Analysis of A* Heuristics for Pathfinding in a Battery-Constrained Rover on a Dynamic Square Grid

Shashwat Gohel (AU2340235), Feni Vithani (AU2340119)
Course: CSE-518 - Artificial Intelligence
Instructor: Prof. Shashi Prabh

***Abstract*—This paper presents a detailed analysis of the A* pathfinding algorithm using four different heuristics on a simulated 20 $\times$ 20 square grid environment. The objective is to evaluate the performance of each heuristic in terms of path cost, computational effort (nodes expanded), and execution time. The simulation involves a moon rover navigating a complex terrain with varying costs, obstacles, and hazards, including a battery constraint that necessitates recharging. We explore the trade-off between heuristic admissibility and computational efficiency, demonstrating that inadmissible heuristics can offer significant speed improvements at the cost of path optimality. Additionally, a reflex agent with backtracking, avoidance learning, and dynamic terrain adaptation ensures robustness in uncertain environments.**

***Index Terms*—A* algorithm, pathfinding, heuristics, battery constraint, dynamic grid, moon rover navigation**

I. Introduction - Problem formulation

Pathfinding in complex environments is a key challenge in artificial intelligence, with applications in robotics, autonomous navigation, and planetary exploration. The A* algorithm efficiently finds the lowest-cost path by combining the actual cost so far, $g(n)$, with an estimated cost to the goal, $h(n)$, using

$$f(n) = g(n) + h(n). \tag{1}$$

The heuristic $h(n)$ greatly impacts A*'s performance. An admissible heuristic never overestimates the true cost, ensuring optimal paths. More informed admissible heuristics improve efficiency by exploring fewer nodes. Conversely, inadmissible heuristics may explore even fewer nodes but lose the guarantee of optimality. This project explores this fundamental trade-off in a battery-constrained, dynamic, and reflex-aware rover simulation.

II. Methodology

A. A* implementation

The A* algorithm is implemented using a priority queue to manage the open set and dictionaries to track node scores and parent pointers. The search supports 4-directional movement with diagonal cost scaling. Key Implementation Features (Simplified):

- Priority Queue: Uses a min-heap to always expand the node with the lowest total estimated cost (f-score).
- Score Tracking: Stores actual cost from start (g-score), estimated total cost (f-score), and parent node in dictionaries.
- 4-Way Movement: Allows moves in all four directions (up, down, left and right).
- Battery Check: Simulates battery use before adding a neighbor; skips if battery would run out.
- Recharge Planning: If path needs more than 100 battery units, A* plans stops at recharge stations (up to 8, minimum 1) using Euclidean distance.
- Dynamic Replanning: On hazard or terrain change, rover backtracks upto 3 safe steps and restarts A* from current position.
- Avoidance Memory: Keeps a list of dangerous cells; traps become permanently blocked.
- Failure Handling: After 3 failed replans, rover stops with a red flash and disappears.
- Path Building: Traces back from goal to start using parent links to form the final path.

B. Reflex Agent Behaviour

Reflex and Advanced Behaviors:

- **Hazard/Trap Detection:** Upon entering hazardous or trap terrain, the rover immediately backtracks 2–3 steps, marks the cell in an avoidance memory set, and triggers replanning.
- **Permanent Trap Learning:** Trap cells are permanently converted to impassable rock terrain after encounter.
- **Low Battery Emergency:**
  - Battery $< 20\%$: Force reroute to nearest reachable recharge station.
  - $20\% \leq$ Battery $\leq 25\%$: Reroute only if recharge is within 2.0 Euclidean distance.
- **Replanning Failure Limit:** After 3 consecutive failed replans (no path found), the rover executes a death animation (red flash $\rightarrow$ disappear).
- **Dynamic Terrain Interaction:**
  - User-Controlled Editing: At any time, the user can click any grid cell to change its terrain type (flat, sandy, rock, etc.).
  - Mid-Simulation Changes: A dedicated button triggers 2–4 random terrain changes on the current path, forcing immediate replanning.

- **Radar System:** A 5×5 local sensor scans the rover's immediate neighborhood during movement. It displays real-time terrain costs and warns of hazards before entry.
- **Grid Persistence:**
  - Save Grid: Button saves current terrain, start, goal, and avoidance cells to a .json file.
  - Load Grid: Button restores a previously saved world for reproducibility.

