Ian Wilhite

Aero 489 – Autonomous Systems

2/5/2025

Preliminary (10 points)

1) As a search problem, the state must contain the rovers location, the action must include rover traversal in all directions (and account for edge collision), and the search problem must also track the path cost in addition to the state. The world must contain cells with danger and a goal (science).
2) Because the environment is known, static, and fully observable, our rover begins the problem knowing all necessary information to search, the information it knows cannot change, and the information it knows is complete. Because it has all information it needs, and that information cannot change, it is not necessary to sense.

Basic Search (20 points)

1) Not all algorithms find the optimal solution: Depth-First Search (DFS) does not *always* find the optimal solution, because DFS will pick one potential path from its options and follow it to completion, and will return the first feasible solution it finds, which may or may not be the optimal solution, while other algorithms prioritize less costly solutions. BFS is less efficient than UCS because it must search all parallel paths but significantly outperforms DFS because it is more likely to find a *more* efficient solution in sweeping its options laterally. A* is the most efficient solution because it is the only informed search of the group and can rely on the Manhattan heuristic to approximate the goodness of a cell towards the optimal solution. For random problems, not all methods find the “correct”/optimal solution.
    a. Optimal Solution:

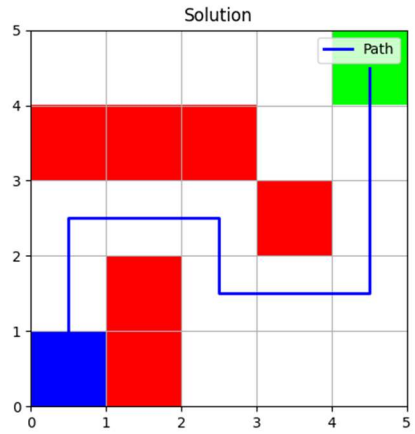

Fig. 1, sample problem with provided solution

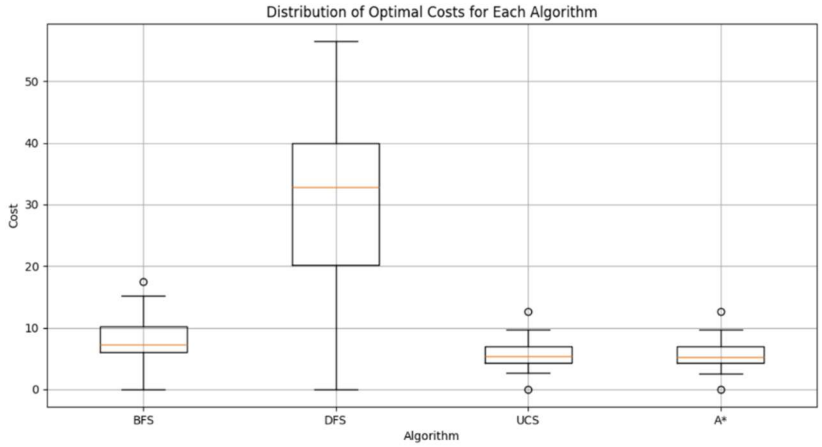

Fig. 2, Costs found by each algorithm

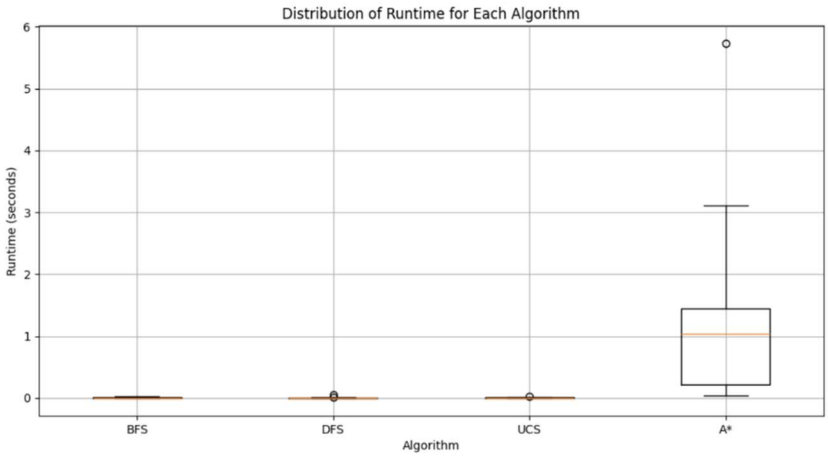

Fig. 3, Runtimes taken by each algorithm

2) In cost performance, the DFS did not find the optimal solution due to its tendency to burrow into an incorrect solution and return a local, suboptimal, solution. A* had a much larger runtime to find the solution. This means that the optimal search to use for this situation would be UCS.

```
Average Cost and Runtime for Each Algorithm:
BFS: Average Cost = 22.25, Average Runtime = 0.0032 seconds
DFS: Average Cost = 319.75, Average Runtime = 0.0032 seconds
UCS: Average Cost = 22.25, Average Runtime = 0.0035 seconds
A*: Average Cost = 22.25, Average Runtime = 0.7053 seconds
```

