Does Fast Beat Thorough?:

Comparing Real-Time Search Algorithms LSS-LRTA* and RTAA*

Shane Kochvi

February 2, 2018

1 Introduction

While there are many graph search algorithms, such as Dijkstra's algorithm and A*, which can yield optimal trajectories through a search domain given unlimited time constraints, real-life scenarios often require that an agent begin traversing a search domain before a complete path to a goal can be found. Real-time search algorithms attempt to achieve fast planning, fast execution, and minimal trajectory costs. Local Search Space - Learning Real-Time A* (LSS-LRTA*), first described by Koenig & Xiaoyun[1] is one of the most popular in a class of real-time search algorithms which iteratively interleave action executions along a path with time-limited planning, which comprises an A* lookahead and updates to the heuristic values of nodes within the local search space. Real-Time Adaptive A* (RTAA*) is a real-time heuristic search algorithm proposed by Koenig & Likhachev[2] as an alternative to LSS-LRTA* that uses a faster but less thorough strategy for updating heuristic values. Each of these algorithms can be modified for search with a partially observable state space and a fully observable state space. The goal of this research was to replicate the results presented in the paper wherein RTAA* was first described, compare the performance of my implementations of LSS-LRTA* and RTAA* in the Grid World domain, both partially observable and fully observable, and the Traffic domain, and evaluate the claims of Koenig & Likhachev (2006).

Contributions

I found several statistical errors in the work of Koenig & Likhachev. I did not find support for the notion that RTAA* outperforms LSS-LRTA* with tighter time constraints on the planning phase with a partially observable state space. In fact, RTAA* performs better with looser time constraints. I found that RTAA* performs better relative to LSS-LRTA* in grid path-finding scenarios when the domain instance contains few obstacles. This is especially apparent with a partially observable state space. I found a qualitative difference in both algorithms' behavior after the obstacle density in random maps reached 40%. I found no difference in the behavior of the algorithms in the Traffic domain.

2 Background

General Description

Both RTAA* and LSS-LRTA* proceed through two main phases during each iteration: planning and action execution. Planning consists of a lookahead phase, during which A* generates the local search space around an agent and discovers the optimal path from the source node to a frontier node, and a learning phase, during which the heuristic values of the nodes within the local search space are updated according to a specific strategy. Action execution is simply the transitioning of an agent from node to node along the local cost-minimal path discovered during A* lookahead.

Learning strategy

The main difference between the two algorithms is in the way they handle the learning phase. LSS-LRTA* uses Dijkstra's algorithm to update the heuristic values of the nodes within the local search space in an outside-in manner, starting with the nodes on the frontier of the local search space on the priority queue. The upshot is that after learning, every node within the local search space has a heuristic value equal to the cost to reach a frontier node plus the heuristic value of that frontier node such that the cost to reach a frontier node plus that frontier node's heuristic value is the minimum possible. In other words, a node's updated heuristic value reflects its best path to its "best" frontier node. RTAA* updates using a novel formula for updating the heuristics of the nodes in the local search space:

$$h[s] \leftarrow h[\bar{s}] + g[\bar{s}] - g[s]$$

where s is the node to be updated and $\bar{s}$ is the node that was about to be expanded during the A* lookahead.

LSS-LRTA*, by using Dijkstra's algorithm, updates the heuristic values of the nodes within the local search space in a more granular way than RTAA* and thus should tend to have more accurate and substantial updates, while RTAA* updates more quickly, but with less accurate and less substantial increases in heuristic values.

Tie-breaking

Following Koenig and Likhachev, I used a heap-based priority-queue for A*. Ties between nodes with identical f-values were broken in favor or nodes with greater g-values, and remaining ties were broken randomly. At first, I implemented random tie-breaking by applying an O(n) reservoir sampling algorithm to the minimum f-valued nodes of the heap, which were accessed via breadth-first search. However, I found this method too slow, especially for domains with many ties. I eventually chose to add a random integer variable to each node to be used for breaking ties during heap operations.

2.1 Experiment

Implementation

The algorithms were implemented in C++ and run on an Optiplex 960, Core2 duo E8500 3.16 GHz with 8GB RAM. Most of the code base for both algorithms was already written. I wrote the code for RTAA* and modified the code to handle partially observable state spaces. I also significantly altered the code for the Traffic domain and modified the algorithms to check for dead ends, which sometimes occur in the Traffic domain. Nodes were generated from a memory pool as they were encountered during A* and stored in a hash table.

Algorithm Metrics and Parameters

In both algorithms the A* lookahead phase can be limited by the number of expansions allowed per search episode or by the amount of time allowed per search episode.
In my experiments and in the experiments in the paper, the A* lookahead phase is expansion-limited. To infer the performance of the algorithms in a time-limited scenario, amortized planning time per search episode was calculated by dividing the planning time by the number of search episodes.

2.1.1 Grid Path-Finding

The Grid World domain models a grid with obstacles. It consists of traversable cells and obstacle cells, and each node has four potential successors, to the north, east, south, and west.

Known Terrain vs. Unknown Terrain

For grid path-finding, I used two different versions of each algorithm, one for domains in which an agent knows all obstacle states ahead of time: a scenario with "known terrain," and another, in which the agent can only learn whether a state is an obstacle during action execution: a scenario with unknown terrain. Below are the specifications for a real-time heuristic search with unknown terrain:

- Obstacle states are initially unknown to the agent

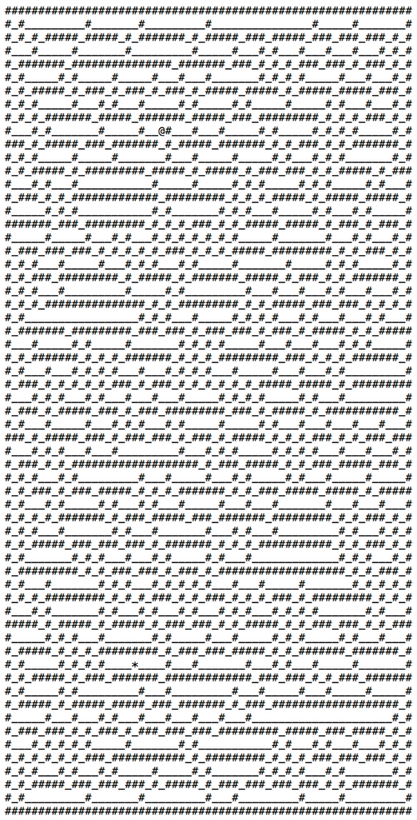


Figure 1: 30×30 maze

- The agent can only observe obstacles in immediately adjacent nodes and during A* lookahead assumes all unobserved terrain is free of obstacles (this is known as the freespace assumption)
- The agent can observe adjacent nodes as it moves through the terrain during action exuction.
- When the agent encounters an obstacle in the path of an action execution, it begins a new A* search.

Real-time heuristic search with known terrain is easier to understand than real-time heuristic search with unknown terrain. All of the paths given by A* search in the lookahead phase are free of obstacles, and searches do not restart during action execution.

The only significant implementation difference between the versions of the algorithms for known and unknown terrain is that, in unknown terrain, obstacle states were stored in a separate hash table only as they were observed during action execution.

2.1.2 Mazes

I generated 2500 mazes according to the specifications described by Koenig and Likhachev:

- The size of the mazes must be 151×151

- The mazes must be generated with a depth-first search
- The start and end points must be determined randomly

Grid World domains were represented with ASCII files. I generated the mazes by first creating an array of cells and an associated array of characters representing obstacles. Each cell was associated with a shuffled list of directions: north, east, south, or west. Cells were visited recursively. After a cell was visited, the first direction from its list was chosen such that the cell associated with that direction was neither visited nor out of range. The characters in the character array associated with the current cell and the next cell, and the obstacle character between them, were then replaced with characters representing traversable cells. If the list contained no directions with cells that could be visited, the recursion backtracked.

To ensure that the mazes adhered to the specifications, I used some ad hoc procedures:

- To ensure that the mazes were unbiased, I confirmed that the number of north, south, east and west turns during depth-first maze generation were approximately equal.
- To ensure that the start and end points were indeed chosen randomly for each maze, I confirmed that the average Manhattan distance between all start and end points for each maze was approximately 100. (Distances between x and y coordinates of the start and end points should each be on average 1/3 the total width and height of the mazes, respectively).

An example maze is shown in Figure 1.

Unknown Terrain

The experiments described by Koenig & Likhachev were peformed on mazes with unknown terrain. I will first describe my attempts to replicate their results before presenting my own and comparing them.

Replication

Once fully implemented according to the above specifications and run over all 2500 maze instances, as per Koenig & Likhachev (2006), some of my results did not match the results of the paper well. Because of differences between my experiments and the paper's authors' in the hardware used and in implementation details, it was not unexpected that time-based metrics would not accord well with the paper. (In particular, it seems likely that the authors used a 2D array to store nodes for fast access, while I used a hash table). However, large differences between some non-time-based metrics remained.

I concluded that this was because the method used to calculate all of the fractional metrics was incorrect. Specifically, it appears that for metrics that were derived by dividing one by the other, such as trajectory cost/planning time per search episode, the authors of the original paper first obtained the average of the results over all instances for the raw metric, and then divided, instead of calculating the average of the derived metrics as obtained from each instance. Additionally, in the table presented in the paper, the metric termed “standard deviation,” appears to be in reality, a statistical measure known as the “estimate of the mean standard error,” calculated as follows: $s/\sqrt{N}$ where s is the standard deviation of the sample values, and N is the number of elements in the sample.

To verify these assumptions, I compared my non-time-based results with the table presented in the paper, using the erroneous methods described above to generate derived metrics and estimates of mean standard error, as shown in Tables 1 and 2.

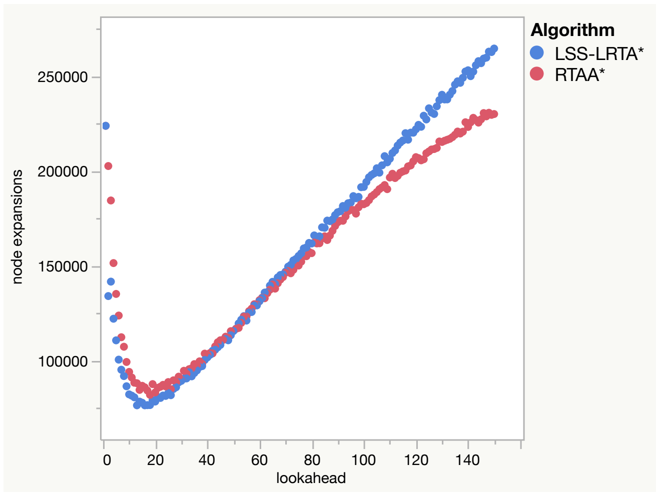
My results are similar to those in Koenig & Likhachev (2006), which suggests that my implementation of RTAA* is correct, and my interpretation of how their analysis erred is correct. However, there does seem to be a trend, even in non-derived metrics, for LSS-LRTA*: my implementations perform notably better at lower lookaheads and worse at higher lookaheads. For larger lookaheads, in the metric "heuristic increase per update," which represents each node’s h-value increase during the learning phase, the discrepancy is pretty large as a percent, although the absolute difference is still small, and a greater heuristic increase per update is favorable. In any case, exactly how much of the latter can be accounted for by the errors in Koenig & Likhachev (2006) is not clear.

lookahead	expansions	expansions [mean standard error]	search episodes	trajectory cost	trajectory cost [mean standard error[	action executions/ search episode	heuristic increase/ update
RTAA*							
Paper Results							
1	248537.79	5454.11	248537.79	248537.79	5454.11	1.00	1.00
9	104228.97	2196.69	11583.50	56707.84	1248.70	4.90	2.40
17	85866.45	1701.83	5061.92	33852.77	774.35	6.69	3.10
25	89257.66	1593.02	3594.51	26338.21	590.00	7.33	3.30
33	96839.84	1675.46	2975.65	22021.81	521.77	7.40	3.28
41	105702.99	1699.20	2639.59	18628.66	435.06	7.06	3.08
49	117035.65	1806.59	2473.55	16638.27	390.09	6.73	2.90
57	128560.04	1939.38	2365.94	15366.63	361.95	6.49	2.79
65	138640.02	2019.98	2270.38	14003.74	314.38	6.17	2.63
73	150254.51	2176.68	2224.29	13399.01	309.72	6.02	2.57
81	160087.23	2269.20	2172.94	12283.65	270.03	5.65	2.39
89	172166.56	2436.73	2162.75	12078.40	261.16	5.58	2.37
My Results							
1	223852.9	4880	223852.9	223852.9	4880	1.00	1.00
9	99201.62	2100	11024.92	53714.11	1188	4.87	2.39
17	84130.85	1625	4959.84	32915.07	730	6.64	3.07
25	88589.19	1607	3567.93	25990.32	593	7.28	3.28
33	95395.86	1625	2932.01	21374.63	494	7.29	3.22
41	103891.42	1655	2595.93	18043.36	413	6.95	3.02
49	115459.48	1772	2441.05	16355.55	372	6.7	2.88
57	127356.19	1895	2345.07	15096.23	340	6.44	2.75
65	140066.26	2045	2292.33	14236.98	321	6.21	2.64
73	147845.51	2131	2191.3	12956.61	284	5.91	2.49
81	162197.94	2311	2199.18	12793.9	286	5.82	2.46
89	170983.6	2423	2149.2	12027.24	256	5.6	2.36
% Difference							
1	-9.9	-10.5	-9.9	-9.9	-10.5	0.0	0.0
9	-4.8	-4.4	-4.8	-5.3	-4.9	-0.6	-0.4
17	-2.0	-4.5	-2.0	-2.8	-5.7	-0.7	-1.0
25	-0.7	0.9	-0.7	-1.3	0.5	-0.7	-0.6
33	-1.5	-3.0	-1.5	-2.9	-5.3	-1.5	-1.8
41	-1.7	-2.6	-1.7	-3.1	-5.1	-1.6	-1.9
49	-1.3	-1.9	-1.3	-1.7	-4.6	-0.4	-0.7
57	-0.9	-2.3	-0.9	-1.8	-6.1	-0.8	-1.4
65	1.0	1.2	1.0	1.7	2.1	0.6	0.4
73	-1.6	-2.1	-1.5	-3.3	-8.3	-1.8	-3.1
81	1.3	1.8	1.2	4.2	5.9	3.0	2.9
89	-0.7	-0.6	-0.6	-0.4	-2.0	0.4	-0.4

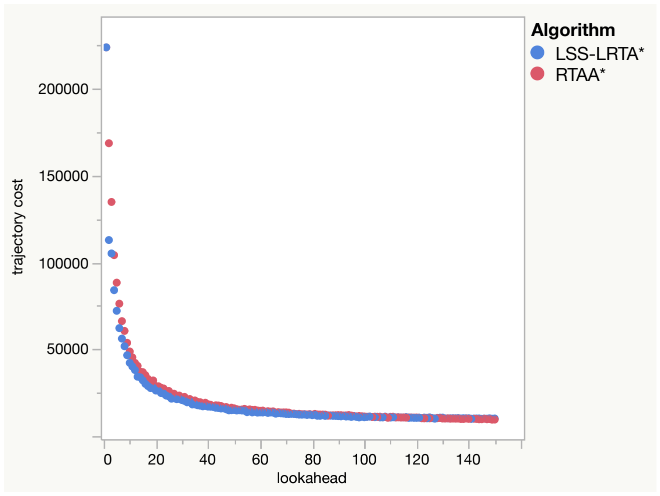
Table 1

lookahead	expansions	expansions [mean standard error]	search episodes	trajectory cost	trajectory cost [mean standard error[	action executions/ search episode	heuristic increase/ update
LSS-LRTA*							
Paper Results							
1	248537.79	5454.11	248537.79	248537.79	5454.11	1.00	1.00
9	87613.37	1865.31	9737.38	47290.61	1065.07	4.86	2.93
17	79312.59	1540.44	4676.76	30470.32	698.08	6.52	3.61
25	82850.86	1495.61	3338.86	23270.38	551.75	6.97	3.74
33	92907.75	1548.37	2858.19	20015.55	472.86	7.00	3.71
41	102787.86	1619.33	2570.83	17274.12	403.65	6.72	3.54
49	113139.63	1716.88	2396.66	15398.47	360.45	6.42	3.38
57	125013.41	1829.10	2307.68	14285.14	328.39	6.19	3.25
65	133863.67	1956.49	2201.60	13048.50	300.69	5.93	3.12
73	146549.69	2080.76	2181.76	12457.92	277.60	5.71	3.02
81	157475.45	2209.65	2150.04	11924.96	262.61	5.55	2.95
89	166040.29	2355.33	2102.91	11324.72	246.94	5.39	2.88
My Results							
1	223852.9	4880	223852.9	223852.9	4880	1.00	1.00
9	86384.27	1853	9600.43	46574.03	1055	4.85	2.93
17	76412.89	1506	4501.87	28811.16	676	6.4	3.55
25	82808.73	1467	3325.83	22870.93	533	6.88	3.72
33	93799.07	1544	2864.57	19915.57	462	6.95	3.73
41	102953.62	1601	2544.22	16913.02	389	6.65	3.59
49	113310.64	1681	2357.05	14942.37	340	6.34	3.46
57	125578.93	1788	2261.57	13666.98	300	6.04	3.35
65	141503.22	1985	2250.01	13542.4	299	6.02	3.41
73	152804.62	2148	2181.46	12742.63	282	5.84	3.38
81	165998.27	2280	2154.68	12261.06	266	5.69	3.36
89	176718.28	2437	2106.07	11618.75	250	5.52	3.34
% Difference							
1	-9.9	-10.5	-9.9	-9.9	-10.5	0.0	0.0
9	-1.4	-0.7	-1.4	-1.5	-0.9	-0.2	0.0
17	-3.7	-2.2	-3.7	-5.4	-3.2	-1.8	-1.7
25	-0.1	-1.9	-0.4	-1.7	-3.4	-1.3	-0.5
33	1.0	-0.3	0.2	-0.5	-2.3	-0.7	0.5
41	0.2	-1.1	-1.0	-2.1	-3.6	-1.0	1.4
49	0.2	-2.1	-1.7	-3.0	-5.7	-1.2	2.4
57	0.5	-2.2	-2.0	-4.3	-8.6	-2.4	3.1
65	5.7	1.5	2.2	3.8	-0.6	1.5	9.3
73	4.3	3.2	0.0	2.3	1.6	2.3	11.9
81	5.4	3.2	0.2	2.8	1.3	2.5	13.9
89	6.4	3.5	0.2	2.6	1.2	2.4	16.0

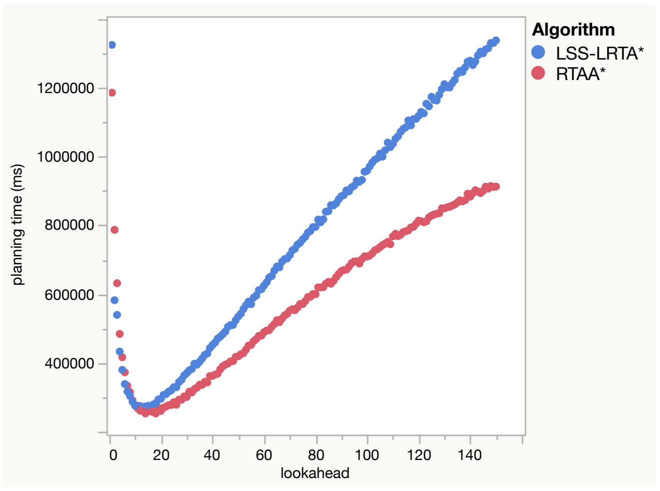
Table 2



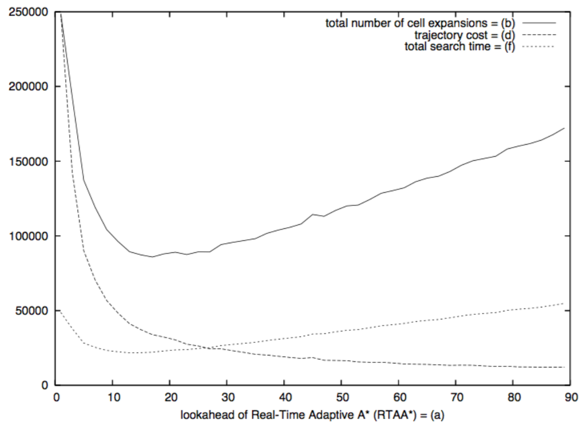
(a) mazes, unknown terrain: node expansions vs lookahead



(b) mazes, unknown terrain: trajectory cost vs lookahead (see Appendix A for log scale)



(c) mazes, unknown terrain: total planning time vs lookahead



(d) Koenig & Likhachev

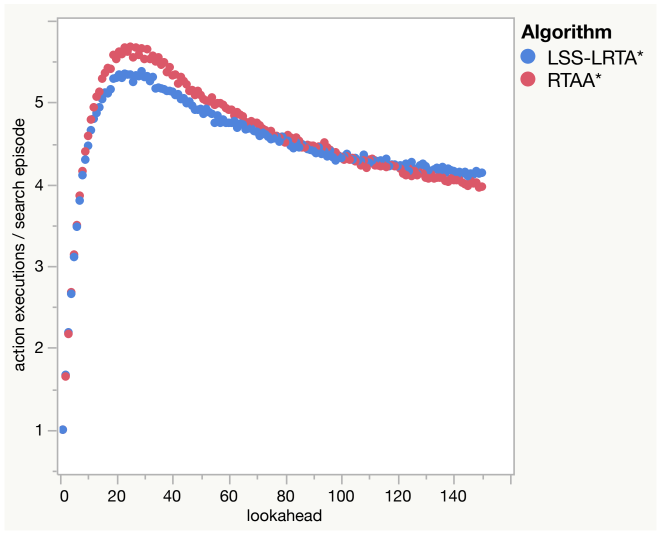
Figure 2

My Results

In generating the plots, I ran each algorithm over the same set of mazes for lookahead values from 1 to 150. The overall shapes of the plots in Figures 2a, 2c, and 2b are similar to the plots given in Koenig & Likhachev, shown in Figure 2d.

