S01547146 Chung-Yu (Vincent Chang) Project 1

1. Q1: Both the simple way and the hard way contain State Space Setup (boundary), Space Information (State Valid Checker), Problem Definition (start/goal state, x coordinate setting), a route planner and a solver.

   Q2: The simple way create a simple setup object in State Space and use a default planner with default parameters to solve the problem, while the hard way tend to instantiate by its own, such as the space information instance, the problem definition instance, and the RRT planner instance to solve the problem.

2. Q1: The idea of Control Space is being taken into consideration. Instead of randomly completing two points in the State Space and checking whether the route works, we define the Control Space (cbound and grid decomposition) and the propagate function (x, y, yaw), and integrate over small steps to get next state with State Propagator.

   Q2: On average the one with control runs faster (0.0579s vs 0.8798s). I think the one with control decomposes the State Space into smaller grids to guide the search, which can eliminate those illegal routes faster, while the simple setting has to deal with more possible routes in the whole State Space.

3. I ran 7 problems with all 4 planners, the ur5 robot, and the default parameters. For each pair of them, I ran 3 indices (same between pairs), 2 trials (7*4*3*2=168), using successful rate in 24 trials than time as the difficulty of problems.
   Task difficulty tier in descending:
   cage (8.3%, 14.46s) > bookshelf tall (29.16%, 4.20s) >
   bookshelf thin (37.5%, 2.25s) > table pick (37.5%, 1.09s) >
   bookshelf small (54.1%, 1.67s) > table under pick (62.5%, 3.43s) >
   box (70.8%, 8.99s)

4. I used the successful rate of each 4 planners than their performance under harder task.

   Performance in descending:

   RRTC (78.5%) > KPIECE (47.6%) > PRM (35.7%) > RRT (9.52%)

   Configuration:

   a. RRTC can still perform well if in extreme limited planning time

   b. KPIECE can do better if we extend the planning time.

   c. PRM somehow strike a balance between the efficiency and successful rate.

   d. I didn't find apparent advantages of RRT under different configuration…

5. a.

   I chose table pick problem for PRM to test the value of max_nn = 8, 20(def), 40, 60, 80 and 100.

   The successful rates are 0%, 0% (def), 66%, 16.67%, 16.67%, and 50%.

   I guess when the max_nn increases, the planner needs more computational resources to build a denser road map and have higher chance to reach the goal while also the time limit if we keep the time limit in 30s.

   b.

   I chose cage problem for RRT, RRT-Connect and KPIECE to test the range in 0.5(def), 5, 10, 15, 20 and 25

   For RRT-Connect, The successful rates are 33%, 33%, 16.67%, 33%, 0%, 16.67% and the rest of the planner remains 0% rates in all test cases.

   I think this case is just too hard to find a route.

   C

   I focus on cage problem for RRT-Connect with range = 15 and test the value of goal bias = 0.01(def) 0.1 0.25 0.5 0.75 0.95.

   The successful rates are 33%, 0%, 33%, 16.67%, 0%, 16.67%

   Although I still can't recognize the sweet spot of this parameter, but I personally would give goal bias = 0.01~0.1 and 0.1~0.25 more trials if we aim to find the best result.

6. Note: all of the value refers to “in terms of median value in distribution”

   a.

   For Home, I think RRTConnect with range = 50 performed the best since it achieved almost 100% solved solution and spent the fewest time compared to RRTCorrect with range = 5 and 25. As for other planners, they performed worse in time, graph states and solution length.

   b.

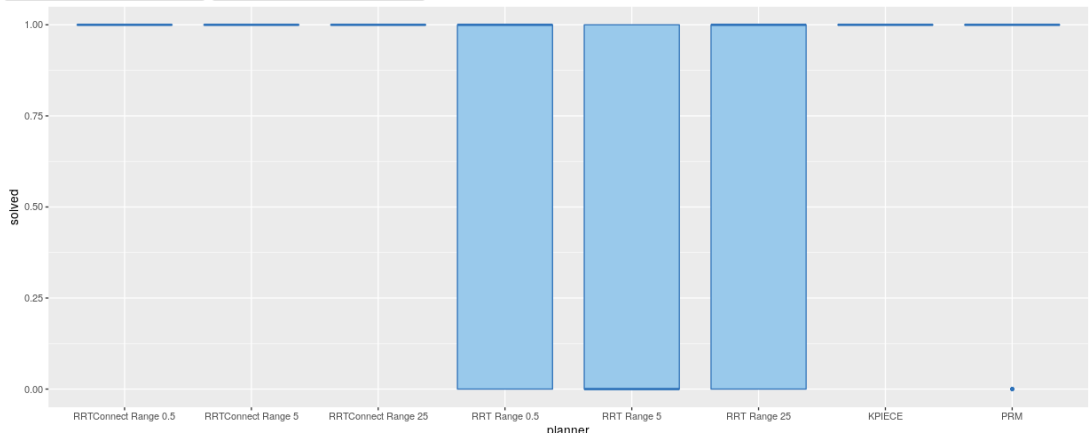
   For Twistcool, I think RRTConnect with range = 25 performed the best since it spent the fewest time and graph states to find shortest length and remains almost 100% solved rate. As for other planners, they performed worse in time and graph states