Fig. 4, Code output of search algorithms without Danger

3) By allowing the search to enter dangerous cells with a penalty, many of the less efficient algorithms average goes up by the increase of possible-but-less-optimal solutions, while A* can take the most advantage of the new opportunity and is the only algorithm to decrease its average cost. All algorithms required more runtime (about double) to search the larger pool of possible solutions.

```
Average Cost and Runtime for Each Algorithm:
BFS: Average Cost = 34.14, Average Runtime = 0.0073 seconds
DFS: Average Cost = 667.95, Average Runtime = 0.0039 seconds
UCS: Average Cost = 22.52, Average Runtime = 0.0104 seconds
A*: Average Cost = 21.29, Average Runtime = 1.3169 seconds
```

Fig. 5, Code output of search algorithms with Danger

Part 2:

Non-Deterministic World (30 points)

1) Changes:
    a. Creating transition_with_failure function that only moves the rover 75% of the time, and leaves the rover in its current state 25% of the time.
    b. Creating a transition_with_danger_and_failure function that is identical to the transition_with_danger except it calls the transition_with_failure for the base transition function then adds danger to the cost.
2) I believe that the solution will look similar to the optimal solution found by A*, only with nested conditions such that each step will fall into a conditional while loop where it will perform each action sequentially until the path is complete.

3) Yes, the algorithm completes the search, but no, it does not come close to other deterministic methods in terms of performance. It is not reasonable to compare

these, however, because the and-or search cannot assume that each function will work, it is extending additional effort to branch out along each potential option, and is also searching the cases in which the rover does not move, however this causes additional computation as it is simply in the same situation it previously was. It is worth noting that the And-Or search implemented returned solutions similar to those generated with a traditional Depth-First-Search.