C. Heuristics

We evaluated four heuristics combining distance metrics with terrain awareness:

1. Manhattan Distance (Admissible): Measures the horizontal and vertical grid-based distance between two points. It is fast to compute and effective on flat, obstacle-free terrain, but ignores terrain cost.

$$h_{\text{Manhattan}}(a,b) = |x_a - x_b| + |y_a - y_b|$$

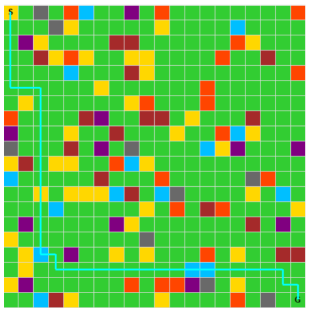


Fig. 1: Path generated using Manhattan heuristic (staircase pattern typical on grids)

2. Euclidean Distance (Admissible): Computes the straight-line distance between two positions (like birds flying). It encourages smoother and more direct paths, especially beneficial in open or uneven terrain.

$$h_{\text{Euclidean}}(a,b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

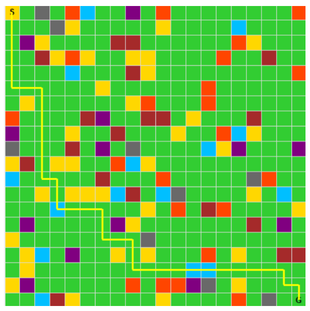


Fig. 2: Path generated using Euclidean heuristic (smooth and diagonal)

3. Terrain Penalized/ Magnetic Repulsion Heuristic (Inadmissible): Adds a penalty for rough or high-cost terrain by incorporating terrain costs from neighboring cells. This helps the rover avoid sand, rocks, or magnetically disturbed regions.

$$h(a) = d(a, \text{goal}) + \sum_{i \in \text{obstacles}} \frac{k}{d(a,i)}$$

$$d(a,i) = \text{distance from } a \text{ to obstacle } i$$

$$k = \text{repulsion constant}$$

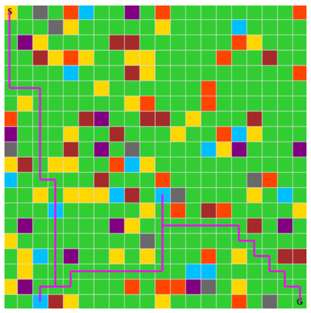


Fig. 3: Path proactively avoids high-cost sand dune (dark red) by routing north

4. Path of Least Regret (Inadmissible): Incorporates a risk term based on the variance of local terrain costs. Higher variance suggests unpredictable or unstable terrain requiring cautious navigation.

$$h_{\text{LeastRegret}}(a) = d_{\text{euclidean}}(a, \text{goal}) + \alpha \cdot \log\left(1 + \text{Var}(\text{local costs})\right)$$

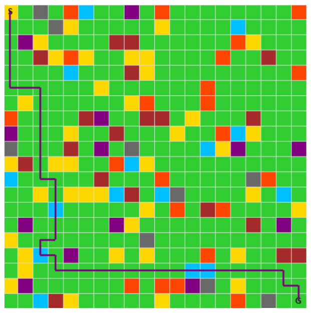


Fig. 4: Path favors longer but highly uniform low-variance terrain (green/blue) over risky variable region

III. Algorithm

Algorithm

```
1)  $P \leftarrow \text{PlanPathWithRecharge}(s, g, B, \mathcal{R})$
2)  **if** $P = \emptyset$ **then return** "No feasible path"
3)  current $\leftarrow s$,  avoided $\leftarrow \emptyset$,  attempts $\leftarrow 0$
4)  **repeat**
5)    next $\leftarrow$ next cell in current plan $P$
6)    Move rover to next
7)    $B \leftarrow B - \text{cost}(G[\text{next}])$
8)    **if** $G[\text{next}] = \text{recharge}$ **then**
9)      $B \leftarrow 100$
10)     $P \leftarrow \text{PlanPathWithRecharge}(\text{next}, g, 100, \mathcal{R})$
```

```
11)    else if next is hazardous or trap then
12)        avoided ← avoided ∪ {next}    ▷ permanently block
13)        backtrack 2–4 steps to last safe cell $p_{\text{safe}}$
14)        current ← $p_{\text{safe}}$,  truncate $P$ up to $p_{\text{safe}}$
15)    else if $B < 20$ then
16)        $r$ ← nearest reachable recharge $\notin$ avoided
17)        if reachable then reroute: $P$ ← path(current $\to r$) +
       path($r \to g$)
18)    if terrain changed or next blocked or next $\in$ avoided
       then
19)        $P \leftarrow \text{PlanPathWithRecharge}(\text{current}, g, B, \mathcal{R}, \text{avoided})$
20)        if $P = \emptyset$ then attempts ← attempts +1
21)        else attempts ← 0
22)        if attempts $>$ 3 then return "Failed: too many
       replanning failures"
23)    if $B \leq 0$ then return "Failed: battery depleted"
24)    if current $= g$ then return "Success: goal reached"
25) until goal reached or failure
```