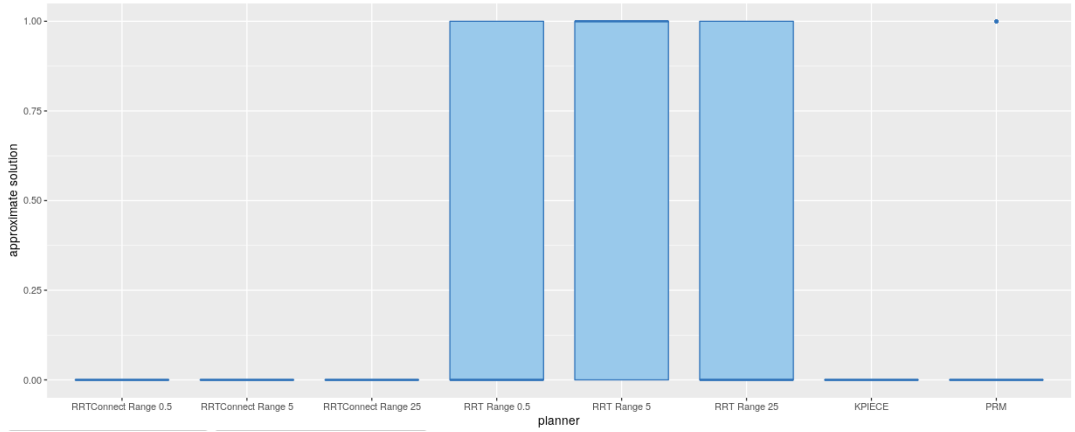
   c.

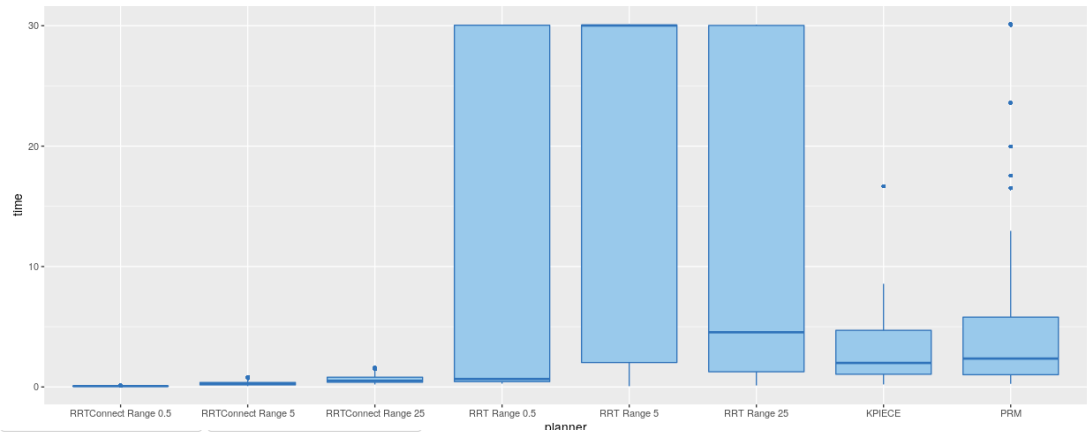
   For Abstract, I think both RRTConnect with range = 25 and RRT with range = 25 performed well. They both achieved almost 100% solved rate, the connect version spent less time (compared to others) finding paths though. As for other planners, they performed worse in time and graph states

