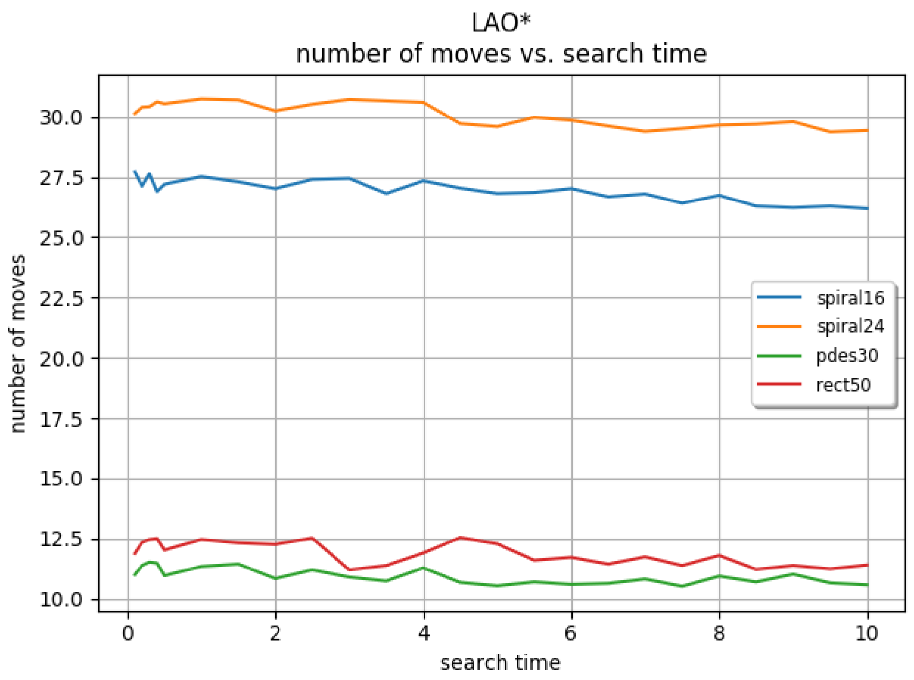
CMSC722 Project Part 2

Fan Yang

The two algorithms that I coded up for this project are LAO* (in proj2a.py) and UCT (in proj2b.py). A couple of modifications were added to both. The data computed during each run was cached to disk, to avoid repeated computation in the following runs.

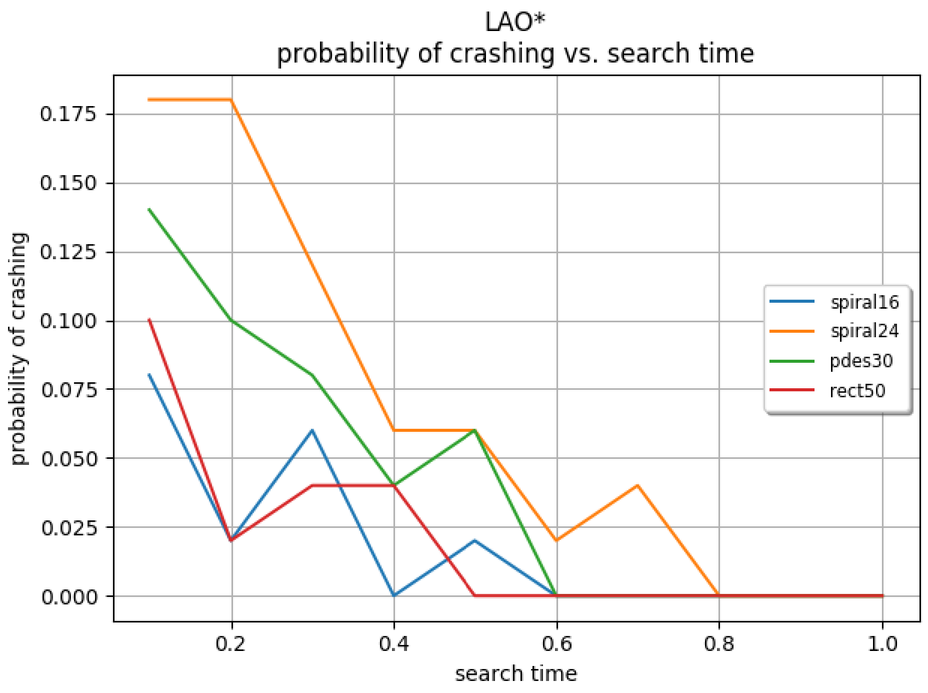
The first hypothesis that I verified is that both algorithms provide better instructions when given more search time. I selected some of the hard problems and counted the number of steps that the algorithms take to reach the finish lines as the amount of search time ranges from 0.1 second to 10 seconds. After filtering out the runs that crashed, the plot below was generated for LAO*:



It can be seen that the number of moves that the problems need did not decrease much as the search time increased. Taking the hardest problem "spiral24" for example, the average number of moves was 29.64 when given a search time of 0.1 second, and 28.94 when given 10 seconds.

