CS599-HPC
Brian Donnelly & Felicity Escarzaga
04/07/21

Preliminary Report

1. Overview

The project, described succinctly as, “optimal navigation for a robotic arm,” has thus far been completed according to schedule. The schedule listed below in section 5, **Table 1,** shows that as of April 7th, three objectives should be accomplished including, creating listed code structures, implementation of a brute force solution, and implementation of a search agent.

Two of the three objectives were semi-completed much earlier than anticipated since they were implemented while programming the preliminary benchmark. The code structures for the World class and Actuator class were created using a customized vector class that contains the methods for vector math and vector rotations. The brute force approach, while not yet parallized gives the validation angles for moving a two-actuator arm to the destination point. Visualization for the robotic arm moving to the destination point is shown in *fig. 1*.

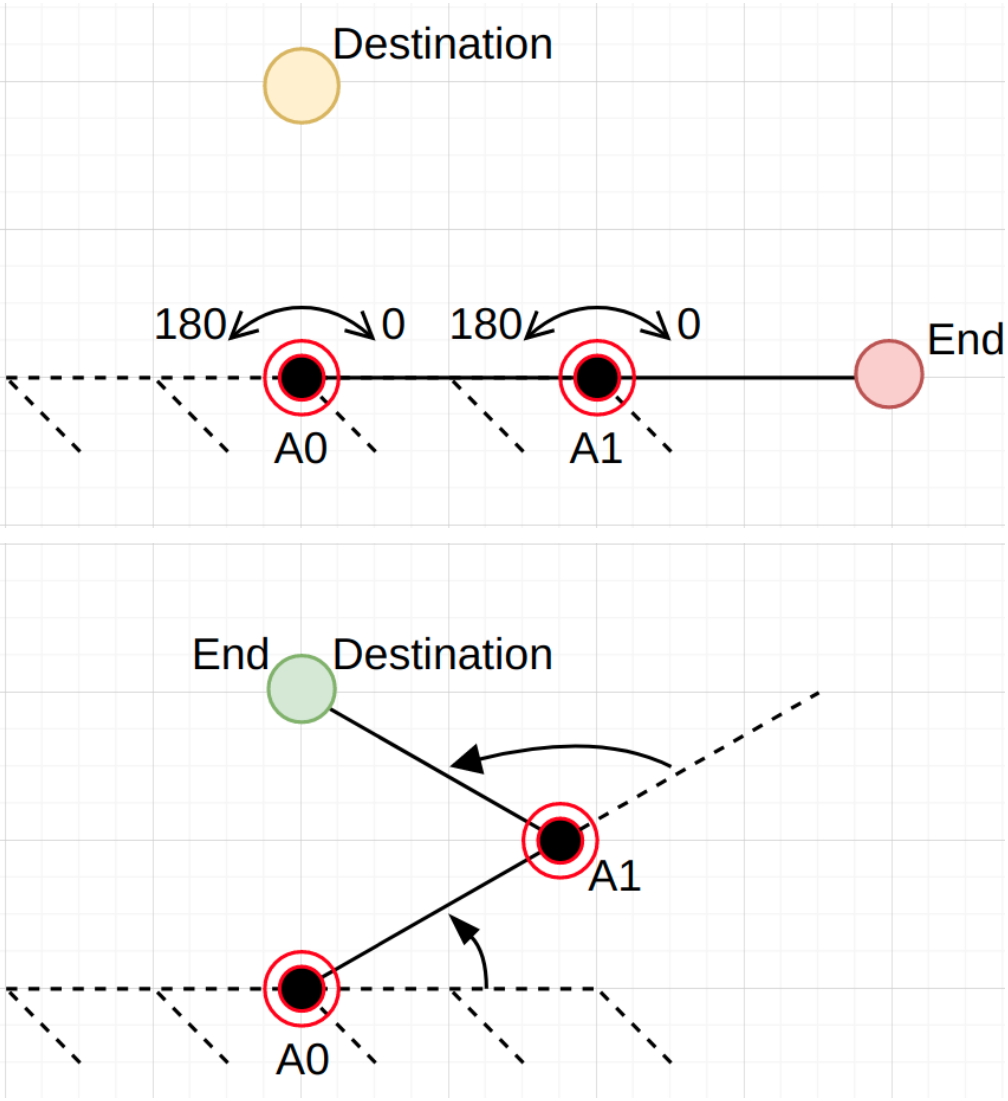

Fig 1. Top: Starting orientation of robotic arm; Bottom: Arm with endpoint at destination location

Since the first two objectives were completed ahead of schedule, some time was spent developing a graphical representation which was listed as a stretch goal in the project proposal. However, as will be discussed in section 2. Challenges, this was dropped as the deadline for the next objective became imminent.