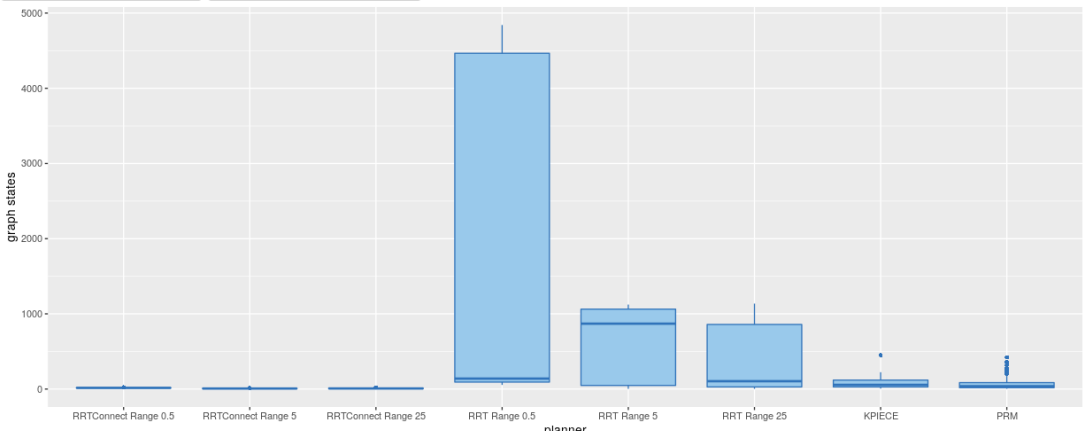
   d.

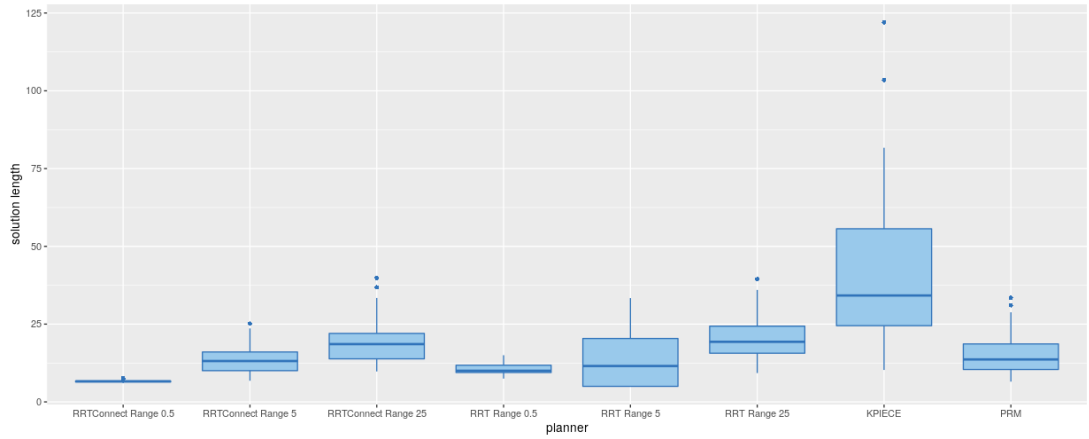
   For ur5 with bookshelf small, I think the RRTConnect with range = 0.5 performed the best since it got shorter solution length most of the time than RRTConnect with other ranges. The rest of the metrics are similar to RRTConnect with other ranges. As for other planners, they have higher time and solution length compared to RRTConnect with range = 0.5.




approximate solution
1.00
0.75
0.50
0.25
0.00
RRTConnect Range 0.5
RRTConnect Range 5
RRTConnect Range 25
RRT Range 0.5
RRT Range 5
RRT Range 25
KPIECE
PRM
planner