The J-shape of the plot in Figure 2a is a consequence of the relationships between lookahead number (hereafter simply "lookahead"), number of search episodes, and number of node expansions.

With a small lookahead, the paths given by the A* search are short and not close to optimal, and there is little information gained during the learning phase, so many search episodes, and consequently, node expansions, are required to find a path to the goal. As the lookahead increases, better paths can be found and more learning



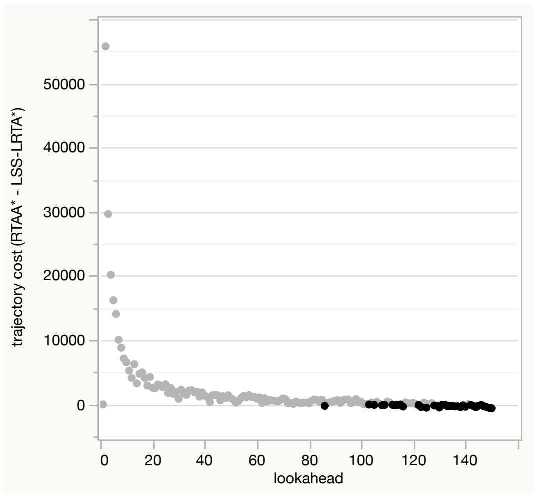
(e) mazes, unknown terrain: action executions per search episode vs lookahead

Figure 2: (continued)

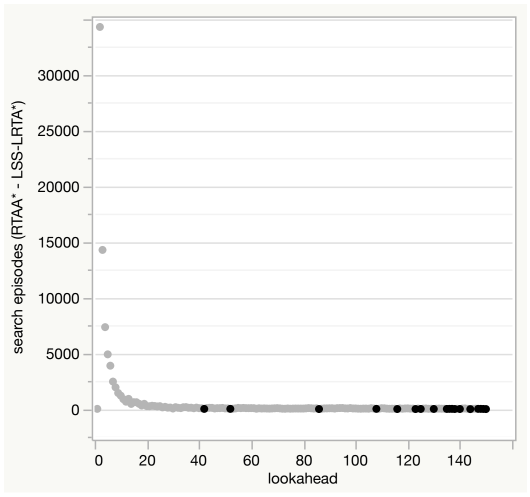
takes place, and as a result, the agent requires fewer search episodes, and consequently fewer node expansions, before the goal is found. Eventually, however, as the lookahead increases, the marginal benefit of an increase in the lookahead relative to the number of action executions per search episode begins to decrease. This is because of the freespace assumption. With larger and larger lookaheads, more of the obstacle nodes within the local search space are incorrectly assumed to be traversable and thus a greater proportion of the path given by the A* search will be blocked by obstacles.

With better heuristics, a larger proportion of the expanded nodes will be along the path found during the A* search, but in Figure 2e, we can see that the number of action executions per search episode never exceeds 7, and even decreases as the number of lookaheads increases, demonstrating that, as expected, using the freespace assumption, with larger lookaheads, the more obstacle-ridden the path given by A* is likely to be. Because the A* search with freespace assumption gives such poor paths and the agent starts a new A* search every time it encounters an obstacle, the number of action executions per search episode will not increase very much even as the number of node expansions per search episode increases approximately linearly with lookahead.

Another fact to note is that the number of node expansions is less than we would expect simply by multiplying the number of search episodes by the lookahead. If the number of node expansions were simply a product of the lookahead and the number of search episodes, even for large lookaheads, the increase in node expansions vs. lookahead would be much more linear. The reason for the difference, again, is the freespace assumption. Often, A* search finds a path to the goal node before all of the nodes allowed by the lookahead have been exhausted. Since this path is often blocked by obstacles, there may be many search episodes with fewer node expansions than allowed



(f) mazes, unknown terrain: trajectory cost (RTAA* - LSS-LRTA*) vs lookahead



(g) mazes, unknown terrain: search episodes (RTAA* - LSS-LRTA*) vs lookahead

Figure 2: (continued)

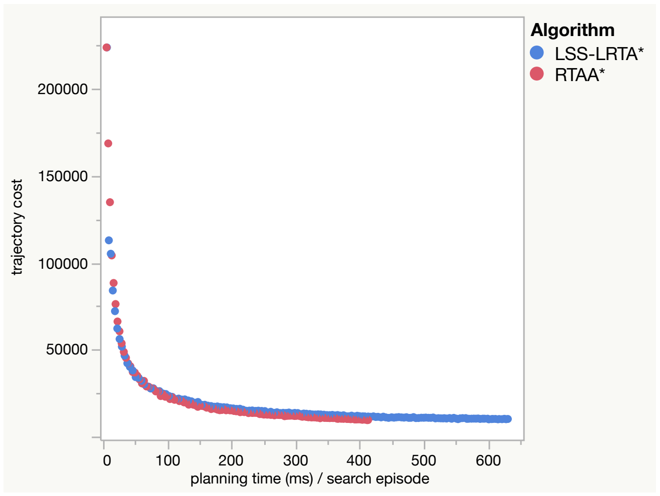
by the lookahead.

The shape of the plot in Figure 2c is of course similar to that in Figure 2a. A greater number of node expansions results in a larger local search space and more time planning.

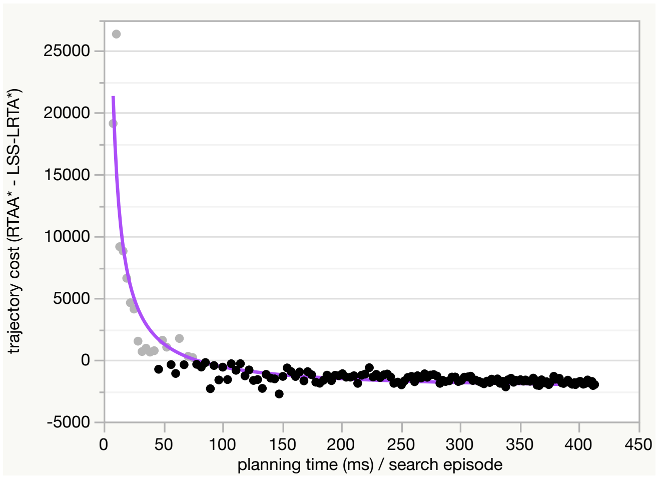
The plot in Figure 2b is easily understood. As one would expect, as the lookahead increases, the agent can observe more of its environment, and thus find a shorter path to the goal. This means, that although with increasing lookahead, the paths given by A* search may be more filled with obstacles, they are not necessarily worse. Just as in life.

In Figures 2f and 2g, data points that fall below 0 are darkened. We can see that at larger lookaheads, RTAA* outperforms LSS-LRTA* both in terms of trajectory cost and in terms of search episodes. Because in theory, LSS-LRTA* has a better learning strategy, we might expect the trajectory costs of its solutions to be shorter than RTAA*'s for a given lookahead. One explanation for this paradoxical result could again be related to the freespace assumption. With larger A* lookaheads, a larger proportion of the nodes within the local search space will be incorrectly assumed to be traversable. The learning strategy of LSS-LRTA* depends directly on the h-values of all of the nodes within the local search space. This thoroughness, in the context of a large number of nodes incorrectly assumed to be traversable, might actually result in less accurate updates to the heuristic values of the nodes within the local search space than those given by RTAA*, whose learning strategy depends, essentially, on only a single node. It should also be noted that the heuristic estimate used in the algorithms, Manhattan distance, is poor for mazes.

Since our ultimate aim is to discover which algorithm performs better for a given time limit on the planning



(h) mazes, unknown terrain: trajectory cost vs planning time / action execution (see Appendix A for log scale)



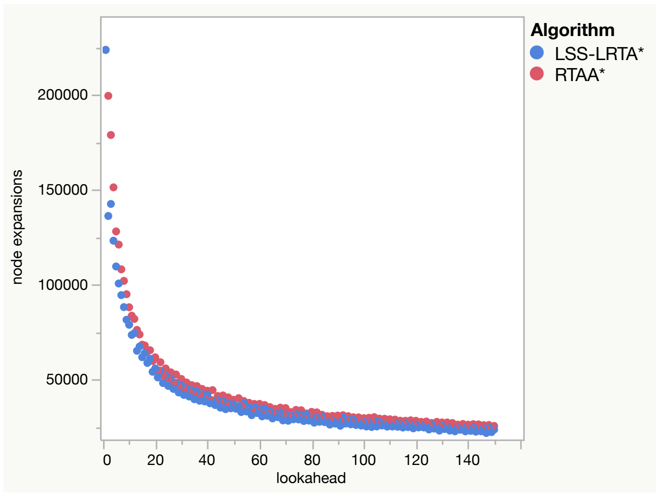
(i) mazes, unknown terrain: trajectory cost (RTAA* - LSS-LRTA*) vs lookahead

Figure 2: (continued)

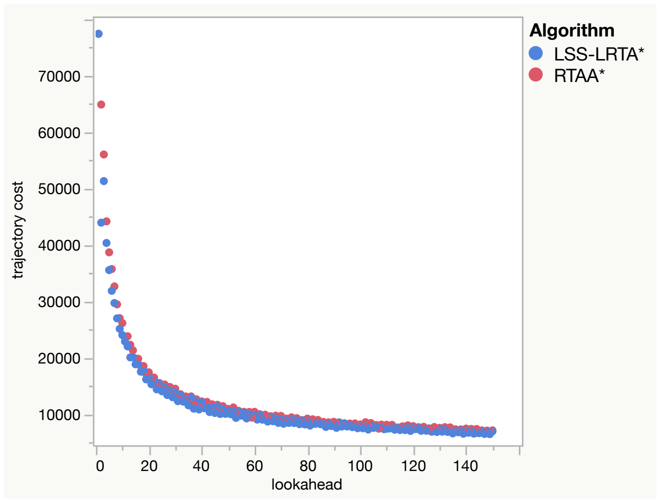
phase, the most important metric to evaluate is the trajectory cost vs. planning time per search episode, as plotted in Figure 2h. In order to compare the performance of two algorithms for a given time limit, the data points for planning time per search episode must match, but they do not, because planning time per search episode is a derived metric. To work around this, I inferred data points between existing data points for LSS-LRTA* using linear interpolation. I chose to interpolate between data points for LSS-LRTA* because the data points for LSS-LRTA* cover a larger range of planning times per search episode. The differences are plotted in Figure 2i. Data points shaded black represent lookaheads with time-limits at which RTAA* outperforms LSS-LRTA*.

According to Koenig & Likhachev, when a search episode is limited to 20.99 microseconds or less, using optimal lookahead values: 67 and 43 for RTAA* and LSS-LRTA*, respectively, RTAA* delivers lower trajectory costs than LSS-LRTA* (notwithstanding the dubiousness of Koenig & Likhachev's data). Because there is no single time-limit above which RTAA* begins outperforming LSS-LRTA*, I overlaid with a regression line ((trajectory cost) = -2476.624 + 191233.06/(planning time (ms) per search episode), r^2 = 0.87). In order to provide a more direct comparison with the claims made my Koenig & Likhachev, I chose the x-intercept of the regression equation, 77.22ms / search episode, as a reasonable time limit above which we might expect RTAA* to outperform LSS-LRTA*, corresponding to a lookahead between 18 and 19 for LSS-LRTA* and between 22 and 23 for RTAA*.

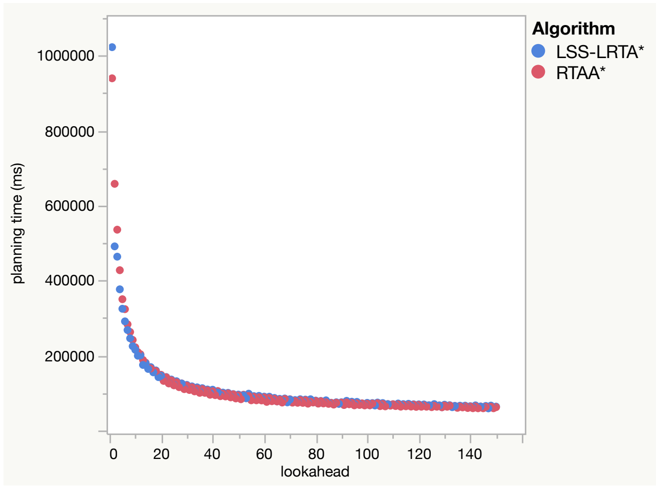
In spite of this, my results are not really comparable to Koenig & Likhachev's. In theory, the greater number of cells expanded by RTAA* for a given time limit make up for less thorough learning, and the effects of this are amplified for tighter time limits. However, my results showed just the opposite. As time per search episode increases, RTAA* performs better relative to LSS-LRTA*. Since lookahead and time per search episode are directly



(a) mazes, known terrain: node expansions vs lookahead (see Appendix A for log scale)



(b) mazes, known terrain: trajectory cost vs lookahead (see Appendix A for log scale)



(c) mazes, known terrain: total planning time vs lookahead (see Appendix A for log scale)

Figure 3

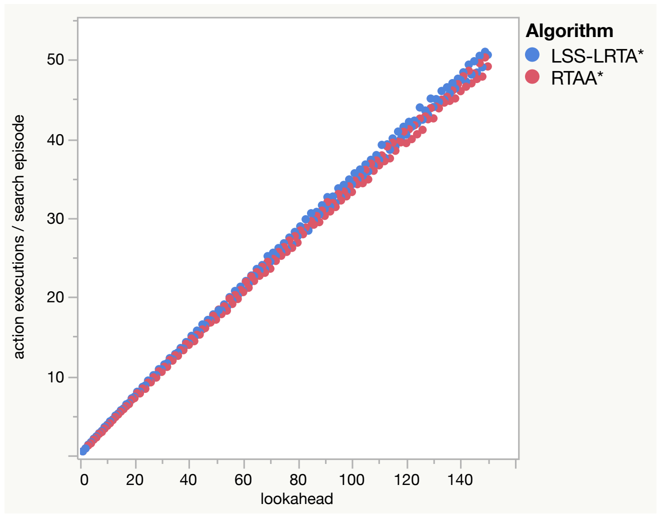
related, this accords with observation that RTAA* performs better relative to LSS-LRTA* as lookahead increases.

Known Terrain

The mazes used with known terrain are identical to those used with unknown terrain.

Results and Discussion

Predictably, trajectory cost, number of node expansions and planning time decrease as lookahead increases, as shown in Figures 3a, 3b, and 2c. Note that the shape of the plot in Figure 3a differs from the plot in Figure 2a. With some simplifying assumptions, and some reasoning, we can see why. Because the terrain is known, each



(d) mazes, known terrain: action executions per search episode vs. lookahead

Figure 3: (continued)

expansion results in more action executions per search episode and therefore fewer search episodes, and therefore fewer node expansions overall.

This is especially apparent after considering Figure 3d. The ratio of action executions per search episode to the number of node expansions per search episode, for which, with known terrain, lookahead is a good proxy, is approximately constant. To see how this affects the shape of the plots, let l = lookahead, a = # of action executions, e = # of search episodes, t = trajectory cost, and n = # of node epxansions. Let c_1, c_2, and c_3 be some constants. As we can see in Figure 3d, roughly,

$$\frac{a}{e} = l \times c_1$$

It is also clear that,

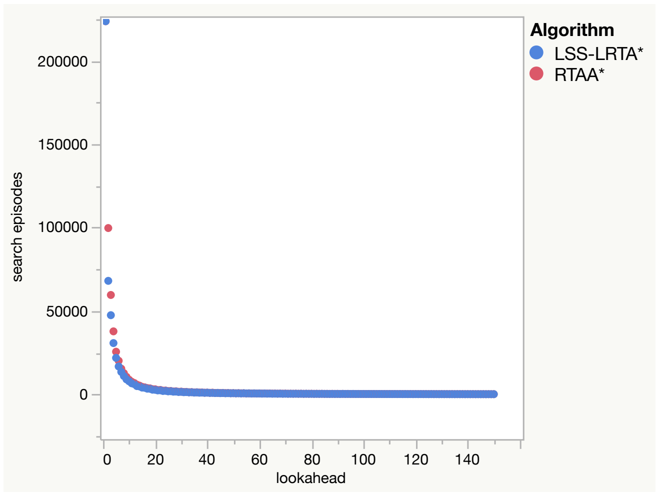
$$e = \frac{t}{\frac{a}{e}}$$

It seems reasonable to assume, based on Figure 3b that t is inversely proportional to l, so we can say,

$$t = \frac{c_2}{c_3 \times l}$$

Furthermore,

$$n = e \times l$$



(e) mazes, known terrain: search search episodes vs lookahead (see Appendix A for log scale)

Figure 3: (continued)

so, putting n in terms of l by expanding terms, we get,

$$n = \frac{\frac{c_2}{(c_3 \times l)}}{l \times c_1} \times l$$

and thus,

$$n = \frac{c_2}{c_1 c_3 l}$$

The number of node expansions is therefore inversely proportional to the lookahead, and this is reflected in Figure 3a.