This observation does not agree with my expectation that the lines in the above plot would show evident downward trends. After a second thought, I attributed this surprise to the effectiveness of h_walldist, the heuristic used in the implementation of LAO*. The explanation is that, though the algorithm cannot see far from the current location when given very limited search time, the heuristic values returned by h_walldist are informative enough so the algorithm can still choose good moves in most cases.

A second hypothesis is that LAO* would crash more frequently when given less search time. The corresponding plot is shown next page.



This time my expectation was demonstrated correct. It's observed that the hardest problem, spriral24, had the highest probability of crashing, followed by pdes30. These probabilities decreased rapidly as the search time increased, and they all fell to 0 when given a search time of 0.8 second or more.

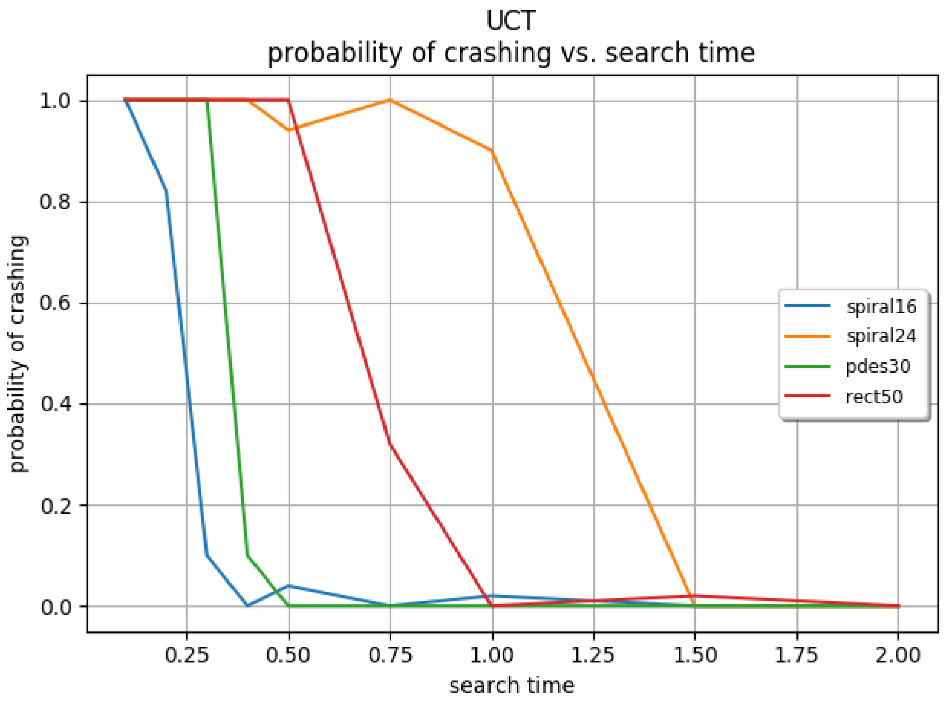
A possible explanation for the trend is that when search time is short, LAO* acts greedily because it can only see its nearby locations. As the result, the moves chosen in this situation may not be optimal and may lead to dead ends in the future.

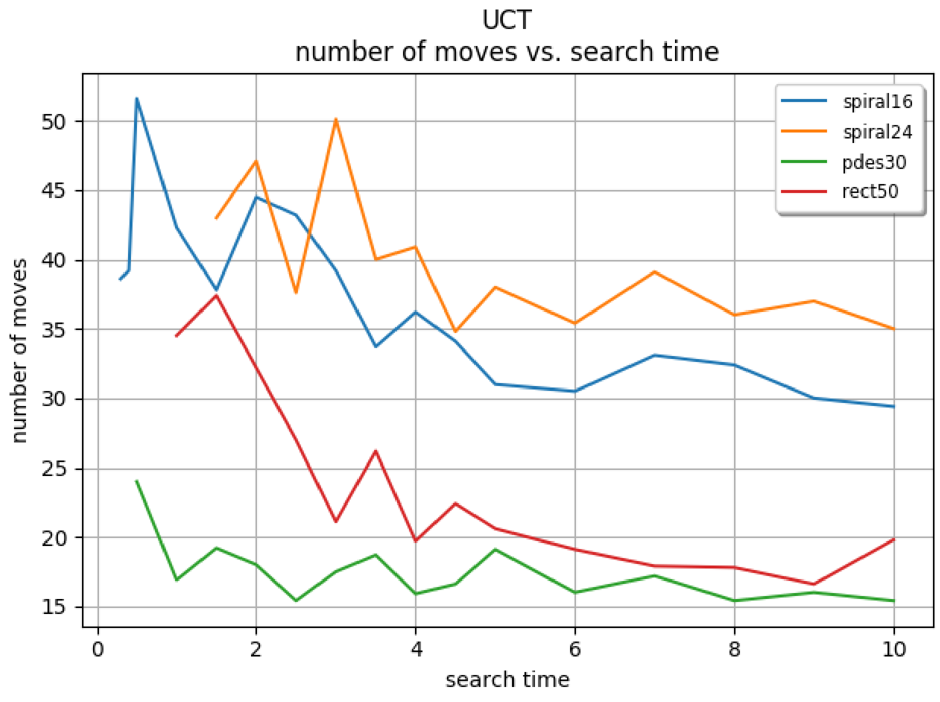
The above two hypotheses were verified against the UCT algorithm too.

