficiency.	Saves run time by optimizing for charging opportunities.
Next Step Cost > B (Unsafe Move)	Replans:Calls the high-level planning module to insert a necessary recharge stop before the greater cost tile.	It ensures that the rover never attempts a move that is not feasible, even if initially planned.

Note : For the code detailed flow understanding, 2 flowcharts are attached in the google drive link.

CSE518_Artificial_Intelligence_Code_Flowchart

3. Explanation of the 4 heuristics :

3.1. Manhattan Distance

- Counts the number of grid steps between any 2 points in both horizontal directions and vertical direction.
- Most effective on motion 4 ways (no diagonals).
- **Formula**: |x1 - x2| + |y1 - y2|

3.2. Euclidean Distance

- Measures the straight-line (diagonal) distance between two points.
- More expensive due to the square root term in the calculation
- **Formula:** sqrt ((x1-x2)^2 + (y1-y2)^2)

3.3. Terrain Aggressive Heuristic :

- It calculates the terrain cost between two points, not just the distance.
- It gathers multiple points along the line in order to find the average terrain cost.
- It penalizes the routes passing through expensive terrains like rocks or sand.
- **Formula :** Distance * (Average terrain cost along the path)

3.4. Adaptive Cost Manhattan(H1) :

- It extends Manhattan by multiplying with an average terrain cost factor ((5+10)/2 = 7.5).
- It maintains balance between flat(low cost =5) and sandy (high cost =10) areas.
- It is though easier to compute.
- **Formula:** $7.5*(|x1 - x2| + |y1 - y2|)$

3.5. Obstacle-Aversion (H2) :

- It works keeping in mind the H1 but adds extra cost when near to the rocks.
- It naturally keeps the path finding rover to stay away from the rock or obstacle.
- It follows an inverse relation i.e. higher penalty for the nearby rocky area.
- **Formula:** (7.5*Manhattan Distance)+(10 / (Min distance to rock)^2)

4. Descriptive Comparison Report :

Heuristic	Efficiency (Nodes)	Optimality (Cost)	Speed (Computation Time)	Consistent	Admissible
Adaptive Cost (H1)	High (few nodes)	Moderate	Fast	Yes	Yes
Euclidean	Low (many nodes)	High (lowest cost)	Moderate	Yes	Yes
Manhattan	Low (many nodes)	High (lowest cost)	Moderate	Yes	Yes
Obstacle Aversion (H2)	Moderate (fewer)	Low (higher cost)	Slow	No	No
Terrain (Aggressive)	Moderate (fewer)	Lowest (least optimal)	Very Slow	No	No

Interpretation Summary :

- **Adaptive Cost (H1):** Best balance between speed, efficiency, and near-optimal paths.
- **Euclidean / Manhattan:** Most optimal but explore more nodes (less efficient).
- **Obstacle Aversion (H2):** Safe but slow and may overestimate cost.
- **Terrain (Aggressive):** Prioritizes fast routes but produces least optimal paths.