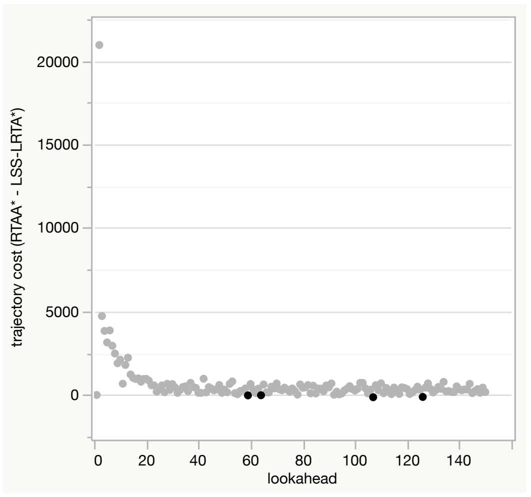
Moreover, by expanding terms, we can also see that the number of search episodes ought to be inversely proportional to the square of the lookahead.

$$e = \frac{t}{\frac{a}{e}}$$

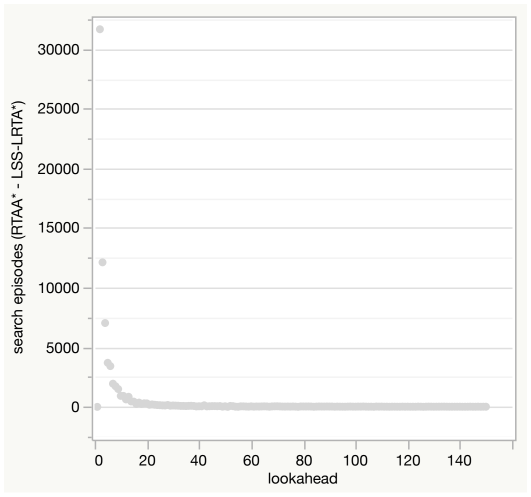
expands to

$$e = \frac{\frac{c_2}{(c_3 l)}}{l c_1}$$

which reduces to



(f) mazes, known terrain: trajectory cost (RTAA* - LSS-LRTA*) vs lookahead



(g) mazes, known terrain: search episodes (RTAA* - LSS-LRTA*) vs lookahead

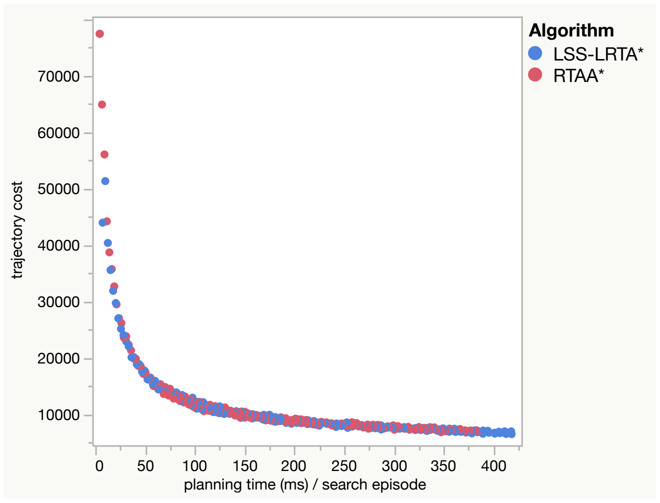
Figure 3: (continued)

$$e = \frac{c_2}{c_1 c_3 l^2}$$

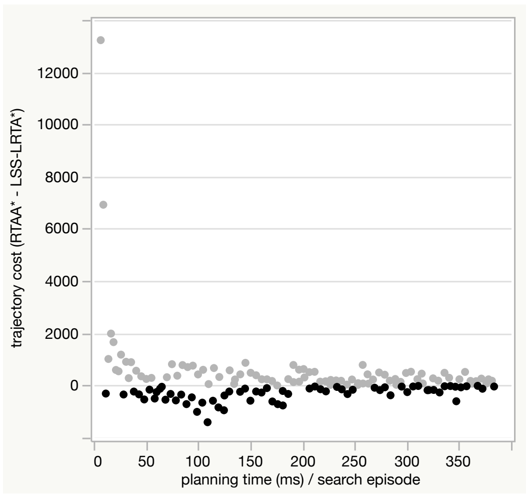
This accords with the shape of the plot in Figure 3e (In functions of the form $y = \frac{k}{x^n}$, as n increases, the graph will resemble more and more an L-shape).

Without losing sight of the object of this inquiry, we can compare the relative performance of RTAA* and LSS-LRTA* by looking at Figures 3f and 3g. At no lookahead does RTAA* clearly outperform LSS-LRTA*. The four instances where it obtains a shorter trajectory cost can easily be accounted for by random tie-breaking.

More important, however, is the performance of the two algorithms with regard to trajectory cost vs. planning time per search episode, as shown in Figure 3h. Differences (RTAA* - LSS-LRTA*) with data points for planning time per search episode interpolated from LSS-LRTA* are shown in Figure 3i. Apart from very short planning times, where LSS-LRTA* outperforms RTAA*, neither clearly outperforms the other. Why this should be is not obvious. Perhaps Manhattan distance is such a poor heuristic in mazes that even a good learning strategy doesn't make a difference when many nodes are expanded.



(h) mazes, known terrain: trajectory cost vs planning time (ms) / search episode (see Appendix A for log scale)



(i) mazes, known terrain: trajectory cost (RTAA* - LSS-LRTA*) vs planning time (ms) / search episode

Figure 3: (continued)

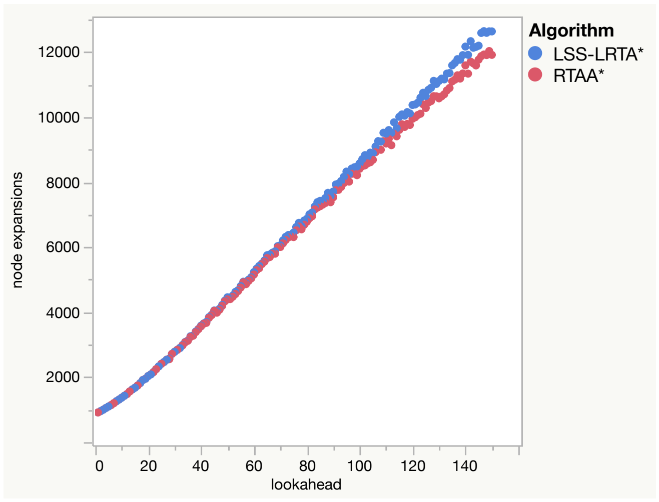
2.1.3 Random Maps

In theory, RTAA* trades slow, thorough, learning for fast, not-so-thorough learning. While LSS-LRTA* uses a learning strategy that results in unique, carefully calculated f-values for every node within the local search space, RTAA* gives a local search space in which every node has the same f-value. If we imagine the state space as a topography of f-values, after RTAA*'s learning phase, the local search space would appear as a flat plane. We might therefore expect that, in general, the more the state space resembles a flat plane, the less the f-values produced by RTAA* will vary from the "true" f-values, and the better RTAA* will perform relative to LSS-LRTA*. To test this assumption, I ran both algorithms over 512×512 grid maps randomly filled with obstacles at 10,20,30, and 40%. The library [3] from which they were sourced orders scenarios into buckets according to optimal solution length. I chose 100 instances from buckets with the longest optimal solution lengths.

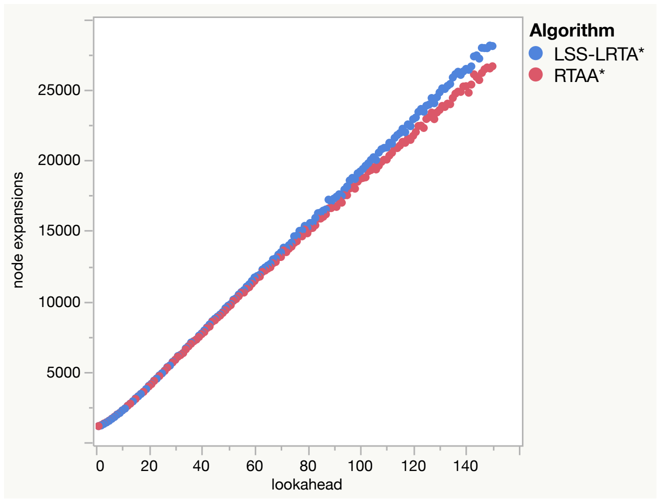
Unknown Terrain

Results and Discussion

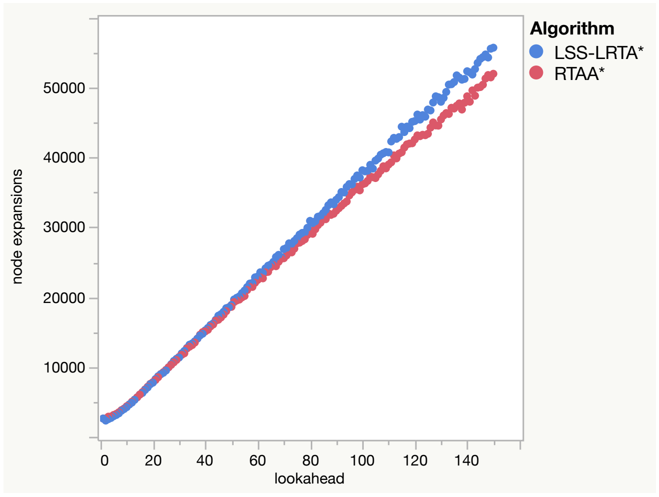
One interesting observation is that for grid maps 10, 20, and 30% filled with obstacles, plots of node expansions vs. lookahead are more or less linear, as shown in Figures 4a,4b, and 4c show. The J-shape observed in mazes with unknown terrain is only apparent for grid maps 40% filled with obstacles, as shown in Figure 4d. In fact, the overall shape of all of the plots 40% filled with obstacles resemble those of mazes, suggesting that there may not be anything particular about uniform mazes other than the percent of obstacle states they contain. I will explain, in my estimation, why the plots in Figures 4a, 4b, and 4c are approximately linear,while the plot in Figure 4d is



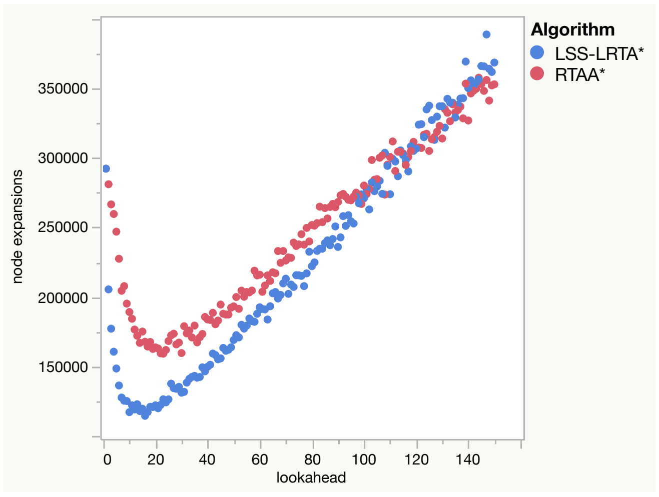
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%



(d) random obstacles 40%

Figure 4:
Random Maps, Unknown Terrain: Node Expansions vs. Lookahead

J-shaped. Let l = lookahead, a = # of action executions, e = # of search episodes, t = trajectory cost, n = # of node epxansions, and r = # of restarts.

Consider a grid map in which there are no obstacles. t is the optimal trajectory cost.

Notwithstanding the final search episode, during which the agent might reach the goal before the lookahead is exhausted, is clear that the following equations are exactly true:

$$n = l \times e$$

likewise,

$$e = \frac{t}{l}$$

Say we add a single obstacle along the optimal path. Since we are using an algorithm for unkonwn terrain, we will have to restart when we encounter it during action execution, which means that we will have an additional search episode. This will not appreciably change the trajectory cost, the number of search episodes, or any other parameter. If we add obstacles only along the optimal path, we can calculate approximately, at least for small numbers of obstacles,

$$e = \frac{t}{l} + r$$

and so,

$$n = l\Big(\frac{t}{l} + r\Big)$$

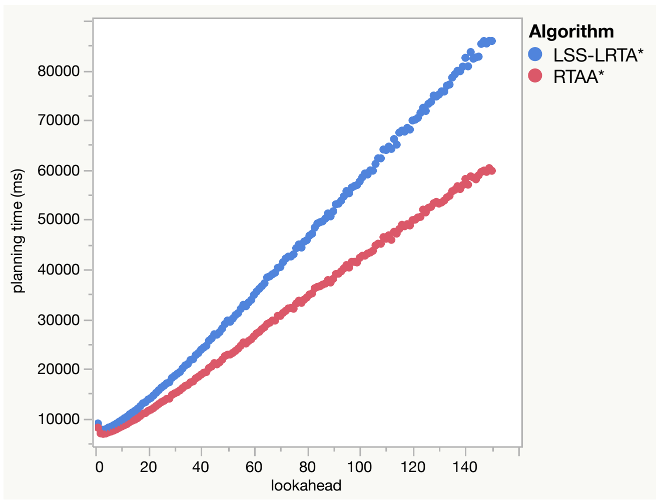
that is,

$$n = t + lr$$

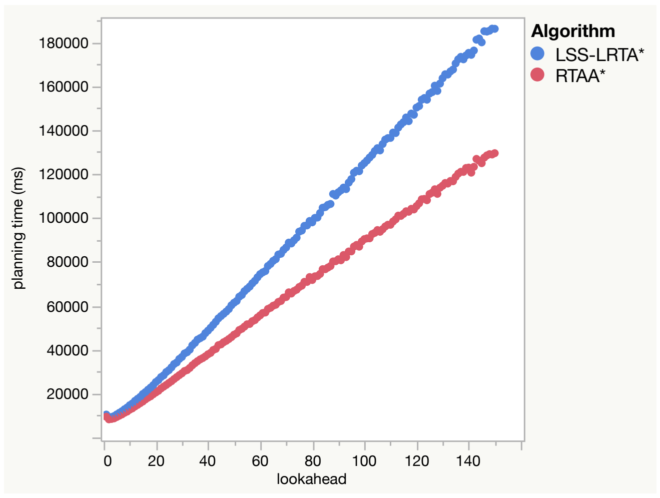
As long as there are not so many obstacles as to necessitate a very large deviation from the optimal path, this linear equation should approximate the number of node expansions, and accords with the plots in Figures 4a,4b, and 4c. The case is different with higher numbers of obstacles, such as in Figure 4d. It is not easy to find a mathematical expression that describes its behavior, although it seems reasonable to say that, while in grids with a low obstacle density, the path to the goal is pretty close to optimal, and increases in the number of node expansions with lookahead are mostly driven by incidental encounters with obstacles, in grids with high-obstacle density, for smaller lookaheads, when the freespace assumption is not very inaccurate, a larger lookahead has some utility for avoiding obstacles.

As expected, plots of planning time vs. lookahead, as shown in Figure 5 roughly accord with the plots of node expansions vs. lookahead in Figure 4, and of course RTAA* is faster than LSS-LRTA*.

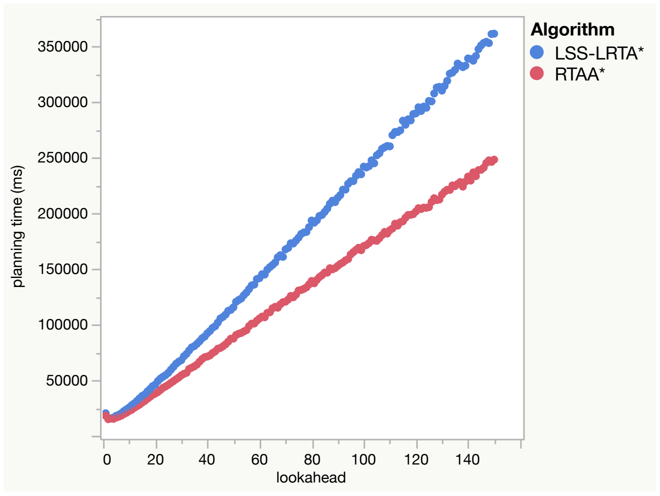
As mentioned earlier in the discussion on mazes with unknown terrain, due to the freespace assumption, with larger lookaheads, as lookahead increases, the marginal benefit of an increase in lookahead with respect to actions per search episode decreases. With a large lookahead, a larger proportion of nodes within the local search space are incorrectly assumed to be traversable, the path given will have a greater number of obstacles, which the increases the likelihood of encountering an obstacle during action execution. We can see the effects of this in Figures 6a, 6b, and 6c. The agent tends to run into an obstacle after the same number of action executions, regardless of the



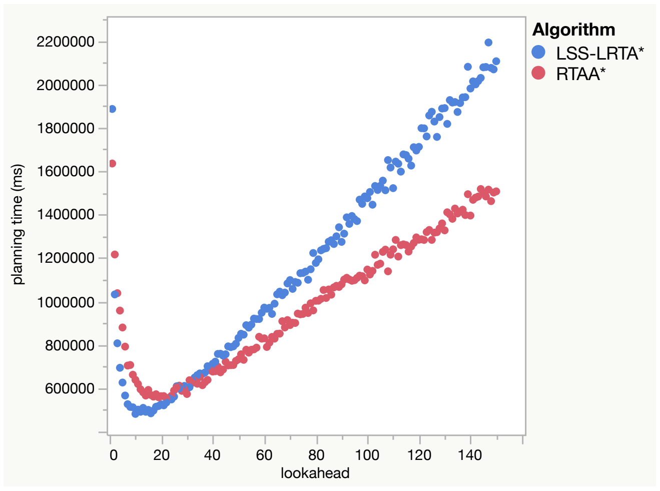
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%



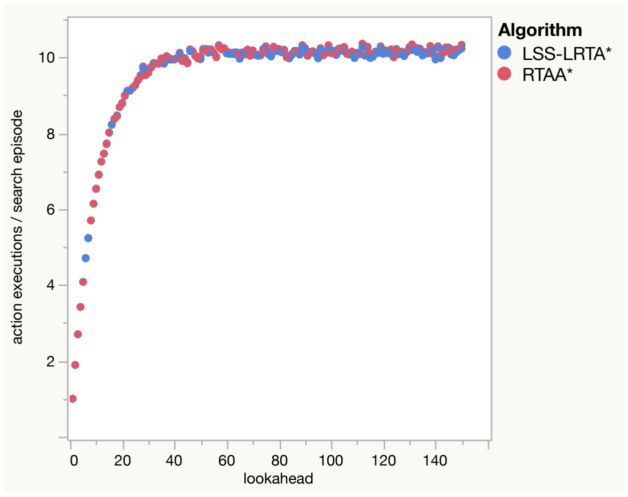
(d) random obstacles 40%

Figure 5: Random Maps, Unknown Terrain: total planning time vs. Lookahead

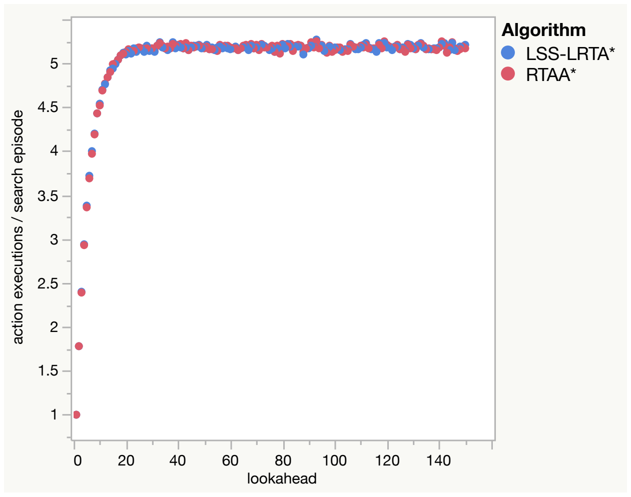
lookahead (which is almost exactly the same thing as the number of expansions per search episode). In fact, the results shown in the plots can be modelled with a marvelous equation, which the margin of this thesis is too narrow to contain.

Figure 6d is an anomaly, however. After increasing with lookahead, actions per search episode begin to decrease again. The most intuitive explanation for this is that, using the freespace assumption, ever larger lookaheads lead to more obstacle-ridden paths, just as with mazes.

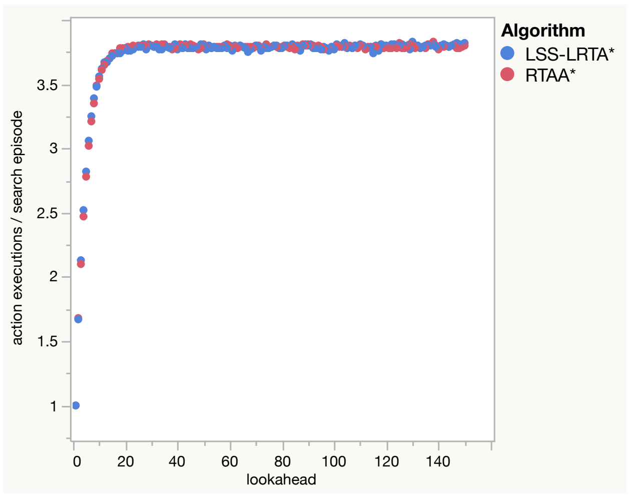
As expected, in general, the fewer obstacles in the grid map, the better RTAA* performs relative to LSS-LRTA*, in terms of number of search episodes, as shown in Figure 7. Moreover, RTAA* performs better relative to LSS-LRTA* as lookahead increases, in accord with the pattern observed in mazes. This can be seen in plots of the differences in the number search episodes vs. lookahead (RTAA* - LSS-LRTA*) in Figure 8.