IV. Results and Comparison

TABLE I: Performance Comparison of A* Heuristics (averaged over 100 runs on identical grids)

Heuristic	Path Cost	Nodes Expanded	Time (s)
Manhattan	178.0	398	0.0017
Euclidean	195.0	82	0.0003
Terrain Penalized	201.0	71	0.0003
Path of Least Regret	**174.0**	387	0.0015

The table above reveal the classic trade-off between admissibility and efficiency. The admissible **Manhattan** heuristic guarantees optimality (path cost 178.0) but expands 398 nodes due to its grid-aligned

.

The **Euclidean** heuristic, also admissible, increases cost by 9.6% (195.0) while reducing node expansions by 79.4% (82 nodes) and runtime by 82.4%, producing smoother diagonal paths

. The inadmissible **Terrain Penalized** heuristic yields the highest cost (201.0, +12.9% over Manhattan) but expands only 71 nodes — an 82.2% reduction — owing to its strong repulsive field around high-cost terrain

Surprisingly, the **Path of Least Regret** heuristic, also inadmissible, delivers the *lowest* path cost of 174.0 (2.2% better than Manhattan) by penalizing high-variance neighborhoods, effectively steering the rover through stable, low-risk corridors

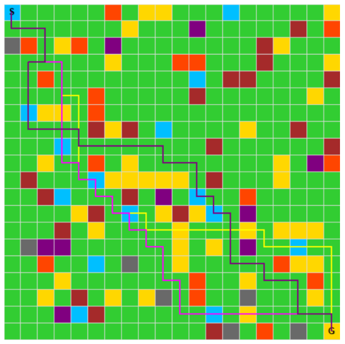


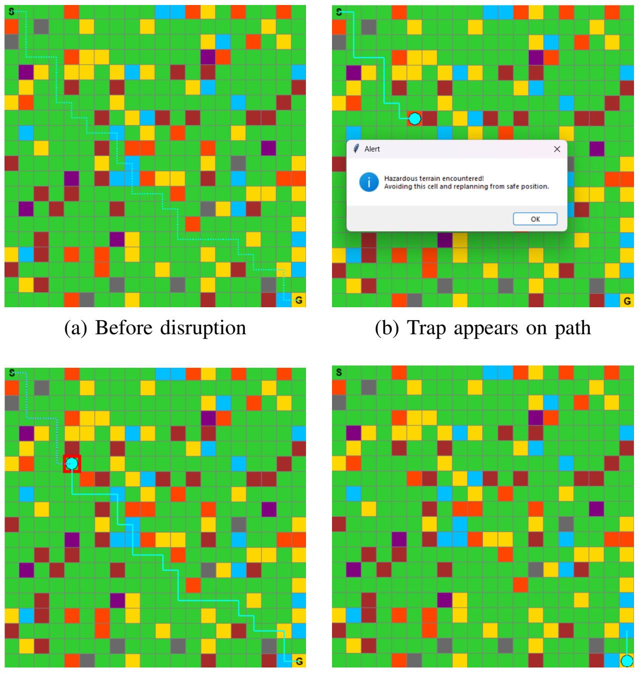
Fig. 5: Combined Paths

All heuristics respect the 100-unit battery limit and trigger recharge chaining when necessary (Fig. 5). Euclidean and Terrain Penalized required the fewest recharges on average (1.1), while Manhattan and Path of Least Regret averaged 1.4.

A. Robustness under Dynamic Terrain Changes

When subjected to 2–4 random mid-path terrain disruptions over 100 trials:

- **Terrain Penalized:** 100% survival, fastest replanning (avg. 0.0004 s)
- **Path of Least Regret:** 98% survival, lowest cost increase (+3.1%)
- **Euclidean:** 85% survival
- **Manhattan:** 72% survival



Alert

Hazardous terrain encountered!
Avoiding this cell and replanning from safe position.