The final objective scheduled to be completed before the preliminary report was the implementation of a search agent. Several search algorithms were theoretically compared based on their optimization and efficiency. A* was determined to be the best search algorithm for this application. The search agent has been implemented; including the additional Node class required for the graph structure A* must traverse. This puts the project on course for the scheduled completion time.

2. Challenges

One of the major challenges we addressed was how to represent the search space. Because we want to design for N actuators, we need a representation that allows our search algorithm to evaluate all of its options at each step. To do this we came up with a graph structure that has nodes which represent world states (i.e. angles and location of all actuators) and connected nodes with edges that represent the possible movements from that world state. We had to implement a graph structure that can be created and discarded easily. The number of possible world states is too much to have a static graph with nodes for each possibility. The node structure we created allows the graph to be generated around each node as a search agent travels through the graph. This was a challenge that we failed to anticipate at the beginning of the project. The current solution does work and has allowed the search agent to rapidly traverse the graph with it generating in front of it and degenerating behind it.

One challenge that we attempted and failed at was to create a graphical representation with OpenGL. We thought that having some way to visualize the search would help us verify the results, but we were unable to overcome certain obstacles. Our current code base leans heavily on object oriented programming, something that is not supported with OpenGL. After a considerable amount of effort trying to get it to work we decided to scrap that branch of development and either forgo a visualizer or implement a different solution (with a different graphics api) after other core work was completed.

3. Remaining Objectives

The remaining objectives include parallelization of both the brute force and search algorithms, testing and performance evaluations of both algorithms, and completing the final report write-up. Parallelization of the approaches will be done using openmp; and testing will be conducted for an increased number of actuators and threads. The final result will show whether the search algorithm has better performance and/or scalability compared to the brute force. The write-up will be done in Overleaf to structure the report as a peer-reviewed paper.

4. Preliminary Results

We have implemented an A* search agent (though it could still use some optimizations) which outperforms the brute force search by ~100 times on our simple test case. The major gain from the search agent was that we reduced the number of evaluated states from 3,240,000 down to

2,700. While there is overhead with the search agent, it will be able to scale to larger problems because of the way it searches through the space. In the simple case with just two actuators, we were able to get the time down to a very usable speed. Below is a readout of the test case with both the brute force and the agent search. While the brute force takes ~2 seconds, the agent search takes ~0.025 seconds to find the exact same solution.

```
Building the world...Complete
Performing test move:
Actuator Locations:
(0.000000, 0.000000, 0.000000)
(0.707107, 0.000000, 0.707107)
Actuator angles: 45.000000, 0.000000,
EndPoint: (1.414214,0.000000,1.414214)
Actuator Locations:
(0.000000, 0.000000, 0.000000)
(0.707107, 0.000000, 0.707107)
Actuator angles: 45.000000, 45.000000,
EndPoint: (0.707107,0.000000,1.707107)

Iterating through 3240000 states...Complete!
Time to brute search: 2.071600

Lowest score was 0.000000 with angles: 30.000000, 120.000000,
Actuator Locations:
(0.000000, 0.000000, 0.000000)
(0.866025, 0.000000, 0.500000)
EndPoint: (-0.000000,0.000000,1.000000)


Running agent code

Visited 2700 nodes
Solution found with time: 0.024916
Actuator locations:
(0.000000, 0.000000, 0.000000)
(0.866025, 0.000000, 0.500000)
Actuator orientations:
(0.000000, -1.000000, 0.000000)
(0.000000, -1.000000, 0.000000)
Actuator angles:
( 30.000000,  120.000000, )
```

5. Checklist and Discussion

As stated above, the objectives have been on-time with minor alterations. The listed code structure has been completed with additional structure included as needed. The brute force approach has been implemented, but not yet parallelized. A search agent has also been implemented, though alpha-beta search was not used as was predicted in the project proposal. A*, being used instead, required a graph structure for the search algorithm to traverse. The final objective, to parallized the search, is on track to be completed by it's April 23rd deadline, with the brute force expected to be parallized by the 14th.

Table 1

Legend
✔ Complete
~ In progress
✘ Removed

- ✔ 1. Create a code structure to handle: (by March 24th)
  - ✔ a. An Actuator Class
    - ✔ i. Class extensions for actuator types
  - ✔ b. A World Class
    - ✔ i. Includes the start and end locations
- ✔ 2. Implement a ~~parallel~~ brute force approach (by March 31st)
  - ✔ a. This will be used for verification
- ✔ 3. Implement a search agent (by April 7th)
  - ✘ ~~a. This will most likely be an alpha-beta search though we still need to evaluate a few more options~~
  - ✔ b. Decided on A* search
    - ✔ i. Requires graph structure
    - ✔ ii. Requires algorithm to traverse graph
- ~ 4. Parallelize brute force approach (by April 14th)
- ~ 5. Parallelize the search agent with either OMP for a single node or MPI for multi-node systems. (by April 23rd)
  - ~ a. What we use here depends on the runtime of the search agent. Ideally we would be able to run on a single node, and only use OMP.