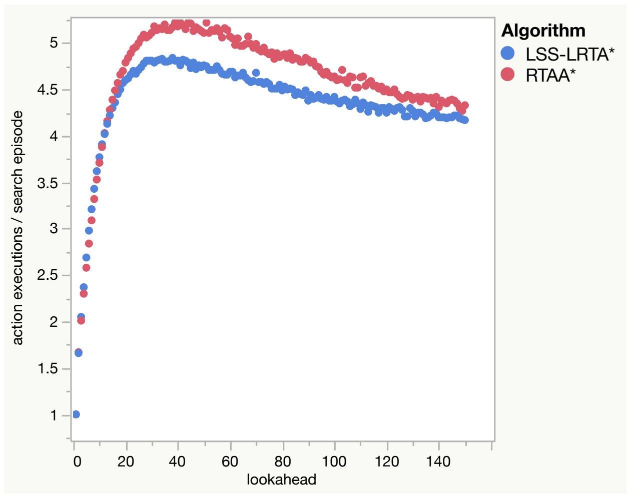
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%



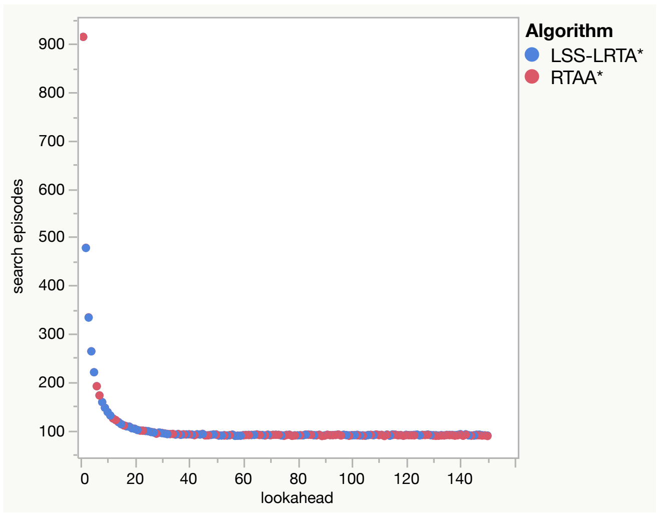
(d) random obstacles 40%

Figure 6:
Random Maps, Unknown Terrain: Action Executions per Search Episode vs. Lookahead

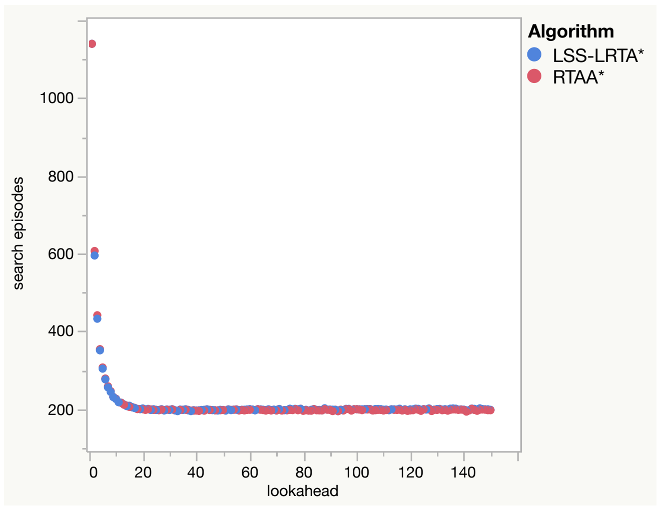
The plots in Figures 9 and 10 show again that RTAA* outperforms LSS-LRTA* with larger lookaheads.

It is also might be valuable to note that the data points in the plots in Figures 7a and 9a are close to their theoretical values in an obstacle-free grid. With few obstacles, t is close to optimal and doesn't change much with increases in lookahead, and the number of search episodes is approximately inversely proportional to the optimal trajectory: $e = \frac{t}{l}$.

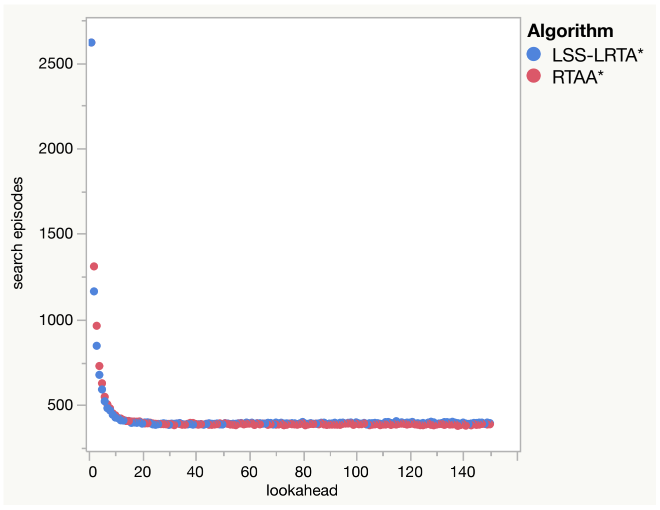
The same pattern of performance of RTAA* against LSS-LRTA*, wherein RTAA* outperforms LSS-LRTA* at larger lookaheads, is apparent in Figures 11 and 12.



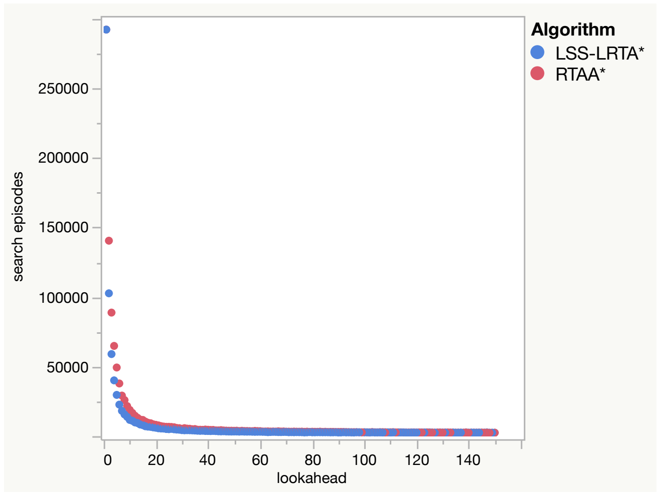
(a) random obstacles 10%



(b) random obstacles 20%

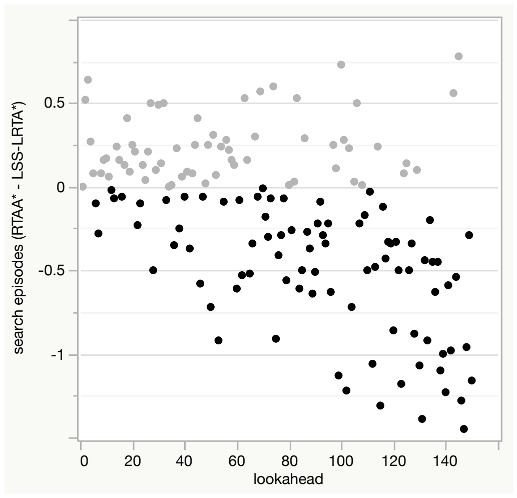


(c) random obstacles 30%

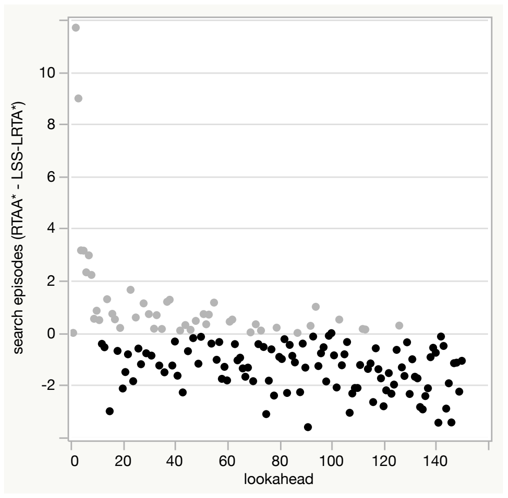


(d) random obstacles 40%

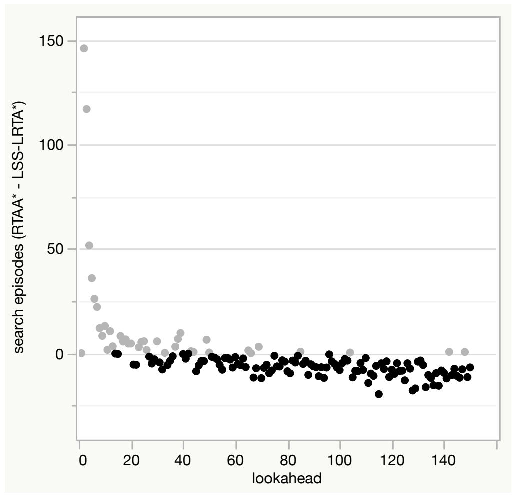
Figure 7:
Random Maps, Unknown Terrain: Search Episodes vs. Lookahead



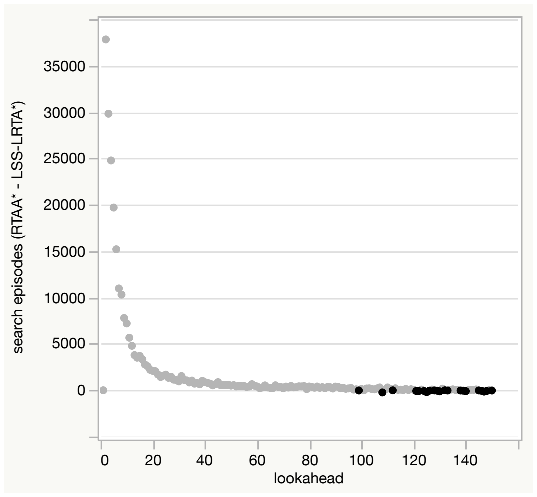
(a) random obstacles 10%



(b) random obstacles 20%

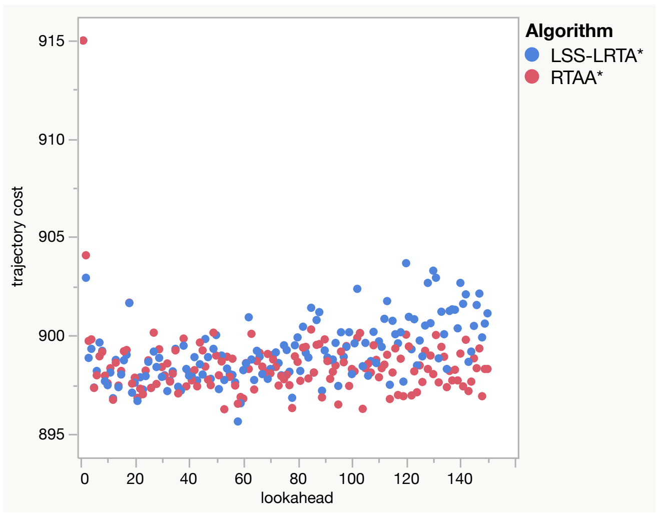


(c) random obstacles 30%

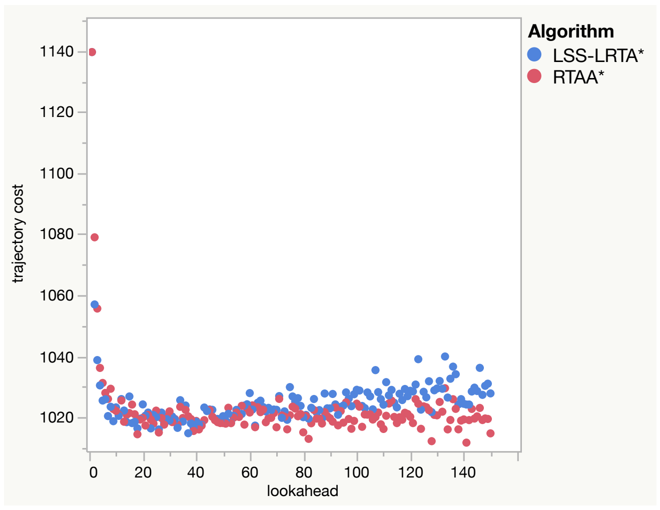


(d) random obstacles 40%

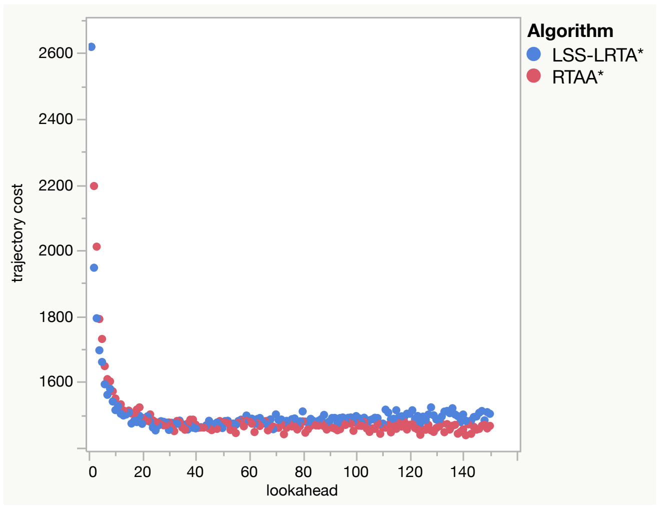
Figure 8:
Random Maps, Unknown Terrain: Search Episodes (RTAA* - LSS-LRTA*) vs. Lookahead



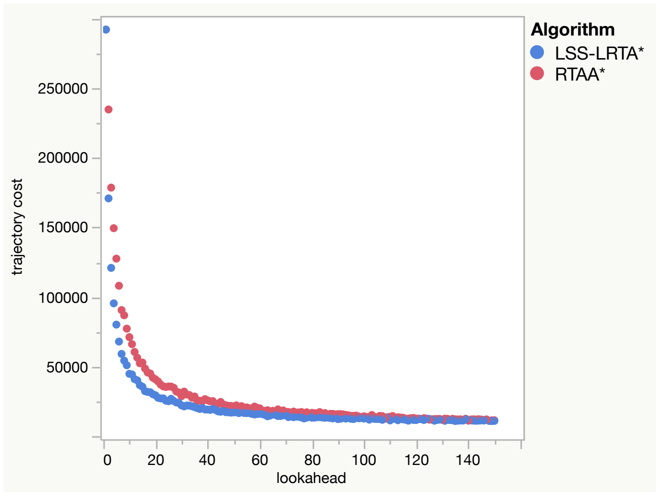
(a) random obstacles 10% (see Appendix A for log scale)



(b) random obstacles 20% (see Appendix A for log scale)

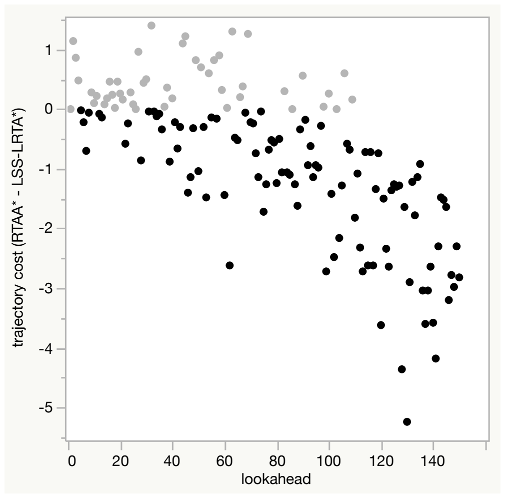


(c) random obstacles 30% (see Appendix A for log scale)

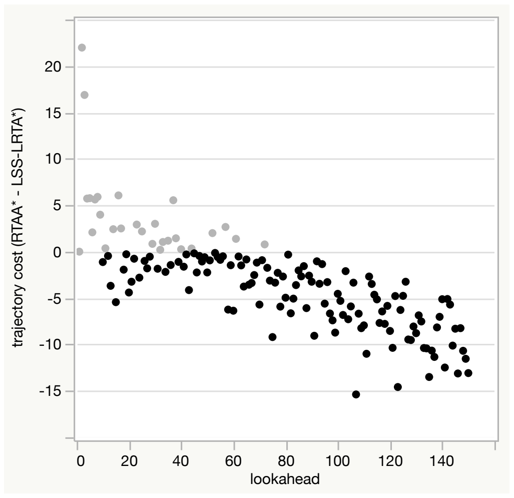


(d) random obstacles 40% (see Appendix A for log scale)

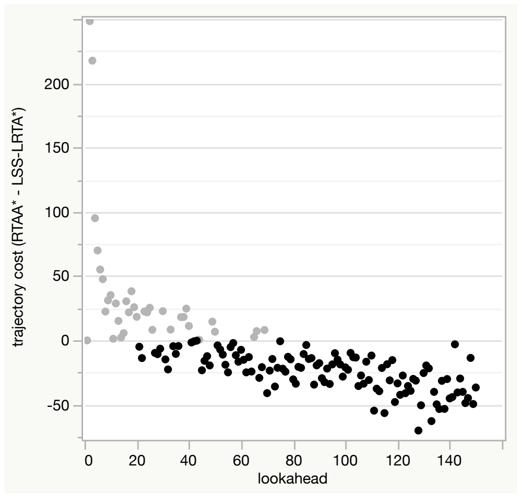
Figure 9:
Random Maps, Unknown Terrain: Trajectory Cost vs. Lookahead



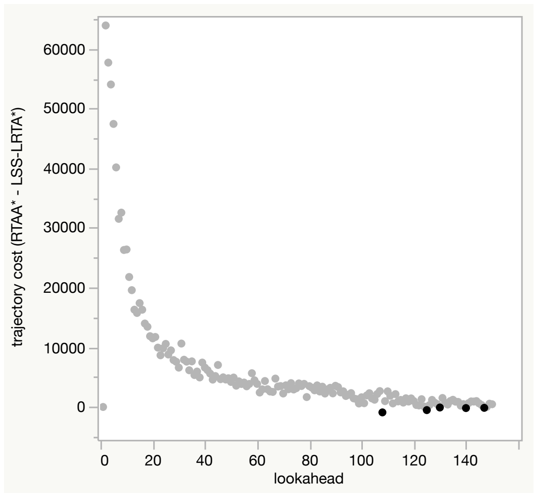
(a) random obstacles 10%



(b) random obstacles 20%

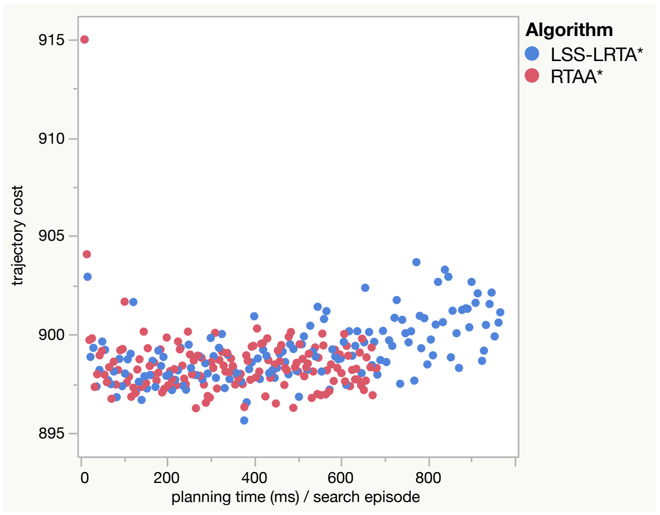


(c) random obstacles 30%

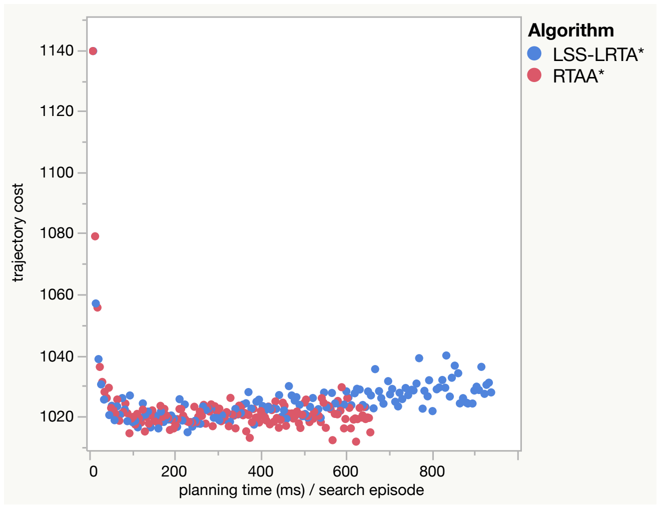


(d) random obstacles 40%

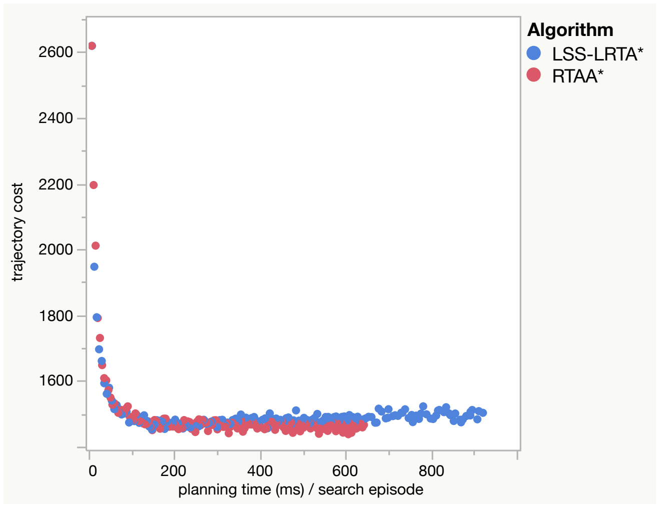
Figure 10:
Random Maps, Unknown Terrain: Trajectory Cost (RTAA* - LSS-LRTA*) vs. Lookahead