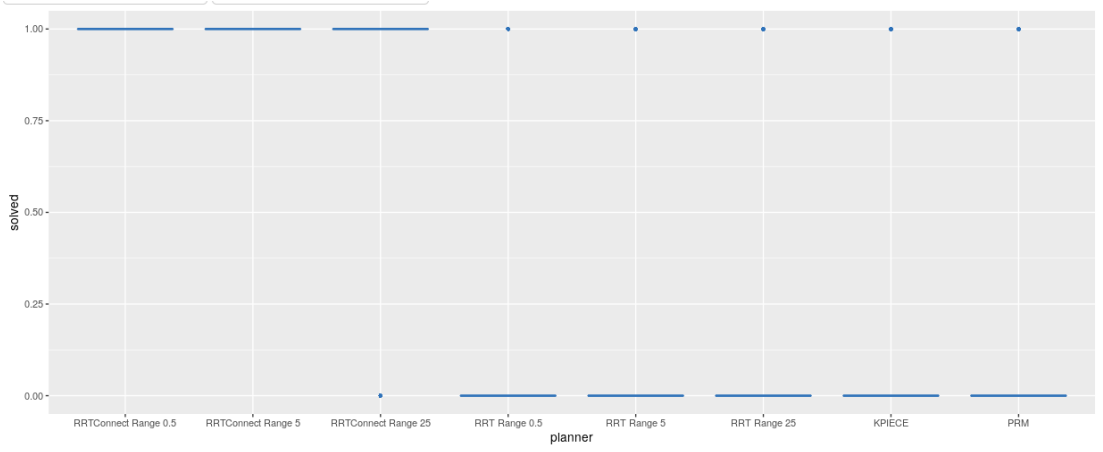

time
30
20
10
0
RRTConnect Range 0.5
RRTConnect Range 5
RRTConnect Range 25
RRT Range 0.5
RRT Range 5
RRT Range 25
KPIECE
PRM
planner

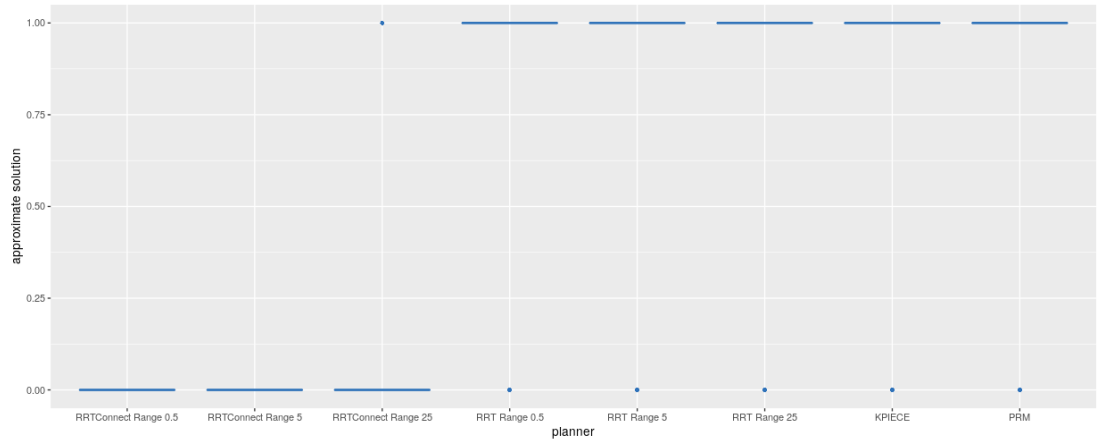

graph states
5000
4000
3000
2000
1000
0
RRTConnect Range 0.5
RRTConnect Range 5
RRTConnect Range 25
RRT Range 0.5
RRT Range 5
RRT Range 25
KPIECE
PRM
planner

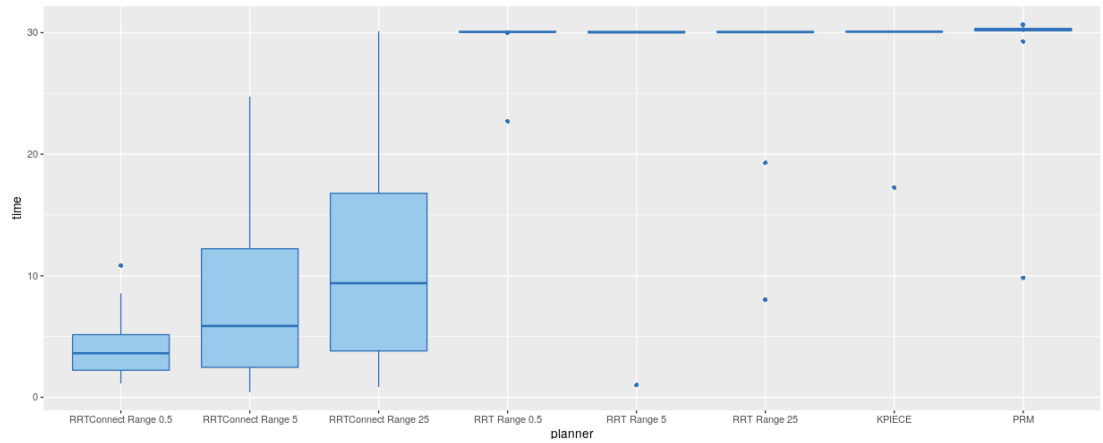

solution length
125
100
75
50
25
0
RRTConnect Range 0.5
RRTConnect Range 5
RRTConnect Range 25
RRT Range 0.5
RRT Range 5
RRT Range 25
KPIECE
PRM
planner

