Project 1 Report

Netids: SS4024, Ktp60

1. Explain the design and algorithm for your Strategy 3, being as specific as possible as to what your bot is actually doing. How does the bot factor in the available information to make more informed decisions about what to do next?
    - We designed Strategy 3 to bias exploration toward the center of the ship. Our idea was that the center is probably likely to connect to multiple open paths and reduce the chance of chasing isolated dead ends too early. This approach helped the bot focus on high access areas first, rather than random edge cells. We implemented this by:
        - At every step, identifying all unknown cells.
        - Picking a target closest to the center using Manhattan distance
        - Using A* to plan a path to the target.
        - If it bumps into a wall or hits a blocked cell, it updates its map and replans.

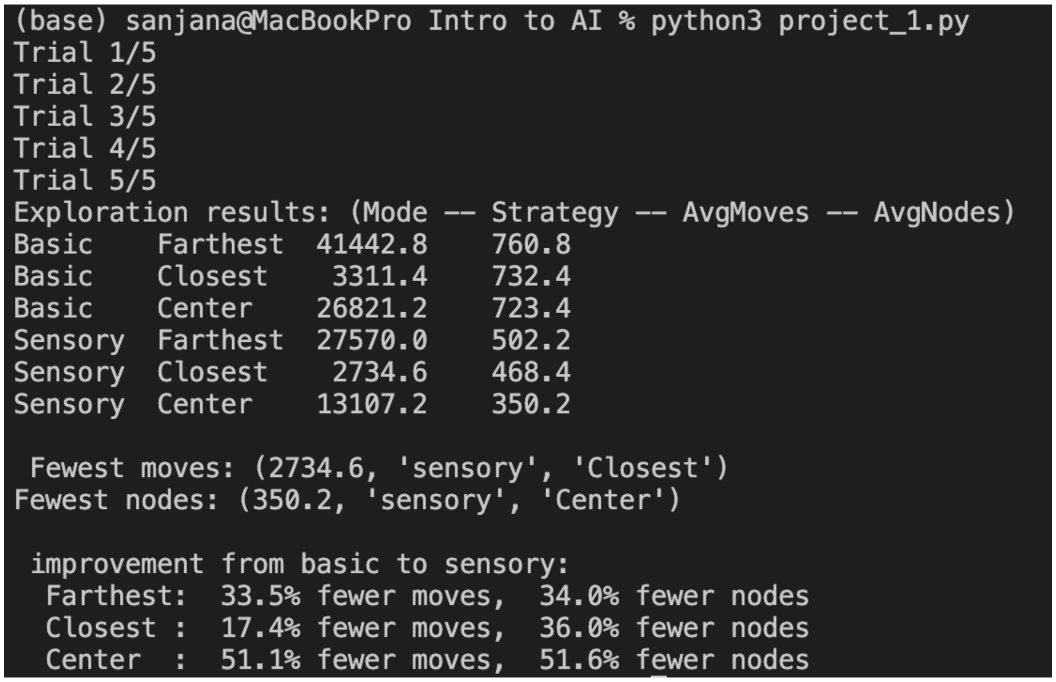
        ```
        return abs(a[0] - b[0]) + abs(a[1] - b[1]) #manhattan distance
        dist_to_center = self.bot._heuristic(cell, (self.bot.ship.D // 2,
        self.bot.ship.D // 2))
        return min(unk, key=frontier_score) #pick target cell closest to
        center based on manhattan dist
        ```

        - How a target is chosen:
            - unk is the list of all currently unknown but reachable cells.
            - Frontier score gives each of those cells a score. Lower score is better:
                - unknown_neighbors → cells with more unknown neighbors get a smaller score, so they're preferred.
                - + 0.5 × wall_neighbors → slight penalty if a cell looks like a dead end.
                - dist_to_center → tiebreaker that nudges the bot toward the center.
            - Min walks through unk, applies frontier_score to each cell, and returns the cell with the smallest score. That returned cell is the actual target for the next A* plan.

2. For each bot and each strategy, I am interested in the following statistics to compare them. How many movements did it take to identify every cell in the ship? How many

nodes were processed in planning algorithms as part of identifying every cell in the ship? (Note, there may be a lot of replanning for any of the bots and any of the strategies!)

- We ran 5 trials for each combination of bot and strategy, using a ship size = 50x50 and a dead end reduction factor of 0.5.