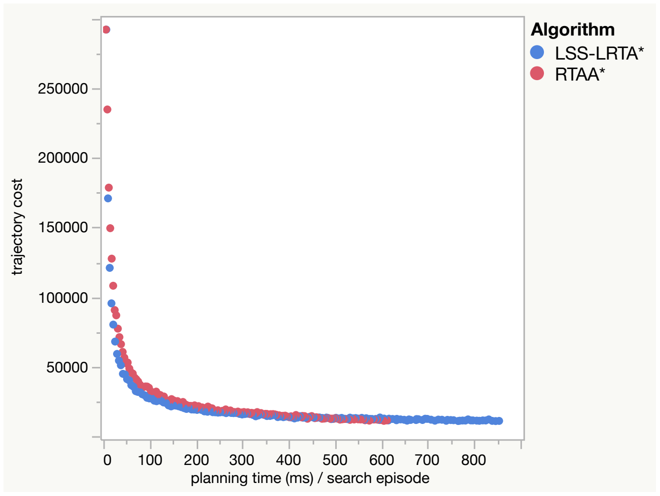
(a) random obstacles 10% (see Appendix A for log scale)



(b) random obstacles 20% (see Appendix A for log scale)

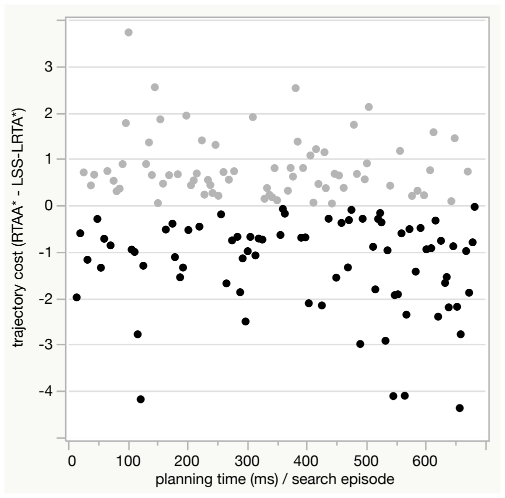


(c) random obstacles 30% (see Appendix A for log scale)

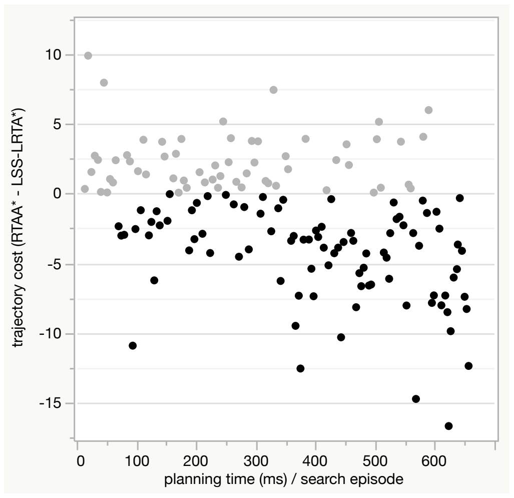


(d) random obstacles 40% (see Appendix A for log scale)

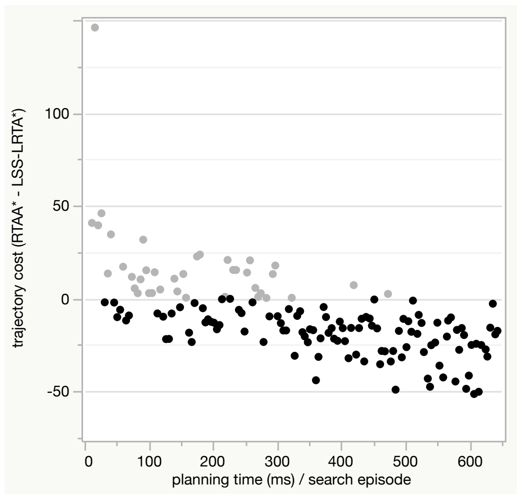
Figure 11:
Random Maps, Unknown Terrain: Trajectory Cost vs. Planning Time / Search Episode



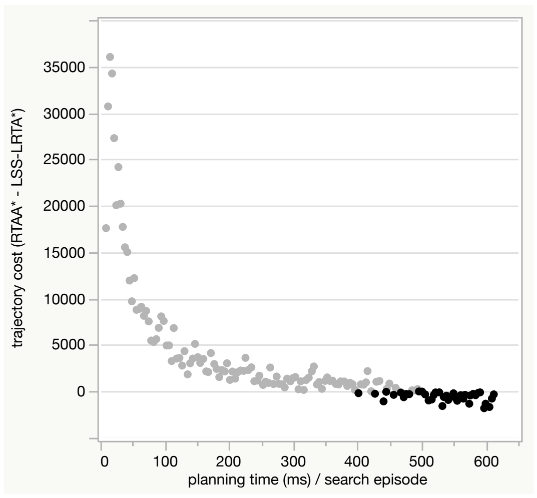
(a) random obstacles 10%



(b) random obstacles 20%

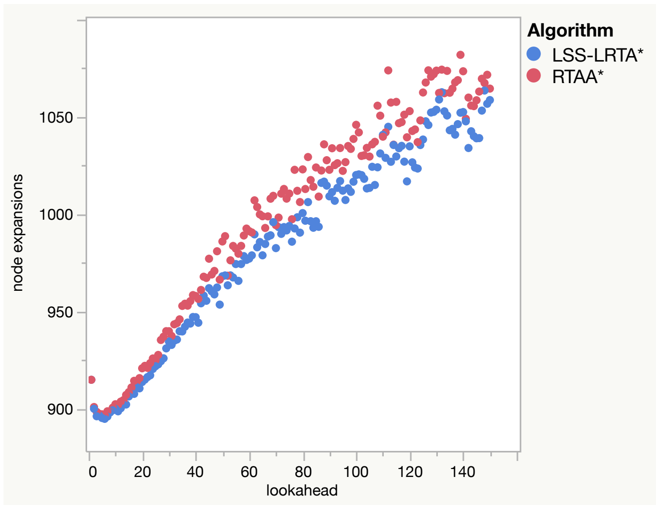


(c) random obstacles 30%

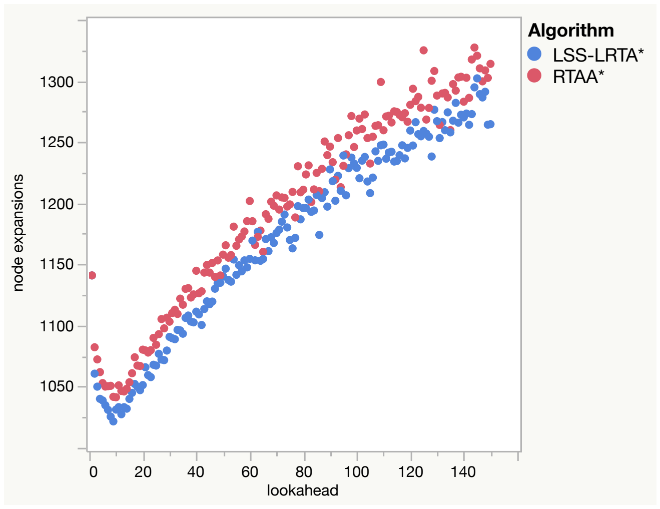


(d) random obstacles 40%

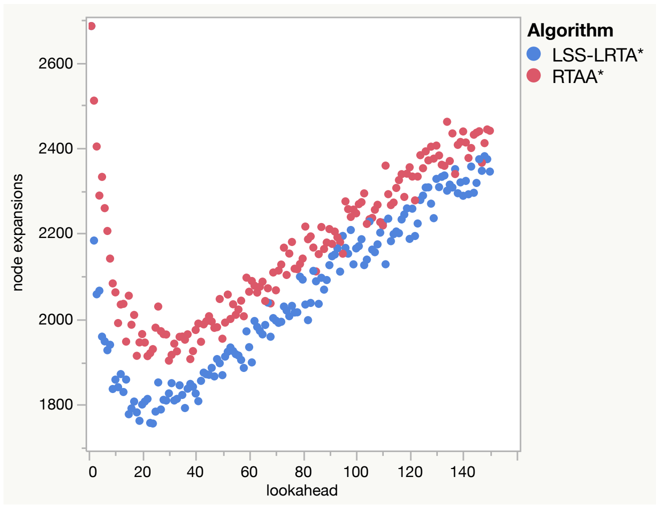
Figure 12:
Random Maps, Unknown Terrain: Trajectory Cost (RTAA* - LSS-LRTA*) vs. Planning Time / Search Episode



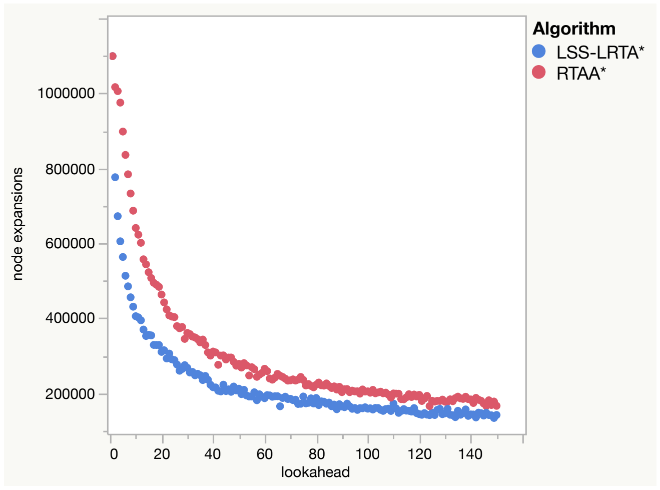
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%



(d) random obstacles 40%

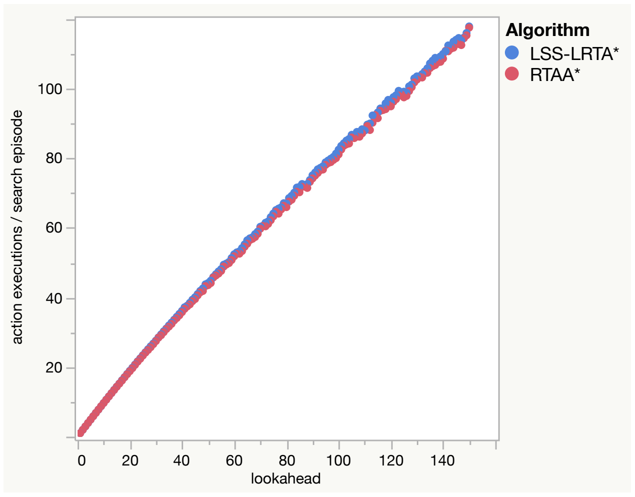
Figure 13:
Random Maps, Known Terrain: Node Expansions vs. Lookahead

<u>Known Terrain</u>

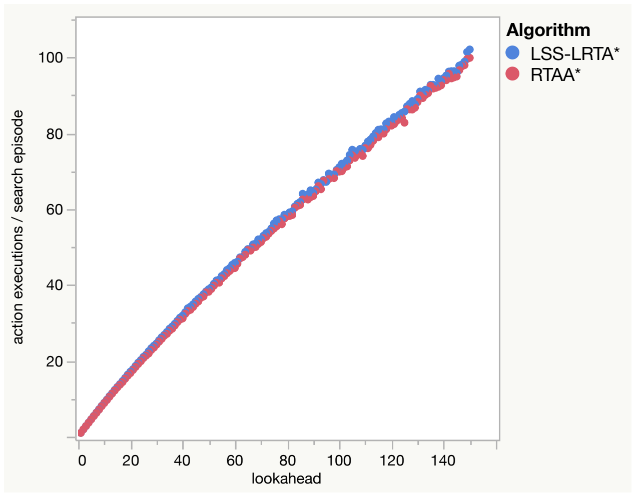
The random maps used with known terrain are identical to those used with unknown terrain.

Results and Discussion

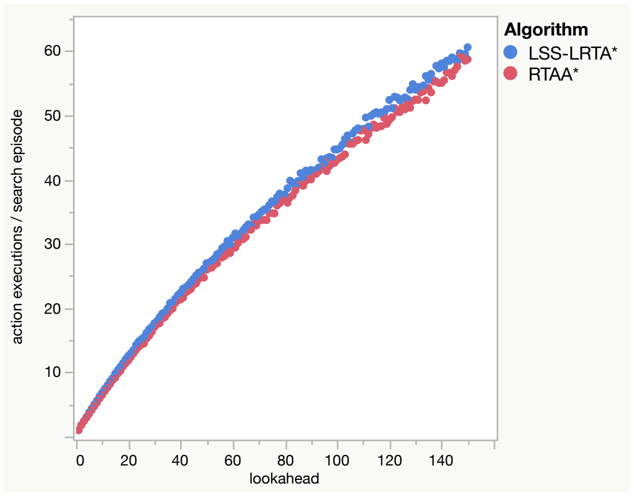
In Figure 13, we can see that, just as with unknown terrain, the real-time heuristic search algorithms seem to behave qualitatively differently when the grid is filled 40% with obstacles. I think it is possible to find a way of modeling the behavior of these algorithms run on low-obstacle grids mathematically, but for now, it is useful enough to observe that the marginal benefit of an increase in lookahead with respect to trajectory cost decreases for larger lookaheads. As to Figure 13d, I think it is enough to say that due to the difficulty of navigating through such a high density of obstacles, every extra expanded node is helpful. Figure 14 contains some interesting information hidden



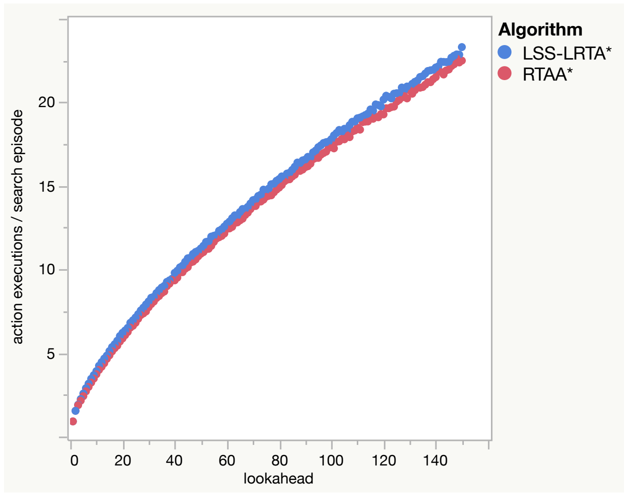
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%



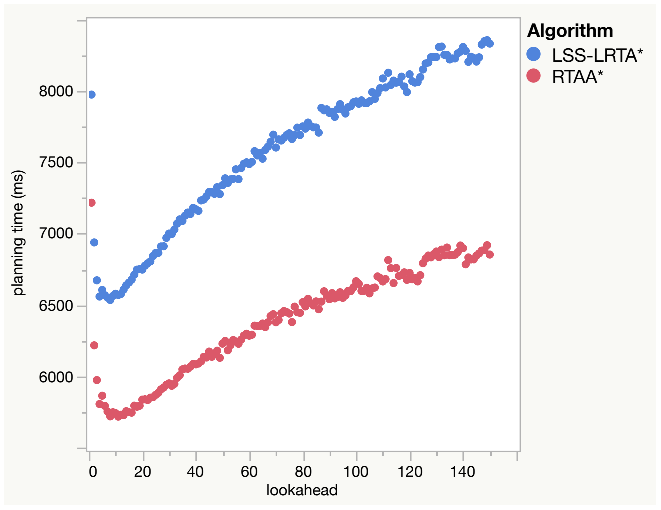
(d) random obstacles 40%

Figure 14:
Random Maps, Unknown Terrain: Action Executions per Search Episode vs. Lookahead

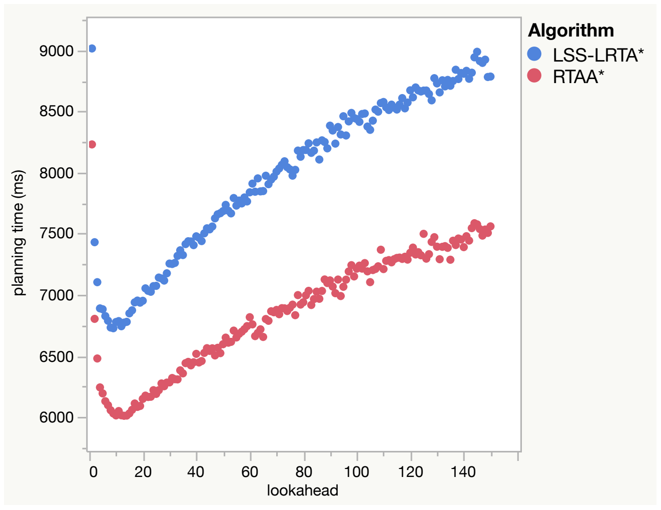
within it. The linear increase in action executions per search episode vs lookahead imply that the effectiveness of the heuristics is just about the same for all lookaheads: just about half the nodes that are expanded each search episode end up being a part of the path.

As usual, as Figure 15 shows, the curves of planning time vs. lookahead match the curves of node expansions vs. lookahead, and RTAA* is faster than LSS-LRTA*.

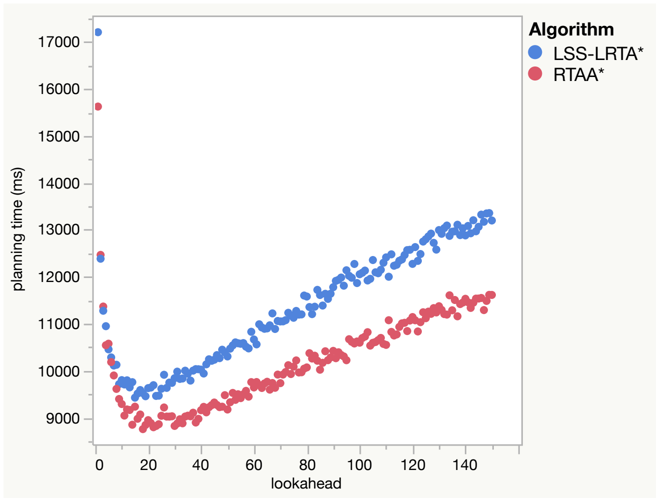
The plots of search episodes vs. lookahead in Figure 16 and the plots of differences between RTAA* and LSS-LRTA*, in Figure 17, show that, just as with mazes with known terrain, RTAA* doesn't outperform LSS-LRTA*, at least with the lookaheads I used. There does seem to be a trend in that direction for Figure 17a, however. It would seem that this is due to the fact that, as suggested earlier, RTAA* performs better relative to LSS-LRTA* when there are fewer obstacles.