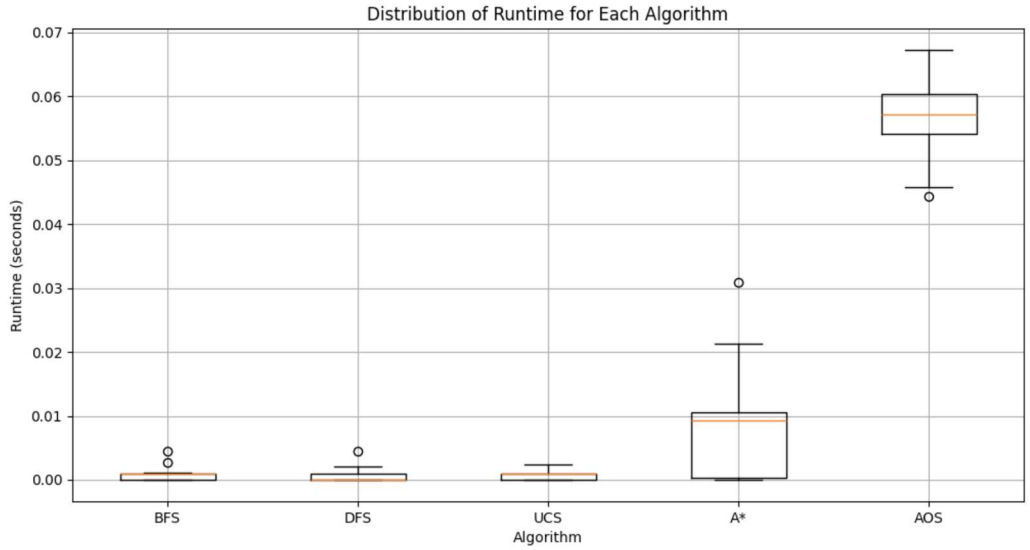

Fig. 6, Runtimes with Danger including AOS

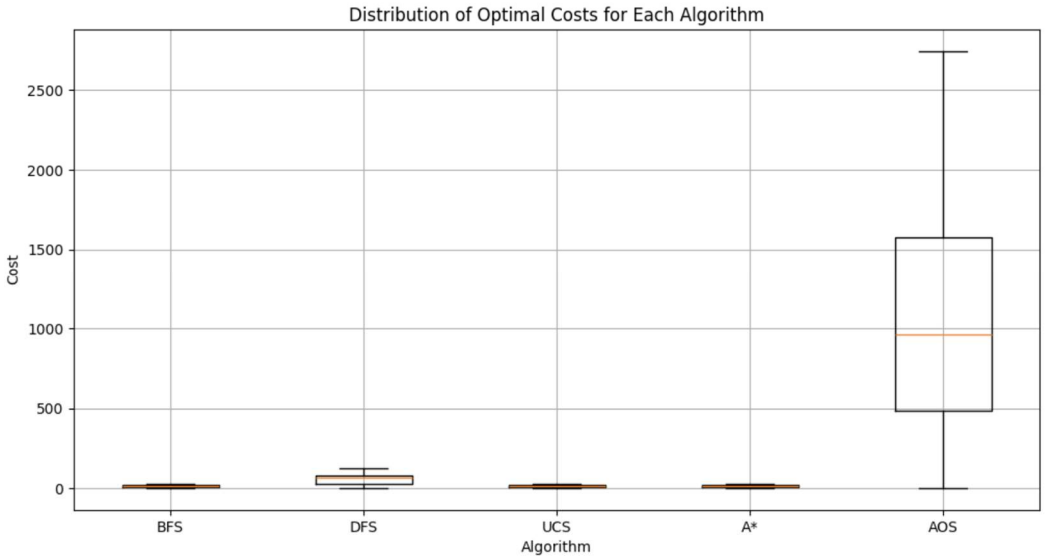

Fig. 7, Costs with Danger including AOS

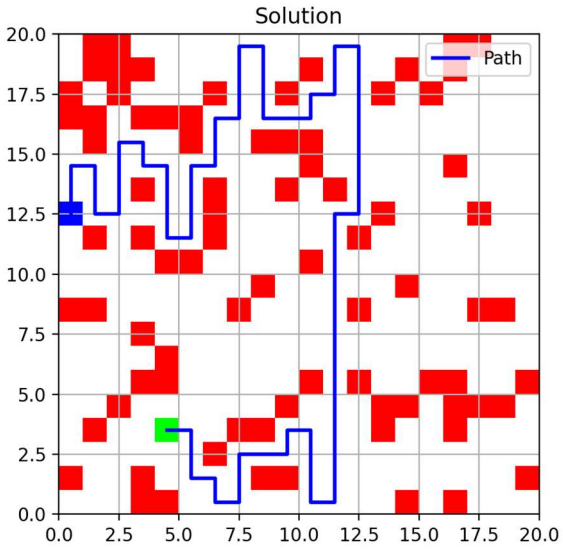

Fig. 8, 1st Sample AOS search solution

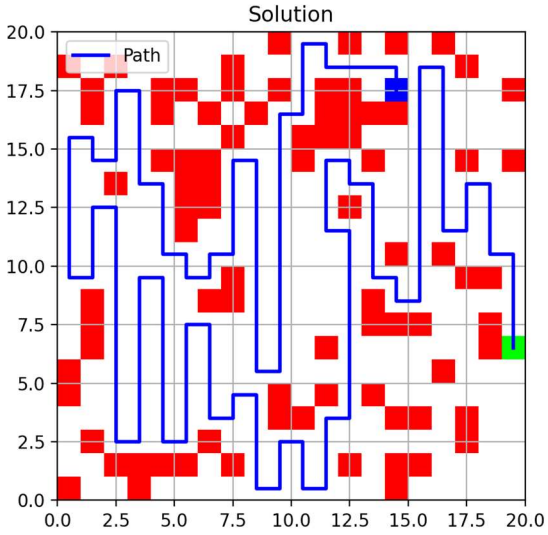

Fig. 9, 2nd Sample AOS search solution

4) Plots provided along with a sample of the path generated from the function.

```
Average Cost and Runtime for Each Algorithm:
BFS: Average Cost = 14.06, Average Runtime = 0.0009 seconds
DFS: Average Cost = 76.88, Average Runtime = 0.0005 seconds
UCS: Average Cost = 9.58, Average Runtime = 0.0012 seconds
A*: Average Cost = 9.29, Average Runtime = 0.0083 seconds
AOS: Average Cost = 164.13, Average Runtime = 0.0010 seconds
```

