Indian Institute of Engineering Science & Technology, Shibpur
Department of Computer Science & Technology
8th Semester Artificial Intelligence Laboratory 2025
CS 4271

1. A disaster has struck a **futuristic smart city**, and your task is to develop an **AI-powered drone navigation system** to **rescue stranded survivors**. The drone must **autonomously navigate through the city**, avoiding obstacles like collapsed buildings and fire zones while optimizing for the fastest route (equivalent to minimum energy consumption).

   Your objective is to implement the *A Search Algorithm** to guide the drone from its **starting position (S)** to **rescue multiple survivors (G1, G2, ... , Gn) (intermediate states) and return to base (B) (treated as goal state)** while minimizing energy consumption and ensuring safe traversal.

Grid Representation of the City (State Space Representation of the City):

The smart city is represented as a **3D grid** where each $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ coordinate defined as: (x, y) in the 2D coordinate and z values is the depth of flying at different levels. Each of the depth 'z' is represented by different notations as:

- **Roads & Open Spaces** (`0`) → The drone can fly here.
- **Buildings/Collapsed Structures** (`1`) → The drone **cannot** pass through.
- **Fire Zones** (`F`) → High-risk zones. Passing through **incurs an extra cost**.
- **Survivor Locations** $(G_1, G_2, \ldots, G_n)$ → The drone must **visit and rescue them**.
- **Recharging Stations** (`R`) → The drone can **stop and recharge energy** if needed.
- **Base Station** (`B`) → The drone must **return here after completing the mission**.
- **Drone's Start Position** (`S`) → Where the drone begins.

Drone Movement and Cost Considerations:

a) **The drone moves in 3D space**:
   - **Up** `(x, y, z+1)`, **Down** `(x, y, z-1)`,
   - **North** `(x-1, y, z)`, **South** `(x+1, y, z)`,
   - **East** `(x, y+1, z)`, **West** `(x, y-1, z)`.
   - **Diagonal movements** (like flying at an angle) **are not allowed** for simplicity.

b) **Energy Consumption Factor:**
   - Moving through **clear space (0)** costs **1 energy unit**.
   - Moving through **fire zones (F)** costs **3 energy units** due to turbulence.
   - Moving **upwards (z+1) costs extra 2 energy units**, while descending `(z-1)` costs **1 energy unit**.

c) **Objective:**
   - The drone **must rescue all survivors** before returning to base `(B)`.
   - The **optimal path minimizes total energy consumption**.

- If needed, the drone can **recharge at (R)**, but stopping at a station adds a **fixed time penalty**.

Example Grid Input (5×5×3):

```
Layer 0 (Ground Level):
S  0  1  0  0
0  F  1  1  G1
F  0  0  0  F
R  1  1  1  0
B  0  1  0  G2

Layer 1 (Mid Level):
0  0  1  0  0
0  1  1  0  0
F  0  0  0  1
0  1  1  1  0
0  0  0  0  0

Layer 2 (Sky Level):
1  0  0  0  0
0  F  0  1  0
0  0  1  0  F
0  0  0  0  0
0  1  0  0  1
```

Optimal Path Using A*:

1. Start at (0,0,0) → "S"

2. Move to (0,1,0) → Free space (1 energy)

3. Move to (0,1,1) → Climb to Mid Level (2 energy)

4. Move to (0,2,1) → Free space (1 energy)

5. Move to (1,2,1) → Free space (1 energy)

6. Move to (1,3,1) → Free space (1 energy)

7. Move to (1,4,1) → Free space (1 energy)

8. Move to (1,4,0) → Descend to Ground Level, Rescue G1 ✅ (1 energy)

9. Move to (2,4,0) → Free space (1 energy)

10. Move to (3,4,0) → Free space (1 energy)

11. Move to (3,3,1) → Climb to Mid Level (2 energy)

12. Move to (3,2,2) → Climb to Sky Level (2 energy)

13. Move to (4,2,2) → Free space (1 energy)

14. Move to (4,3,2) → Free space (1 energy)

15. Move to (4,4,2) → Free space (1 energy)

16. Move to (4,4,1) → Descend to Mid Level (1 energy)

17. Move to (4,4,0) → Descend to Ground Level, Rescue G2 ✅ (1 energy)

18. Move to (4,3,0) → Free space (1 energy)

19. Move to (4,2,0) → Free space (1 energy)

20. Move to (3,1,0) → Free space (1 energy)

21. Move to (3,0,0) → Recharge at (R) ✅ (1 energy)

22. Move to (3,0,1) → Climb to Mid Level (2 energy)

23. Move to (3,1,1) → Free space (1 energy)

24. Move to (3,2,1) → Free space (1 energy)

25. Move to (3,3,1) → Free space (1 energy)

26. Move to (3,3,0) → Descend to Ground Level (1 energy)

27. Move to (4,2,0) → Free space (1 energy)

28. Move to (4,1,0) → Free space (1 energy)

29. Move to (4,0,0) → Base B ✅ (1 energy)

Energy Consumption Calculation:

- Moving through free space: 12 × 1 energy = 12 units

- Moving through fire zones: 0 (avoided!)

- Climbing (z+1): 3 × 2 energy = 6 units

- Descending (z-1): 4 × 1 energy = 4 units

- Using recharging station (R): No energy used, but incurs time penalty.

- Total Energy Used: 23 units

Final Summary:

Optimal Path: (0,0,0) → (0,1,0) → (0,1,1) → (0,2,1) → (1,2,1) → (1,3,1) → (1,4,1) → (1,4,0) → (2,4,0) → (3,4,0) → (3,3,1) → (3,2,2) → (4,2,2) → (4,3,2) → (4,4,2) → (4,4,1) → (4,4,0) → (4,3,0) → (4,2,0) → (3,1,0) → (3,0,0) (Recharge) → (3,0,1) → (3,1,1) → (3,2,1) → (3,3,1) → (3,3,0) → (4,2,0) → (4,1,0) → (4,0,0)

Total Energy Consumption: **23 units**

Rescue Status: **G1 ✅, G2 ✅**

Recharge Status: **Recharged at (3,0,0)** ✅

Mission Completion: **Success! The drone returned to base B!** ✅

Implementation Guidelines:

a) *Use A search algorithm** to compute the most energy-efficient path.
b) **Priority Queue (Min-Heap)** should be used to store nodes with the lowest `f(n) = g(n) + h(n)`.
c) **Heuristic (h(n))**: Use a **3D Euclidean distance** to estimate cost to the goal:

$$h(n) = \sqrt{\left(x_{current} - x_{goal}\right)^2 + \left(y_{current} - y_{goal}\right)^2 + \left(z_{current} - z_{goal}\right)^2}$$

d) **Account for terrain cost penalties** (e.g., fire zones, vertical movement, etc.).
e) **Recharging Logic**: If energy is too low to reach the next goal, navigate to the nearest `(R)` station.
f) **Ensure the algorithm supports multi-goal search**, as the drone must **visit multiple survivors** before returning.
g) **Handle Edge Cases**:
   - No valid path exists to all survivors.
   - Energy depletion before reaching a recharge station.
   - The base `(B)` is unreachable due to obstacles.

Bonus Challenges:

a) **Dynamic Obstacles:** Fire zones `(F)` randomly spread after a few moves. Can the drone **replan** in real-time?
b) **Wind Effect Simulation:** Introduce **random wind disturbances** that alter movement cost dynamically.
c) **Dijkstra vs. A***: Implement both **Dijkstra's Algorithm** and **A*** for comparison.
d) **Multiple Drones Collaboration:** If two drones are available, how should they **coordinate to rescue faster**?