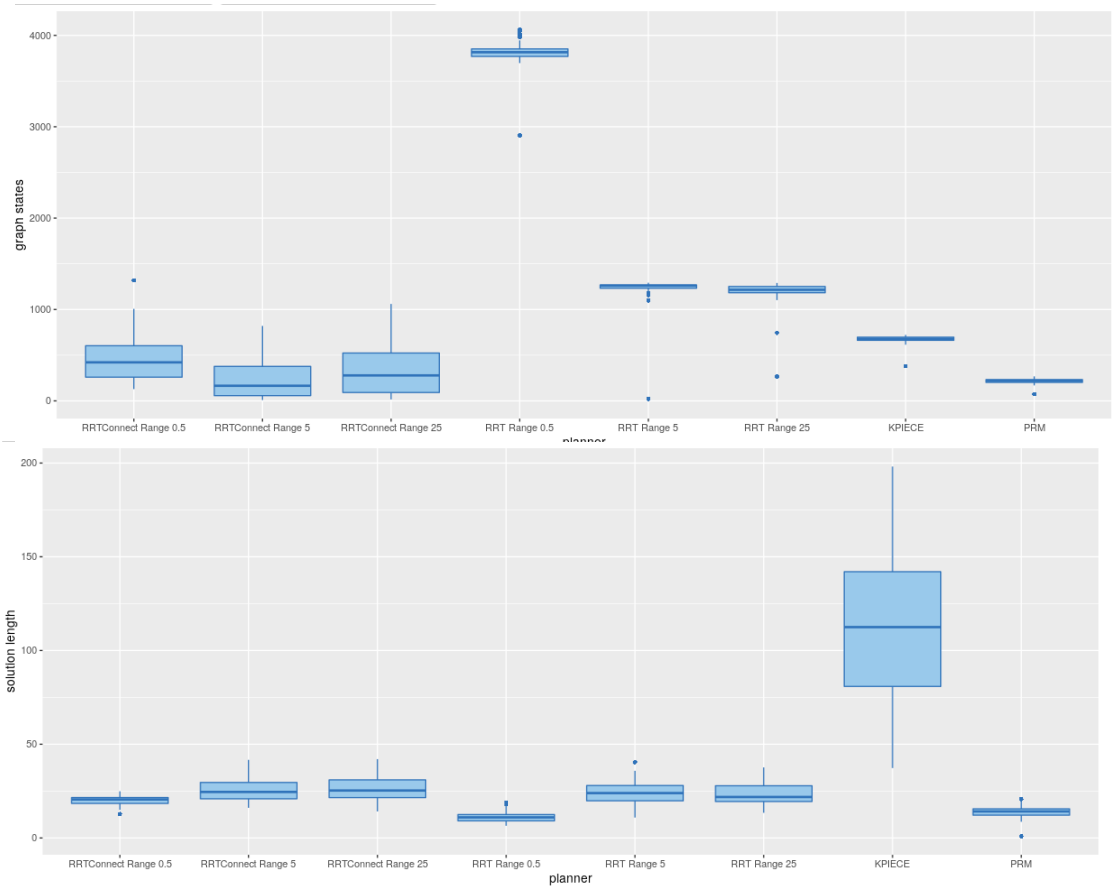
e.

For fetch with table pick, I think RRT Correct with range = 0.5 performed the best since both of its time and solution length are lower than RRT Correct with range = 5 or 25, with only slightly higher graph states. As for other planners, they can't reach over 90% solved









7. 1.1-4, 2 hours

   1-2-8, 3 hours

   2.1-4, 1 hour

   2.2-1, 10 min

   2.3-1, 1 hour,

   2.4-1, 1.5 hours,

   2.5-1, 2.5 hours,

   2.6-1, 10 min

   The most difficult part of this assignment is to understand the architecture and the element in the world. Designing and running the experiment in exercise 1-2 also costs lots of time.