It's observed that in most situations UCT did not perform nearly as well as LAO*. The probabilities of crashing were high despite of the optimizations that I tried. See next page for the plot of "probability of crashing versus search time" for UCT, and the plot of "number of move versus search time" for UCT after filtering out the runs that crashed.

If given a search time of 0.1 second, UCT crashed every time on every of the four problems. In comparison, under the same situation the probability of crashing using LAO* is only 0.18 on the most difficult problem "spiral24". It's only when the search time was increased to around 1 second that the probabilities dropped to near 0.

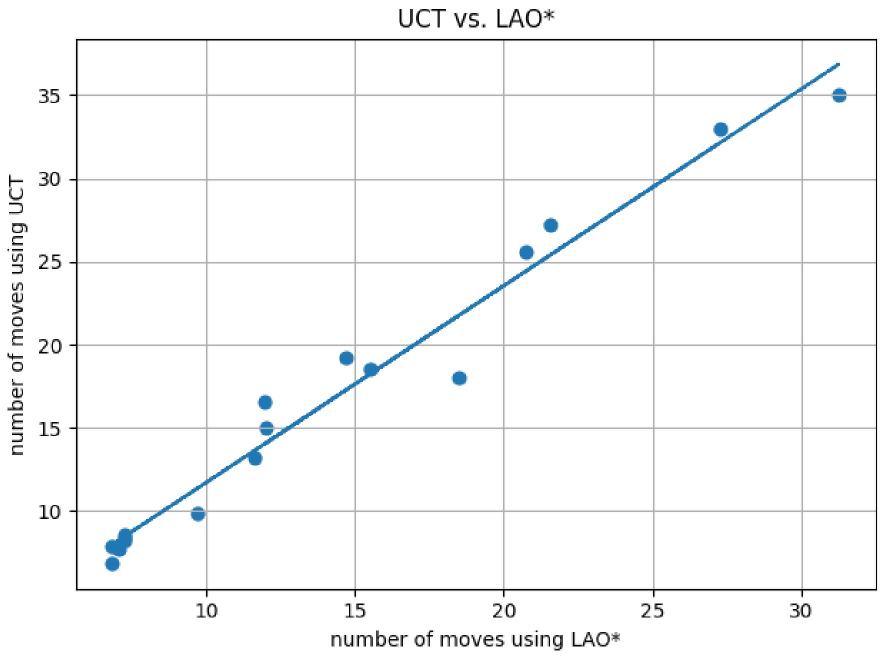
Another observation is the performance of the UCT program fluctuated considerably, even after averaged over 50 runs. This is probably because insufficient search time prevented the algorithm from finding (near) optimal moves and some randomness arose during the exploration. Indeed, it was observed that the range of the fluctuation became much smaller as the search time increased.


UCT
probability of crashing vs. search time
probability of crashing
search time
1.0
0.8
0.6
0.4
0.2
0.0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
spiral16
spiral24
pdes30
rect50


UCT
number of moves vs. search time
number of moves
search time
50
45
40
35
30
25
20
15
0
2
4
6
8
10
spiral16
spiral24
pdes30
rect50

The following scatter plot compares the efficiency of the LAO* program and the UCT program, from the view of the number of moves that they need to arrive at the finish lines when given the same amount of search time (= 5 seconds). The problems used for this plot are the 17 sample problems provided for our project 1.

We can see that the number of steps needed by the UCT program was roughly 1.2 times the number of steps needed by LAO*, averaged over all data points. The regression line fits the points quite well. It’s speculated that the good fit is due to the fact that the two algorithms have become quite stable as the search time is increased to 5 seconds.


UCT vs. LAO*
number of moves using UCT
number of moves using LAO*