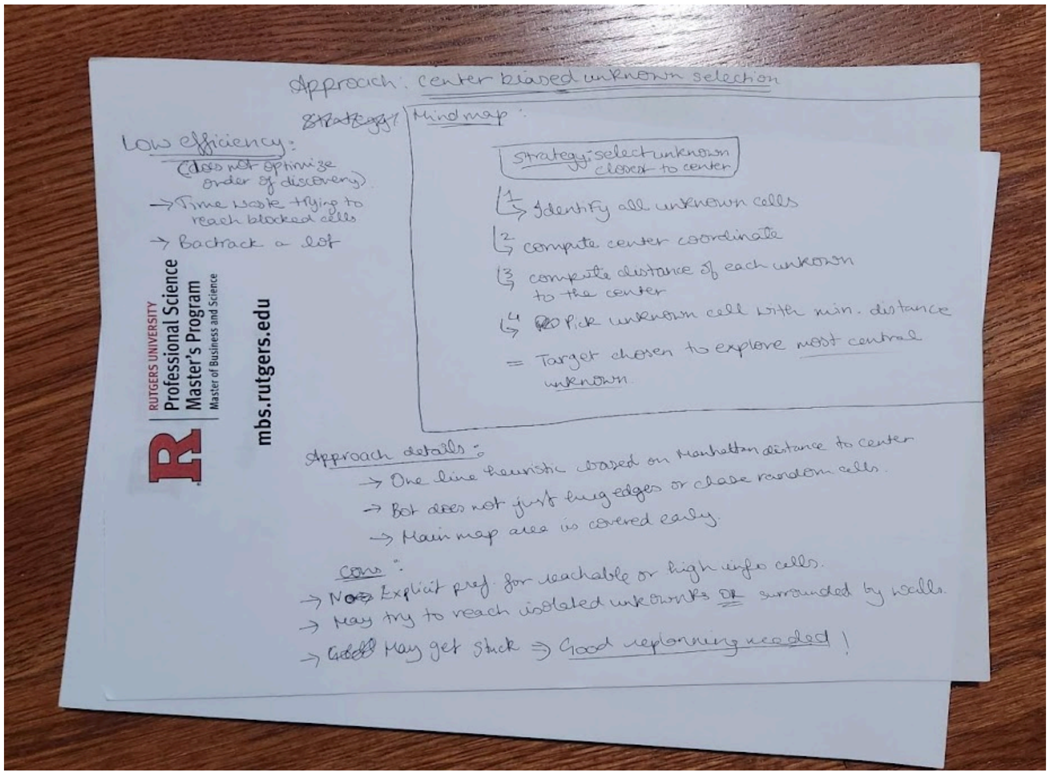


```
(base) sanjana@MacBookPro Intro to AI % python3 project_1.py
Trial 1/5
Trial 2/5
Trial 3/5
Trial 4/5
Trial 5/5
Exploration results: (Mode -- Strategy -- AvgMoves -- AvgNodes)
Basic    Farthest  41442.8    760.8
Basic    Closest    3311.4    732.4
Basic    Center    26821.2    723.4
Sensory  Farthest  27570.0    502.2
Sensory  Closest    2734.6    468.4
Sensory  Center    13107.2    350.2

 Fewest moves: (2734.6, 'sensory', 'Closest')
Fewest nodes: (350.2, 'sensory', 'Center')

 improvement from basic to sensory:
  Farthest:  33.5% fewer moves,   34.0% fewer nodes
  Closest :  17.4% fewer moves,   36.0% fewer nodes
  Center  :  51.1% fewer moves,   51.6% fewer nodes
```

  - The farthest strategy turned out to be the slowest overall. It led to the highest move counts and node processing, especially for the basic bot. This happened because it kept chasing distant unexplored areas, which often required long paths and frequent replanning.
  - Sensory + Closest was the most efficient in moves.
  - Sensory + Center was the most efficient in planning nodes.
  - Farthest strategy was the least efficient for both bots.

- 

Approach: center biased unknown selection

~~Strategy~~ Mind map:

Strategy: select unknown closest to center

1 → Identify all unknown cells

2 → compute center coordinate

3 → compute distance of each unknown to the center

4 → Pick unknown cell with min. distance

= Target chosen to explore most central unknown

Low efficiency:
(does not optimize order of discovery)
→ Time waste trying to reach blocked cells
→ Backtrack a lot

RUTGERS UNIVERSITY
Professional Science Master's Program
Master of Business and Science
mbs.rutgers.edu

Approach details:
→ One line heuristic based on Manhattan distance to center
→ Bot does not just hug edges or chase random cells.
→ Main map area is covered early.

Cons:
→ No Explicit pref. for reachable or high info cells.
→ May try to reach isolated unknowns OR surrounded by walls.
→ May get stuck ⇒ Good replanning needed!

Figure: Center strategy mind map

3. Speculate on how you might construct the ideal strategy. What information would it use, what information would it compute, and how? Do not worry too much about runtime.
    - The ideal strategy might combine multiple heuristics. It would balance between choosing nearby targets and those that help reveal large unknown regions. For example, it could score each unknown cell based on how many unexplored neighbors it has, how far it is, and whether going there opens up new areas. It could also remember past decisions to avoid repeating paths that didn't lead to much discovery. This way, the bot can make smarter moves over time, focusing on high-reward areas instead of just following the closest unknown blindly.

4. What would have to change, and how, if there were a partial sensory bot, that could only sense the total number of blocked cells surrounding its current location? (But could still identify blocked cells by bumping into them.) Be thorough and specific. How would you expect its performance to compare to the other two?
    - We would need to change the bot's sensing logic so that it only stores the number of blocked cells around it, not their exact positions. This means the bot would know how many walls are nearby, but not where they are. This makes planning harder. The bot would need to explore more carefuilly and might bump into walls

more often to figure out the layout. It would have less information than the sensory bot but more than the basic one. So, its performance would likely be between the basic and sensory bots. It could do better than the basic bot in some cases but would not be as efficient as the full sensory bot.

Acknowledgements

- Krish Patel implemented the basic logic (including bot and ship) of the code, strategy 1, and strategy 2.
- Sanjana Suresh implemented strategy 3, added all necessary comments for the code, wrote this report and integrated and refined the code for submission.