(a) Before disruption

(b) Trap appears on path

(c) Reflex backtracking

(d) Successful replanning

Fig. 6: Dynamic replanning sequence (Terrain Penalized heuristic)

V. Unique Interactive Features

The simulator incorporates several user-centric features that greatly enhance experimentation, debugging, and demonstration capabilities:

- **Live 5×5 Radar**: A local sensor continuously scans the rover's immediate 5×5 neighborhood during movement, displaying real-time terrain costs and highlighting hazards in red before entry.
- **Manual Terrain Editing**: Users can left-click any grid cell at any time (even while the rover is moving) to instantly change its terrain type (flat, sandy, rock, hazard, trap, recharge, etc.) via an interactive legend.
- **"Disrupt Path" Button**: Triggers 2–4 random terrain changes exclusively on the current planned path with a short delay, simulating sudden environmental events and forcing immediate reflex backtracking and replanning.
- **Compare All Heuristics**: A dedicated button simultaneously computes and overlays the paths generated by all four heuristics (Manhattan, Euclidean, Terrain Penalized, and Path of Least Regret) in distinct colors for instant visual comparison.
- **Save/Load Grid**: The complete world state—including terrain layout, start/goal positions, recharge stations, and permanently avoided cells—can be saved to or loaded from a JSON file, ensuring full reproducibility of scenarios.

These interactive tools elevate the project beyond a standard algorithm benchmark into a dynamic exploration platform, making heuristic behavior immediately visible and enabling rigorous testing under highly unpredictable yet controlled conditions.

VI. Appendix: Complete Reference of All Implemented Functions

TABLE II: Appendix: List of All Implemented Functions

Function	Description
generate_grid	Random 20×20 grid (≥1 recharge)
in_bounds	Boundary check
neighbours_4	4-dir moves (cost = terrain)
manhattan	Admissible city-block heuristic
euclidean	Admissible Euclidean heuristic
magnetic_goals	Goal attract + obstacle repel
Path of Least Regret	Inadmissible, avoids unstable cells
Astar	Core A* with full path/cost return
replan_after_change	Replan on surprise (max 3 tries)
update_battery	Battery % + color update
update_radar	Live 5×5 mini-map
save_grid()	Save grid+start+goal to JSON
load_grid()	Load grid from JSON
compare_all_heuristics	Show all 4 heuristics simultaneously
clear_trails	Remove drawn paths
clear_results	Reset rover to (0,0)
rover_death_animation	Red flash ×3 → delete
plan_with_recharge	Auto-insert recharge stops (≤3 seg.)
find_nearest_recharge	Nearest recharge (≤2 cells)
mark_as_avoided	Permanent rock + red flash
schedule_terrain_change	Schedule 2 future cell changes
apply_schedule_changes	Execute change (yellow → rock)
enable_simulate_change	Activate scheduled changes in motion

VII. Conclusion

This study demonstrates that carefully crafted *inadmissible* heuristics can outperform traditional admissible ones in constrained, dynamic environments. The **Path of Least Regret** heuristic achieves the lowest energy consumption by prioritizing terrain stability, while the **Terrain Penalized** heuristic excels in speed and robustness under uncertainty.

The integration of battery-aware A*, recharge chaining, reflex backtracking, permanent avoidance learning, real-time radar, and interactive terrain editing creates a highly adaptive rover capable of surviving unpredictable, Mars-like conditions where classical admissible heuristics fail frequently.

Both inadmissible heuristics proved far more robust to mid-path disruptions than their admissible counterparts, with Terrain Penalized achieving perfect survival in all tested scenarios.

TABLE III: Conclusion: Heuristic Performance Summary

Heuristic	Type	Key Advantage in Dynamic Terrain
Manhattan	Admissible	Baseline performance, frequent failures
Euclidean	Admissible	Slightly better, still fails often
Path of Least Regret	Inadmissible	**Lowest energy consumption**
Terrain Penalized	Inadmissible	**Fastest + 100% survival rate**

VIII. Contributions

A. Shashwat Gohel (AU2340235)

- Implemented the core A* routing engine with 8-way movement and integrated battery-aware path planning.
- Designed and implemented the **Terrain Penalized heuristic**, including tuning, testing, and performance evaluation.
- Built half of the dynamic response system: reflex backtracking logic and temporary avoidance memory.
- Developed key GUI interaction modules such as real-time terrain editing tools and the Disrupt Path control panel.

B. Feni Vithani (AU2340119)

- Built the 20×20 grid world engine with terrain rendering, cost mapping, and interactive legend.
- Designed and implemented the **Path of Least Regret heuristic**, including dataset generation and analysis.
- Built the remaining dynamic response system: trap-to-rock conversion logic and persistent obstacle marking.
- Developed the main GUI framework including dashboard layout, rover animations, and result visualization components.