Fig. 10, Code output with Danger including AOS

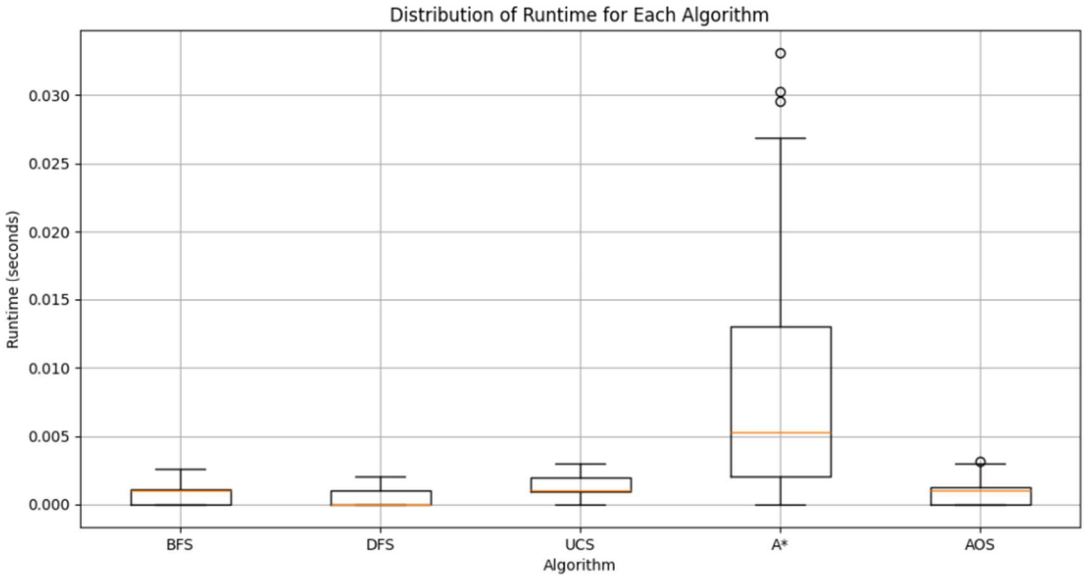

Fig. 11, Runtimes with Danger including AOS

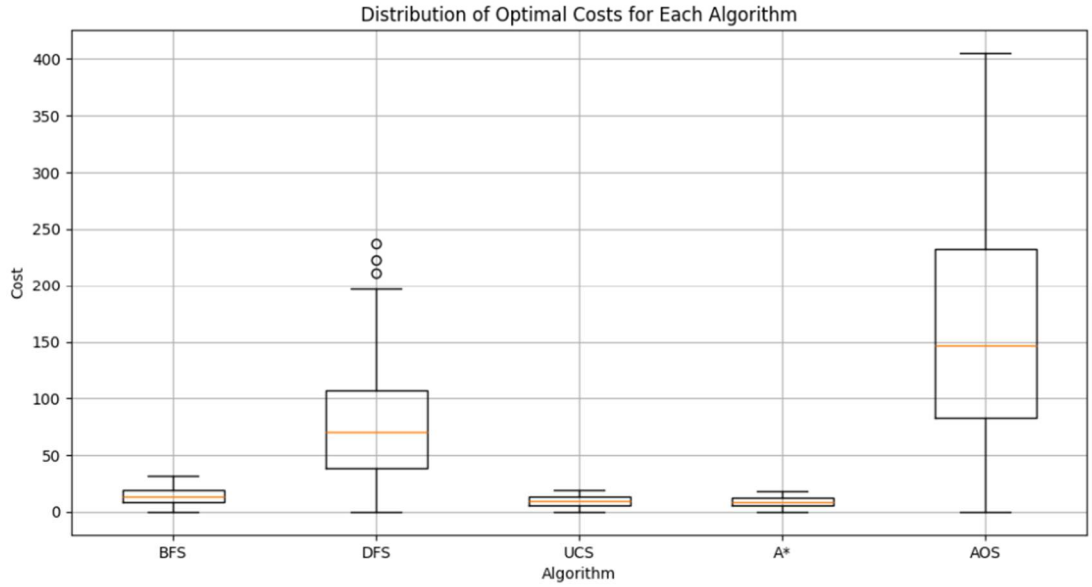

Fig. 12, Costs with Danger including AOS

Unknown World (10 points + 10 bonus)

1) To search the world without knowing the location of the alpha cell, a new heuristic would need to be developed for A*, involving the closeness using the spectrometer, given that the conditions for the search heuristic can be met. The noise must not break the admissibility of the heuristic, which a sensor would likely not so long as the majority of the noise comes from offset and proportional error, and that very little comes from randomness error.