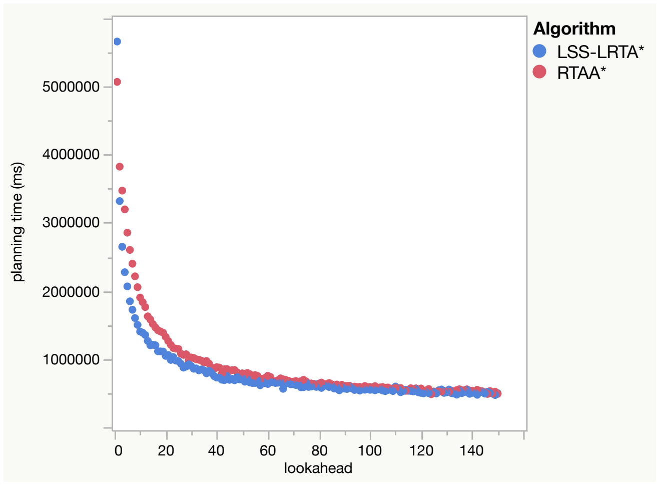
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%

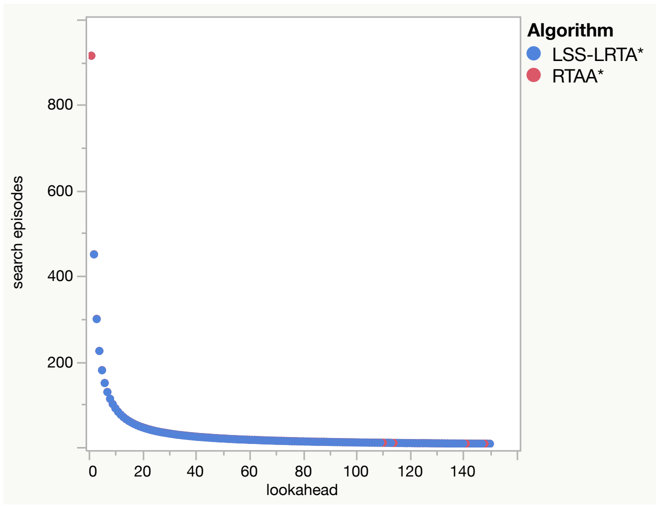


(d) random obstacles 40%

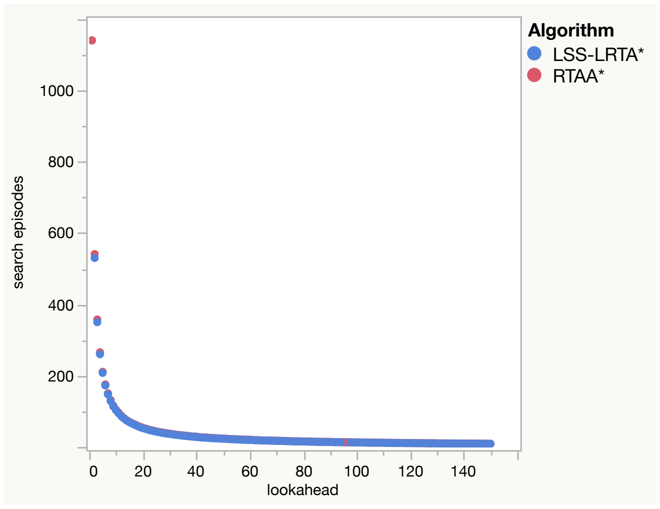
Figure 15:
Random Maps, Known Terrain: total planning time vs. Lookahead

The results shown in Figures 18 and 19 are consistent with the maze results: RTAA* tends to outperform LSS-LRTA* with larger lookaheads and lower obstacle densities.

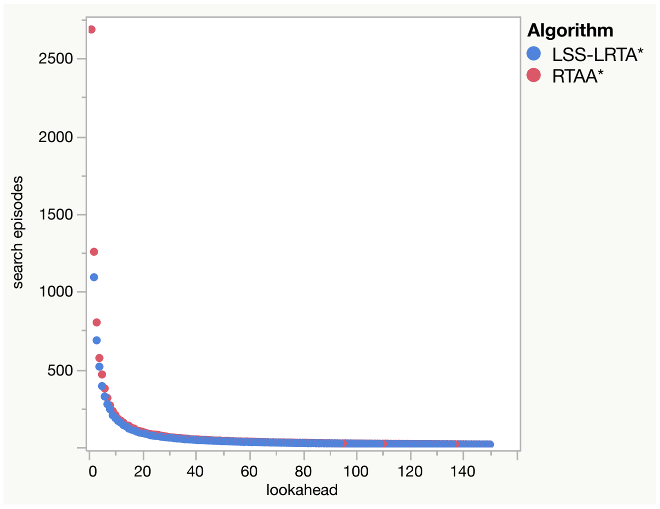
The plots in Figures 20 and 21, the latter of which used linear interpolation as described earlier to match data points, show that, for low-obstacle maps, there is a trend for RTAA* to outperform LSS-LRTA* with larger lookaheads, which is consistent with my other observations, namely that RTAA* performs better with fewer obstacles and with a longer time-limit.



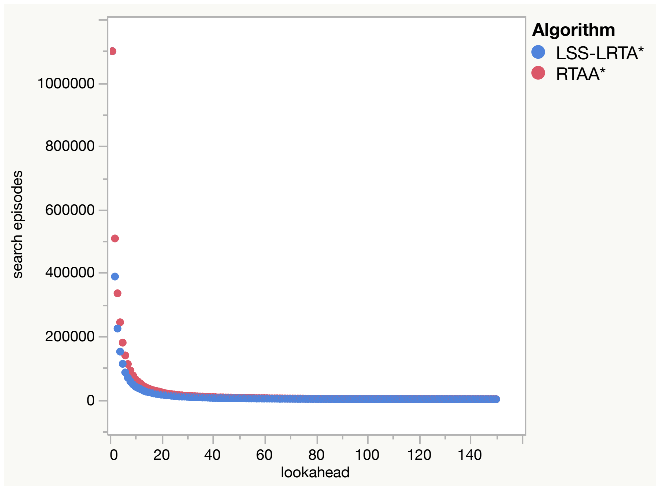
(a) random obstacles 10% (see Appendix A for log scale)



(b) random obstacles 20% (see Appendix A for log scale)

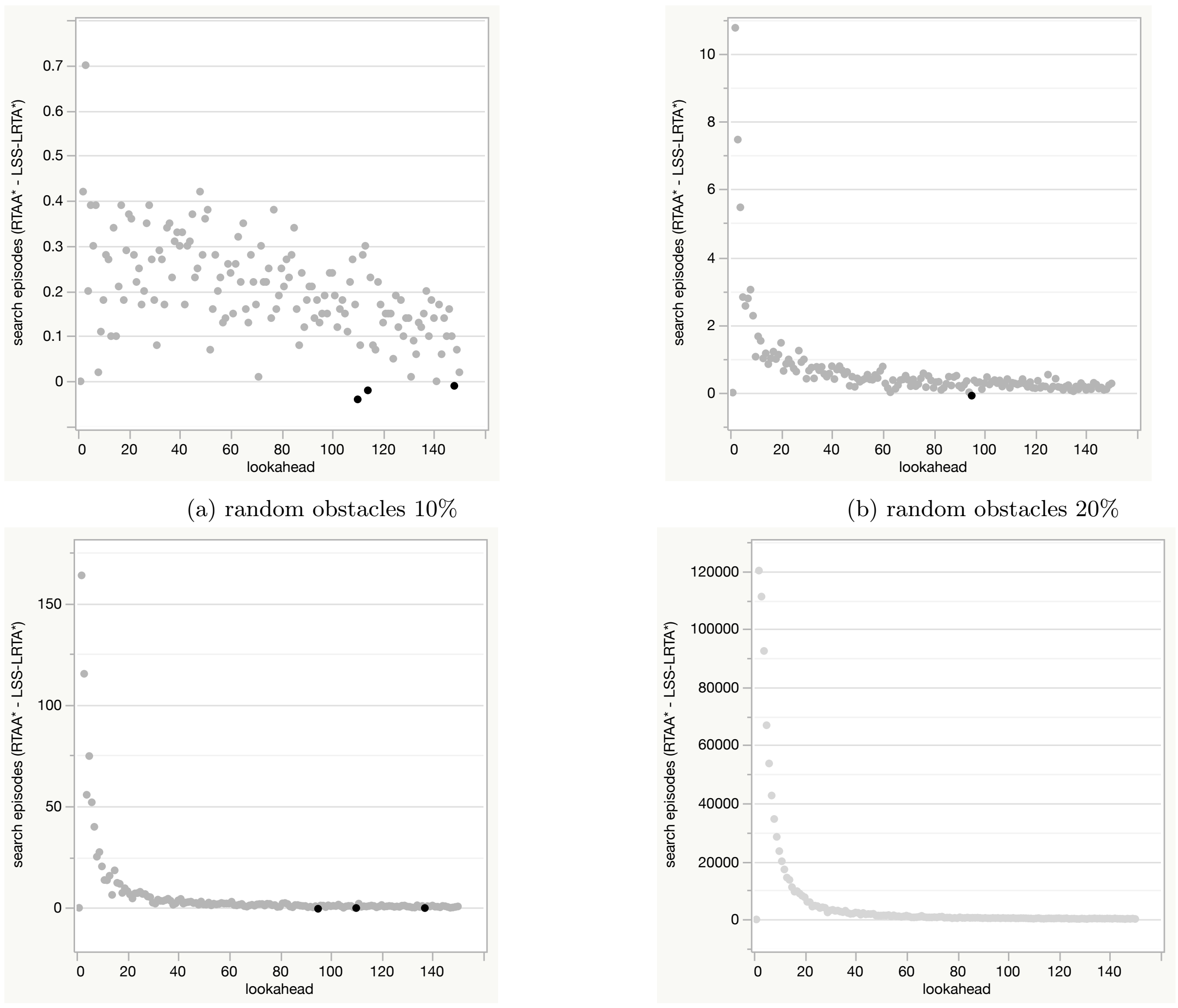


(c) random obstacles 30% (see Appendix A for log scale)



(d) random obstacles 40% (see Appendix A for log scale)

Figure 16:
Random Maps, Known Terrain: Search Episodes vs. Lookahead



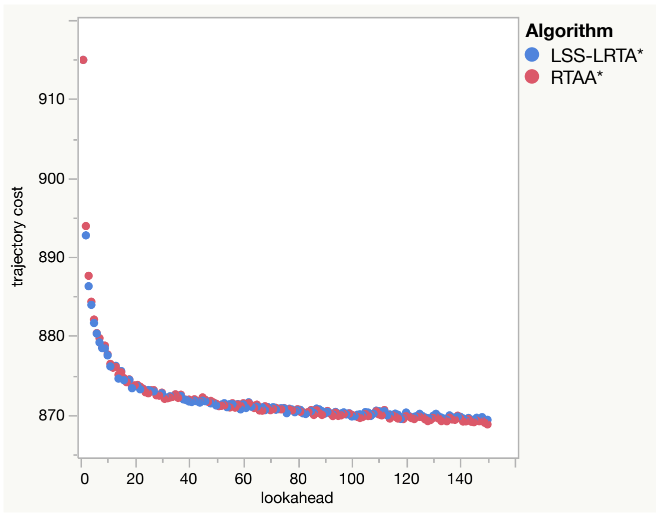
(a) random obstacles 10%

(b) random obstacles 20%

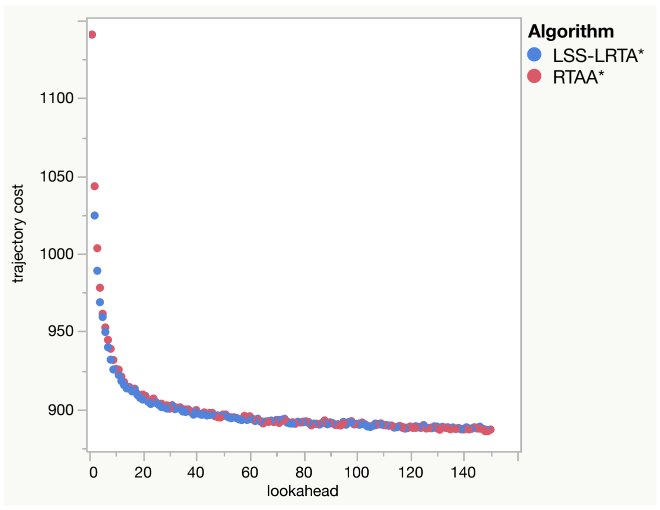
(c) random obstacles 30%

(d) random obstacles 40%

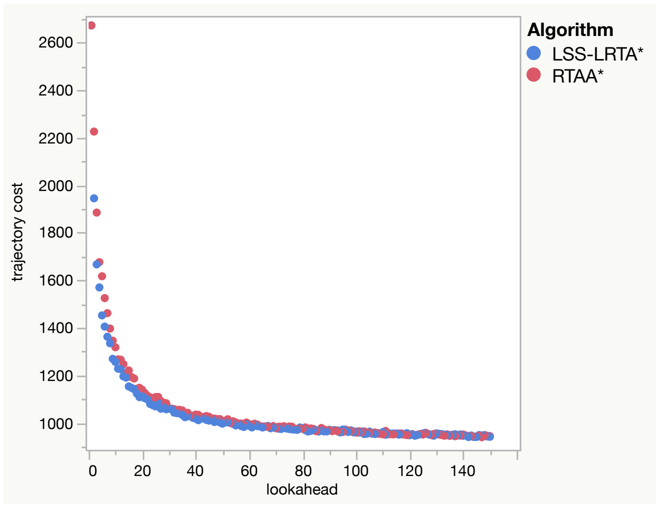
Figure 17:
Random Maps, Known Terrain: Search Episodes (RTAA* - LSS-LRTA*) vs. Lookahead



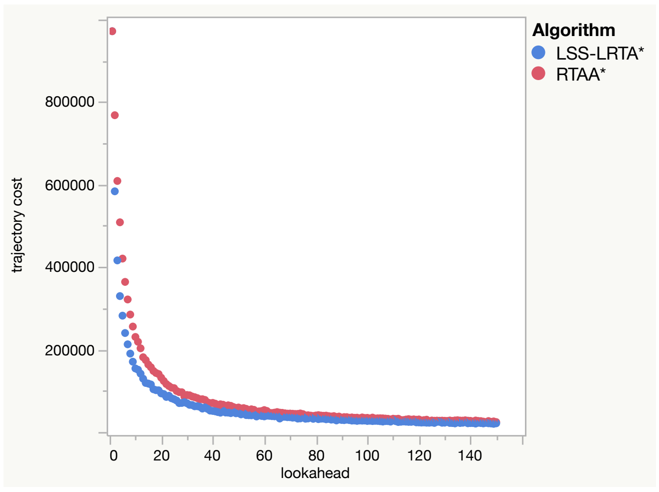
(a) random obstacles 10% (see Appendix A for log scale)



(b) random obstacles 20% (see Appendix A for log scale)

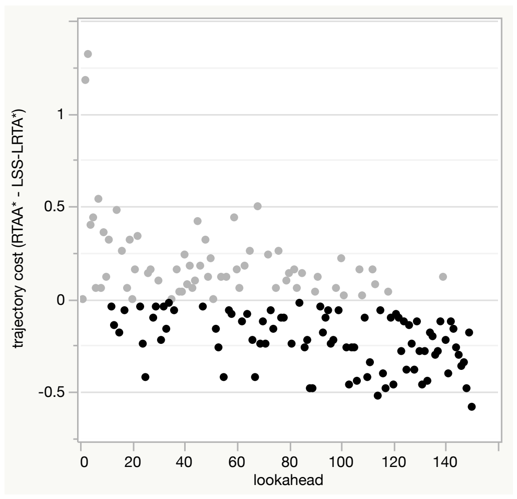


(c) random obstacles 30% (see Appendix A for log scale)

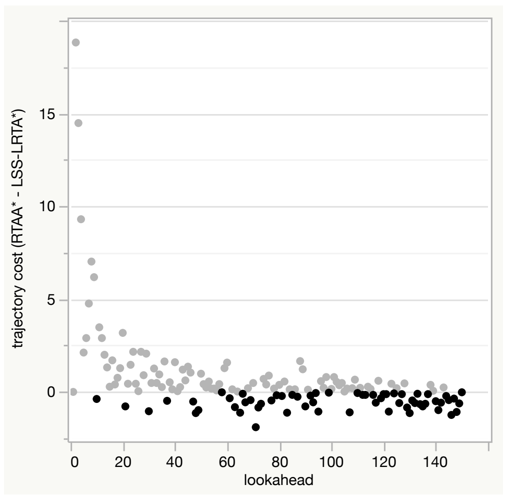


(d) random obstacles 40% (see Appendix A for log scale)

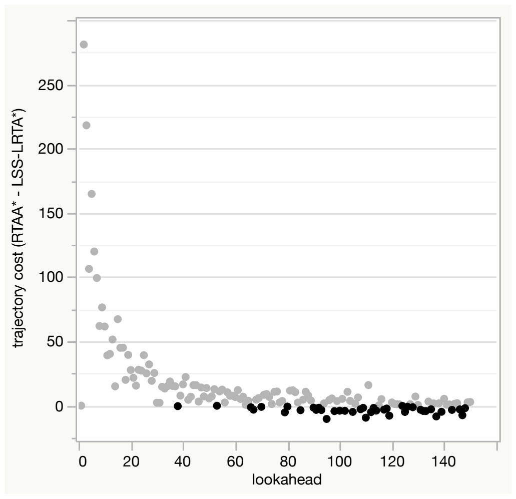
Figure 18:
Random Maps, Known Terrain: Trajectory Cost vs. Lookahead



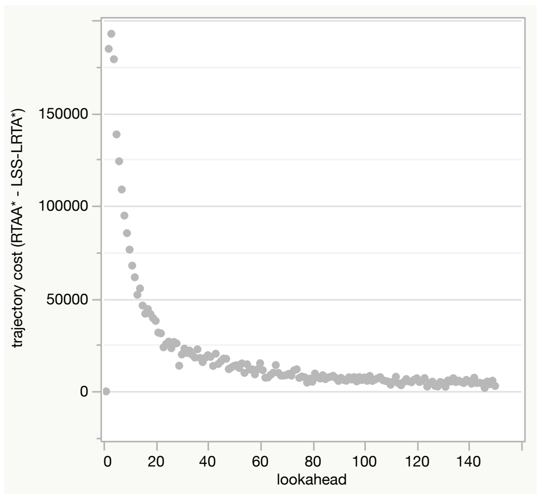
(a) random obstacles 10%



(b) random obstacles 20%

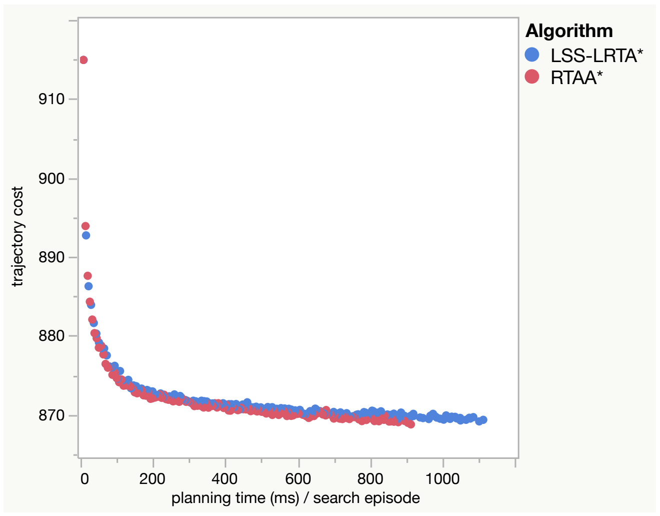


(c) random obstacles 30%

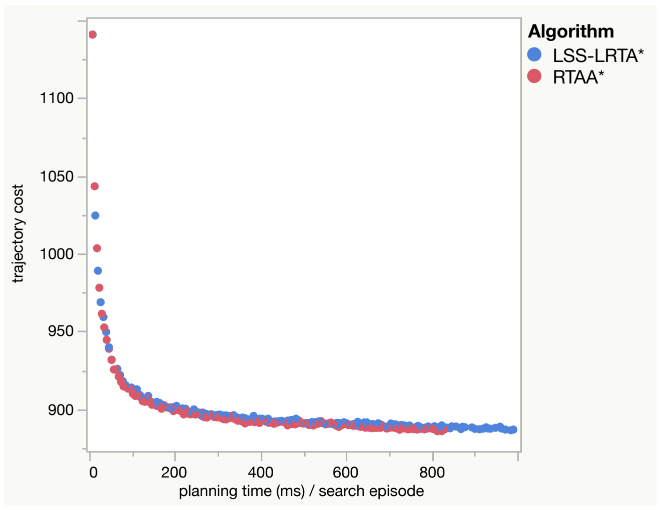


(d) random obstacles 40%

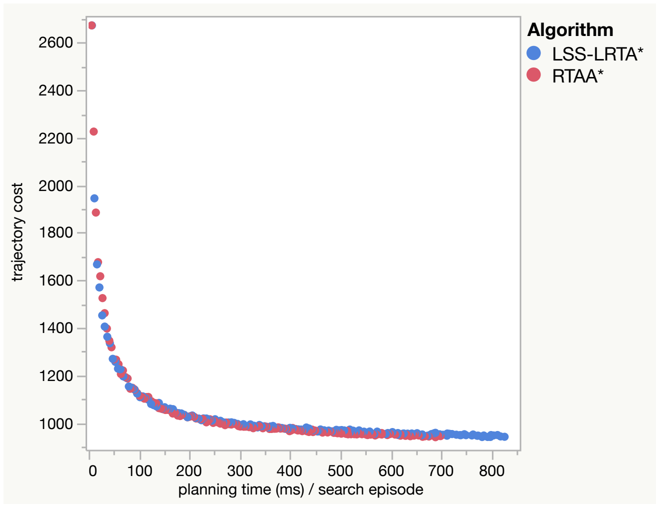
Figure 19:
Random Maps, Known Terrain: Trajectory Cost (RTAA* - LSS-LRTA*) vs. Lookahead



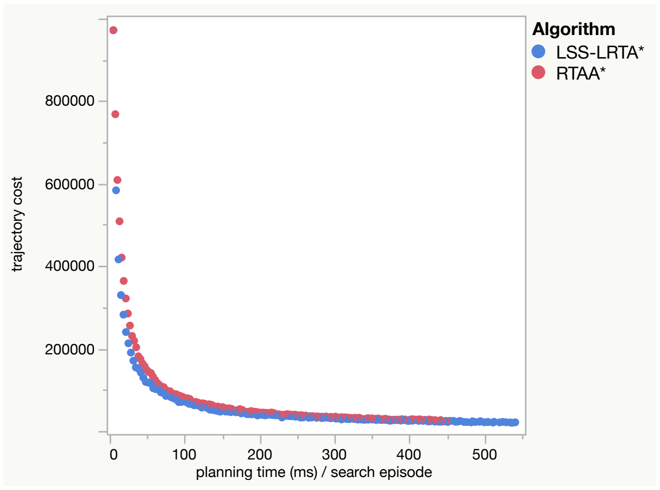
(a) random obstacles 10% (see Appendix A for log scale)



(b) random obstacles 20% (see Appendix A for log scale)

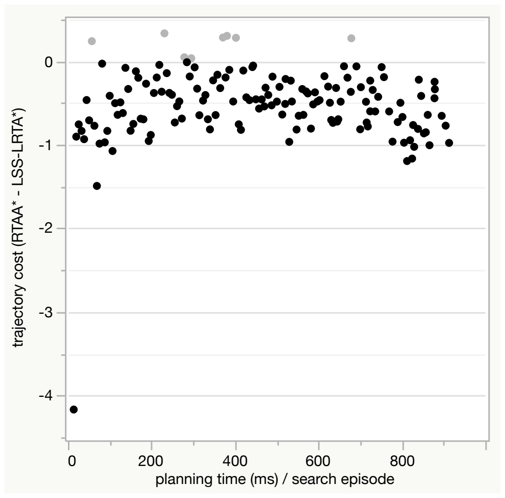


(c) random obstacles 30% (see Appendix A for log scale)

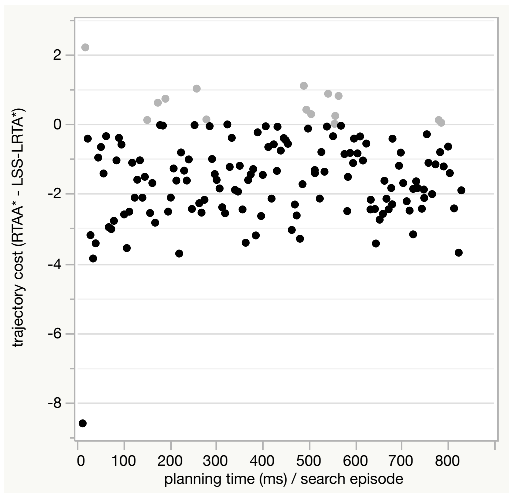


(d) random obstacles 40% (see Appendix A for log scale)

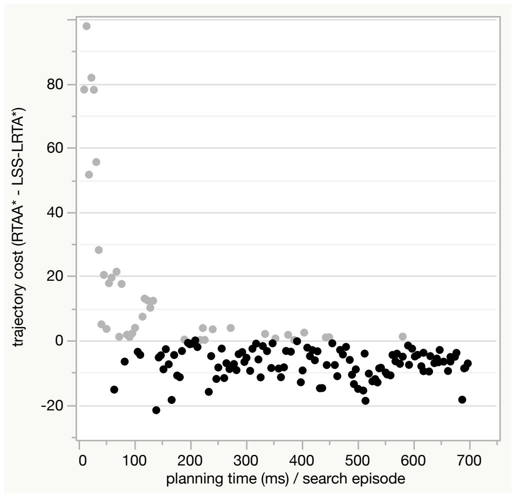
Figure 20:
Random Maps, Known Terrain: Trajectory Cost vs. Planning Time / Search Episode



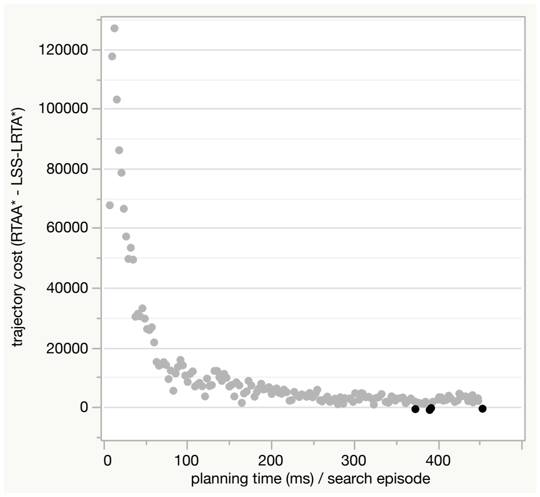
(a) random obstacles 10%



(b) random obstacles 20%



(c) random obstacles 30%



(d) random obstacles 40%

Figure 21:
Random Maps, Known Terrain: Trajectory Cost (RTAA* - LSS-LRTA*) vs. Planning Time / Search Episode

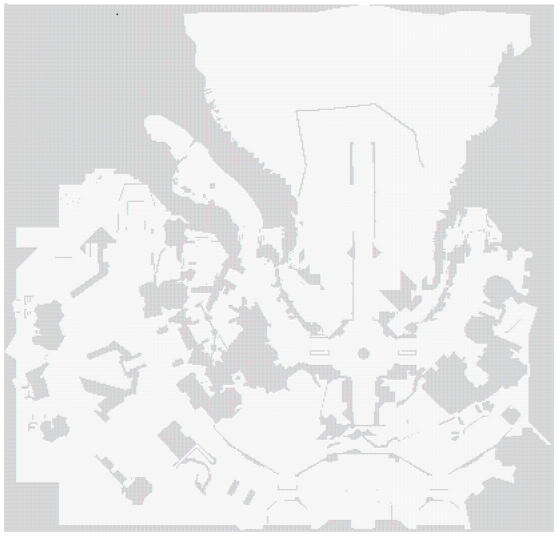


Figure 22: orz100d

2.1.4 Game Map

I chose a map from a library [3] of maps from the game Dragon Age: Origins, orz100d, which is shown in Figure 22. The library instances of the game map scenarios were not readable by my program, so I converted them.

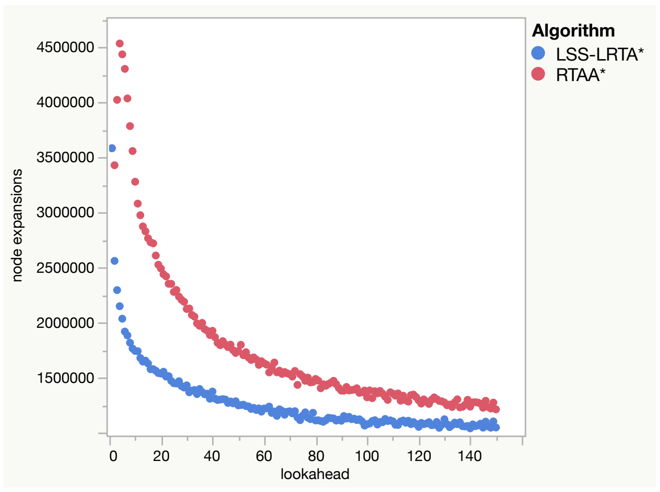
Scenarios with different start and end points are divided into buckets based on optimal solution length. I ran each algorithm on 100 scenarios, using lookaheads from 1 to 150, from the buckets with the longest optimal solution length. Unlike the maze domain or random domains, game maps, and orz100d in particular, contain regions with very many obstacles and regions with very few obstacles. Overall, 38% of the grid nodes are obstacles.

Unknown Terrain

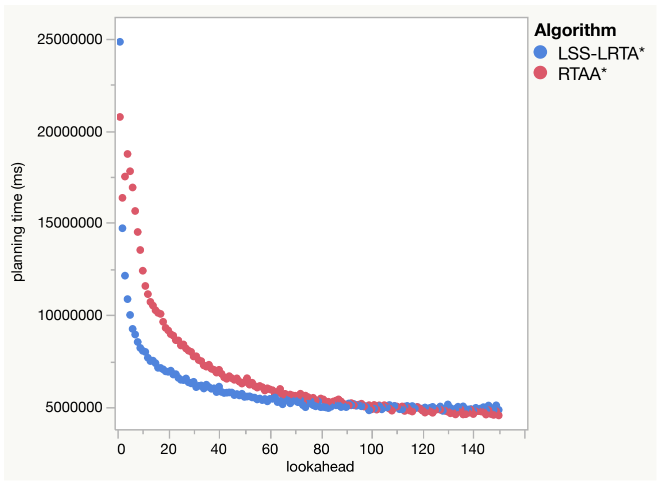
Results and Discussion

The plots for Figures 23a and 23b are notably different from their corresponding plots of node expansion vs. lookahead with unknown terrain in both random maps and mazes, more closely resembling the plots of node expansion vs. lookahead for high-percent-obstacle grids with known terrain. The reason why it differs isn't clear, but I can hazard a guess, which is based on the presumption that there are a large number of search episodes with LSS that are approximately circular. Let l = lookahead, a = # of action executions, e = # of search episodes, t = trajectory cost, and n = # of node epxansions. Let c_1, c_2, and c_3 be some constants.

If the LSS is approximately circular, then the area of the circle would approximate l and the radius would approximate $\frac{a}{e}$, so we would get:



(a) orz100d, unknown terrain: node expansions vs lookahead



(b) orz100d, unknown terrain: planning time vs lookahead

Figure 23

$$\frac{a}{e} = \sqrt{\frac{l}{\pi}}$$

This is broadly consistent with the shape of the plot in Figure 23c, which is overlaid with the regression line $a/e = 1.3 + 0.6\sqrt{l}$ ($r^2 = 0.98$).

If we assume, as we have previously, that t is inversely proportional to l, then we can say:

$$t = \frac{c_2}{c_3 \times l}$$

Clearly,

$$e = \frac{t}{\frac{a}{e}}$$

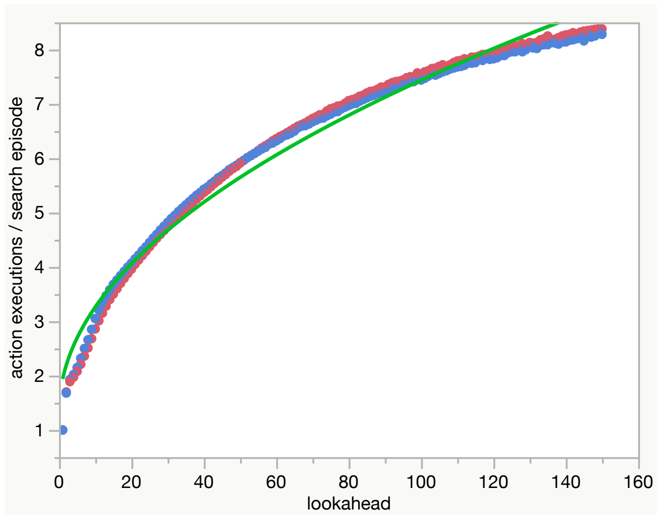
and

$$n = e \times l$$

Putting everything in terms of l, we get:

$$n = \frac{\frac{c_2}{c_3 \times l}}{\sqrt{\frac{l}{\pi}}} \times l$$

and reducing, we get



(c) orz100d, unknown terrain: action execution per search episode vs lookahead

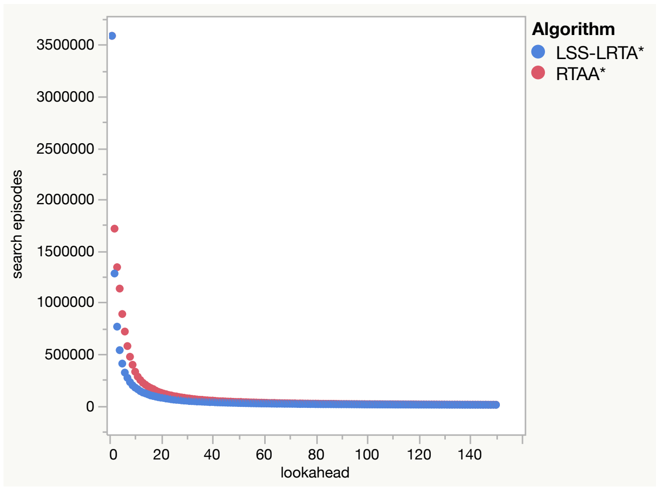
Figure 23: (continued)

$$n = \frac{\frac{c_2}{c_3}}{\sqrt{\frac{l}{\pi}}}$$

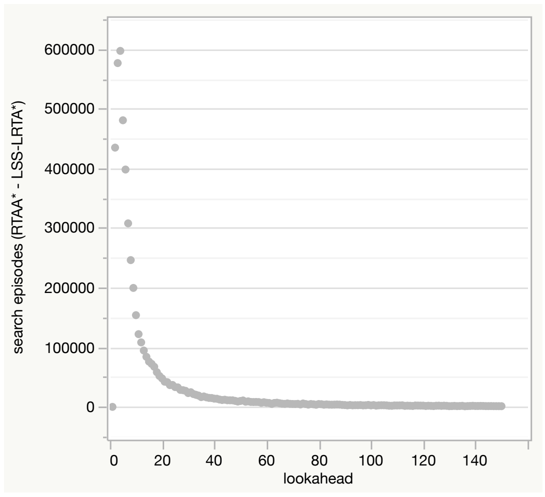
which is consistent with the shape of the plot in Figure 23a.

By making a few reasonable assumptions, we can explain, broadly, the structure of the plots.

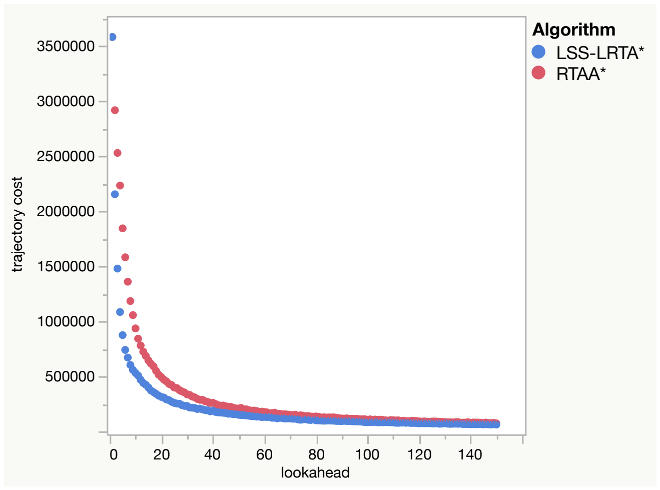
The plots in Figures 23d, 23e, 23f and 23g all demonstrate that LSS-LRTA* outperforms RTAA* for all lookaheads. However, comparing on the essential metric of trajectory cost vs. planning time per search episode in Figures 23h and 23i, we see once again the familiar pattern that RTAA* outperforms LSS-LRTA* at large lookaheads. There is some difficulty making inferences about the causes of the behavior of an irregular game map such as this without more detailed analysis.



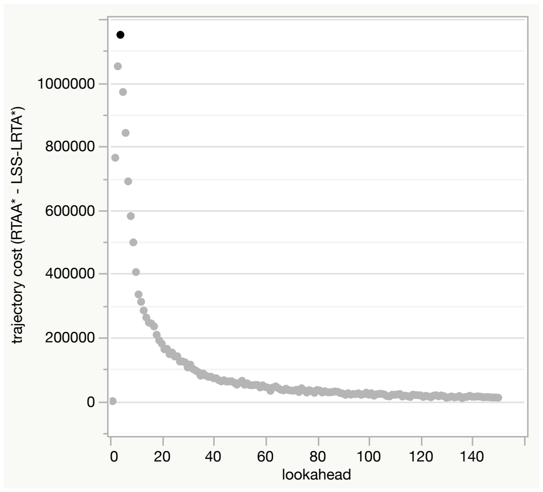
(d) orz100d, unknown terrain: search episodes vs lookahead



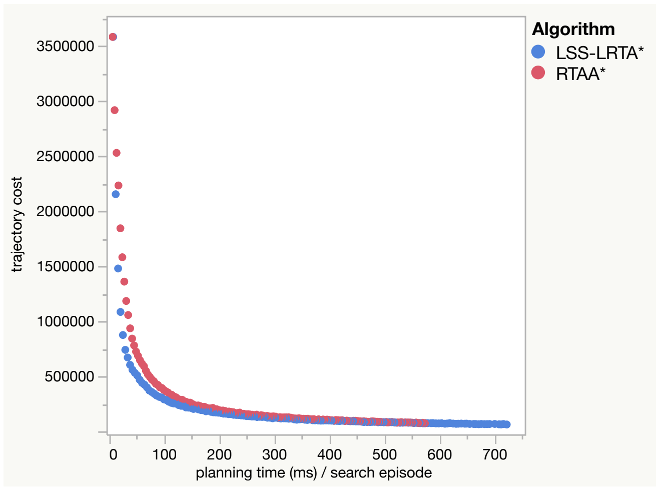
(e) orz100d, unknown terrain: search episodes (RTAA* - LSS-LRTA*) vs lookahead



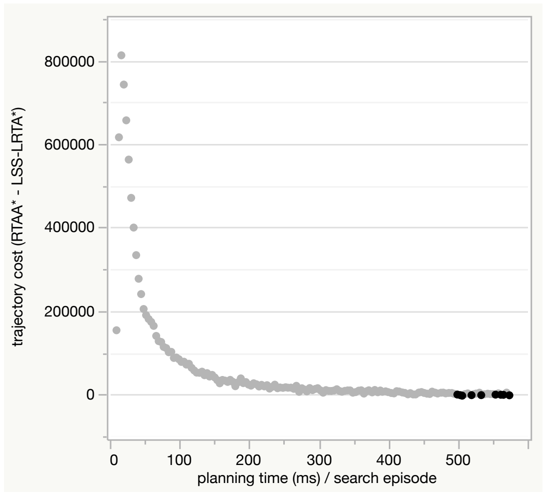
(f) orz100d, unknown terrain: trajectory cost vs lookahead



(g) orz100d, unknown terrain: trajectory cost (RTAA* - LSS-LRTA*) vs lookahead



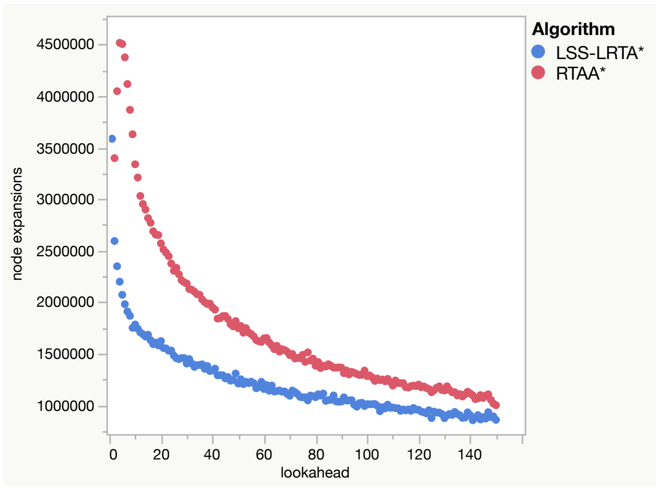
(h) orz100d, unknown terrain: trajectory cost vs planning time / search episodes



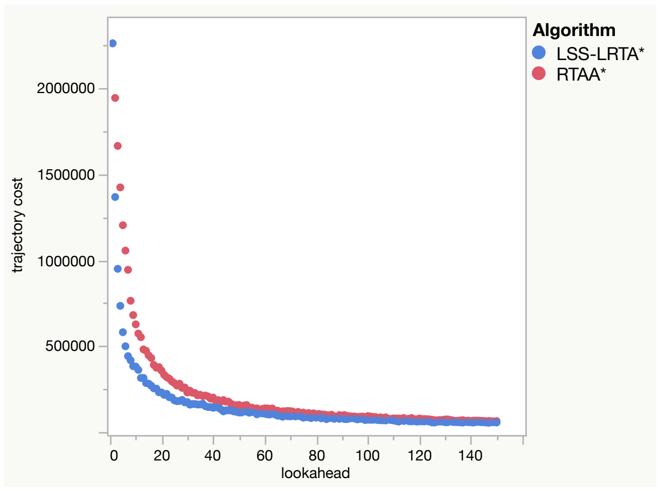
(i) orz100d, unknown terrain: trajectory cost (RTAA* - LSS-LRTA*) vs planning time / search episodes

Figure 23: (continued)

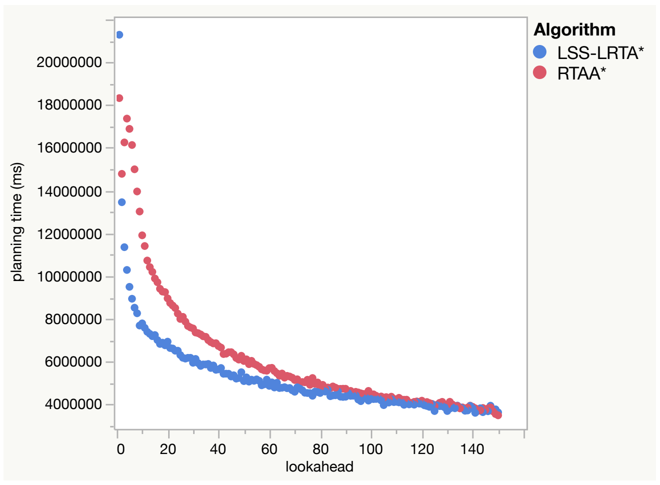
Known Terrain



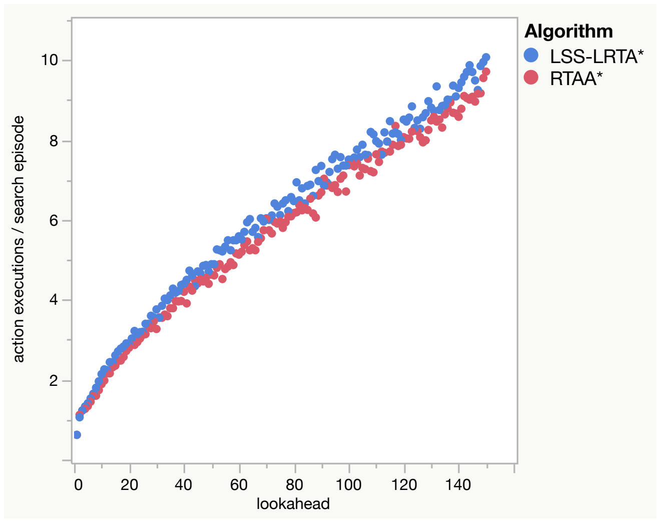
(a) orz100d, known terrain: node expansions vs lookahead



(b) orz100d, known terrain: trajectory cost vs planning time per search episode



(c) orz100d, known terrain: search search episodes vs lookahead



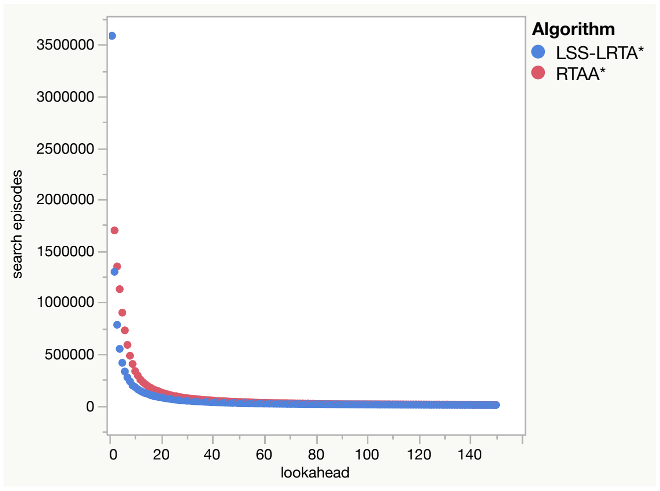
(d) orz100d, known terrain: action execution per search episode vs lookahead

Figure 24

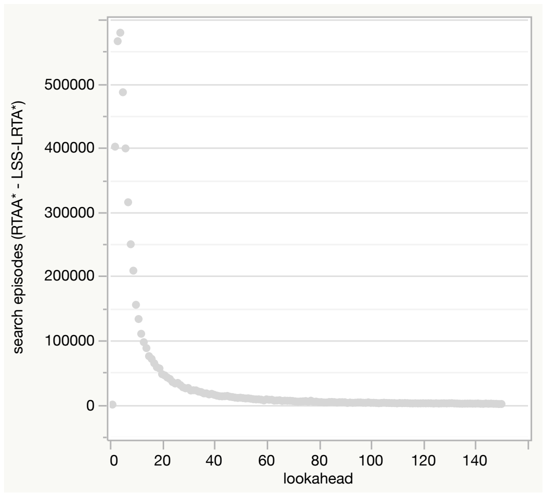
Results and Discussion

Not many of the plots in this section are remarkable at first glance. They are very similar to the plots from random maps with obstacle density 40%. The difference is quite subtle, because the orz100d domain instance is very irregular. This suggests that obstacle density can be a greater determinant of the overall behavior of the algorithms than the specific placement of the obstacles.

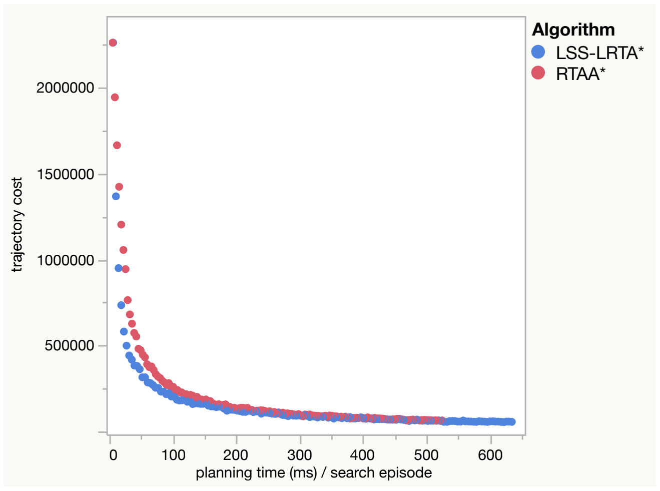
Comparing on the essential metric of trajectory cost vs. planning time per search episode, we see that once again, RTAA* outperforms LSS-LRTA* at large lookaheads.



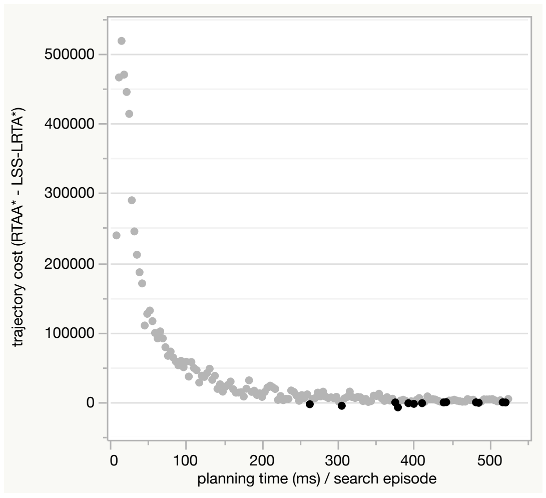
(e) orz100d, known terrain: search episodes vs lookahead



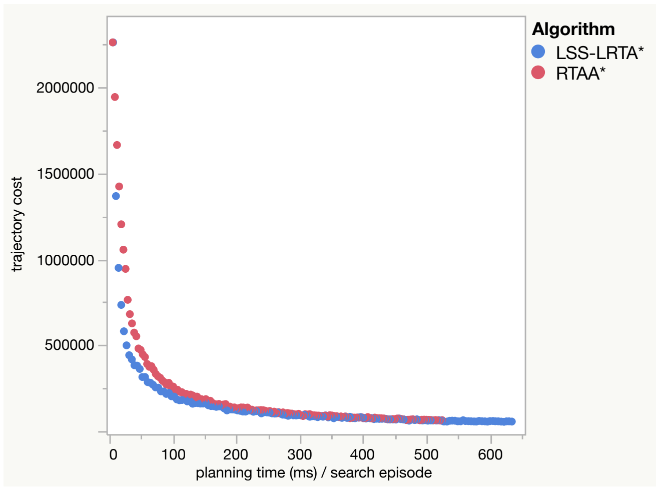
(f) orz100d, known terrain: search episodes (RTAA* - LSS-LRTA*) vs lookahead



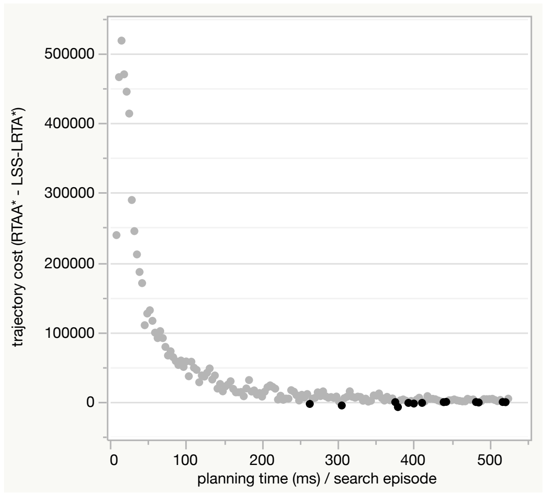
(g) orz100d, known terrain: trajectory cost vs planning time / search episodes



(h) orz100d, known terrain: trajectory cost (RTAA* - LSS-LRTA*) vs planning time / search episodes

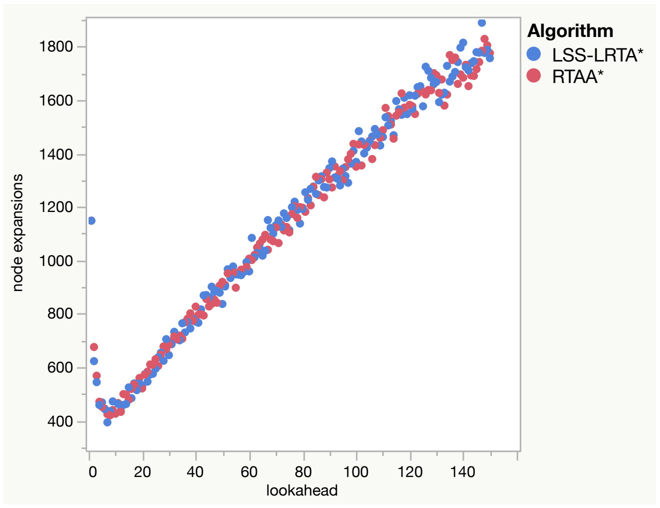


(i) orz100d: trajectory cost vs planning time / search episodes

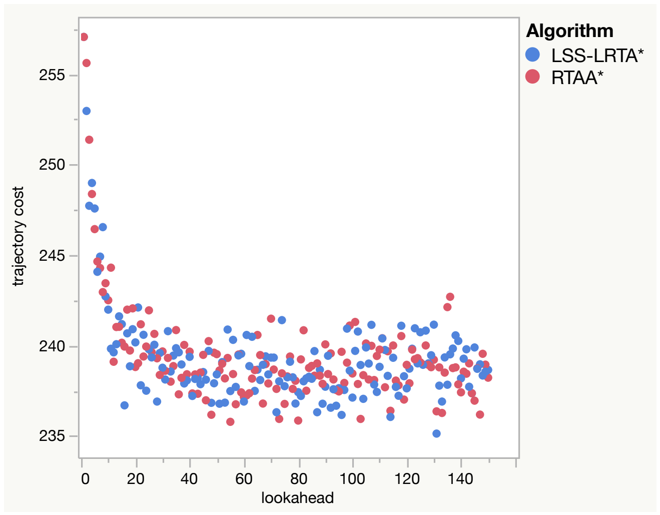


(j) orz100d: trajectory cost (RTAA* - LSS-LRTA*) vs planning time / search episodes

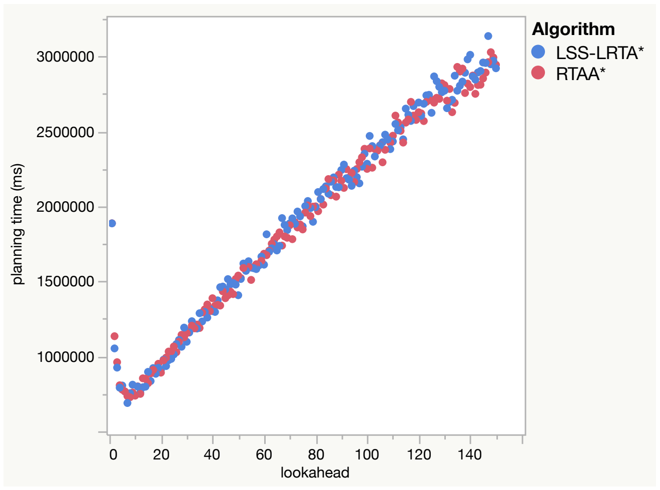
Figure 24: (continued)



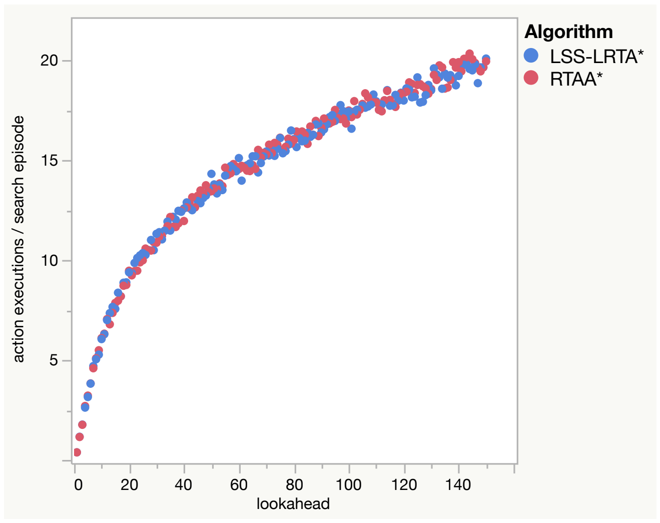
(a) orz100d: node expansions vs lookahead



(b) orz100d: trajectory cost vs lookahead



(c) orz100d: total planning time (ms) vs lookahead



(d) orz100d: action execution per search episode vs lookahead

Figure 25:
traffic, obstacles 50%

2.2 Traffic

The traffic domain is modeled on the once popular video game Frogger. In the version used here, a 100×100 grid is filled 50% with obstacles with start and goal states in each corner. Each obstacle has a direction and velocity. Time is measured in "ticks". Each action execution of the agent and each move of an obstacle takes place during one tick. The agent may move north, east, south, or west, or may remain in the same location. Each state is defined by a particular configuration of obstacles and location of the agent at a particular tick. If, in the course of an A* search, no successors are found, the agent is returned to the start state. The heuristic used is Manhattan distance.

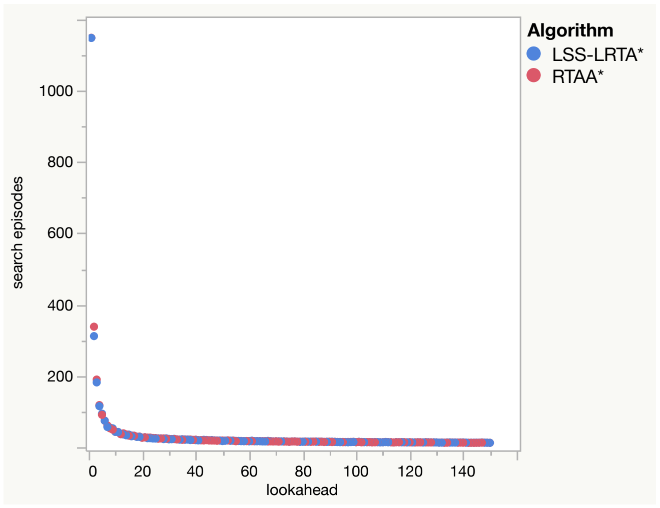
The domain is fully observable.

Implementation

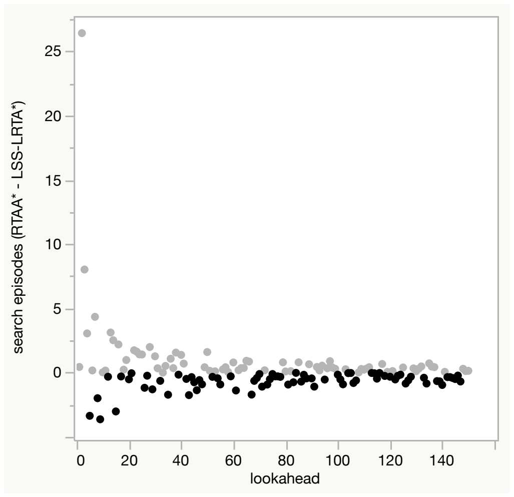
The only interesting thing regarding implementation of Traffic is how states are generated and accessed. In the original code, each state was generated on demand. This took too long, so I modified the implementation to use a hash table for accessing previously generated states.

Results and Discussion

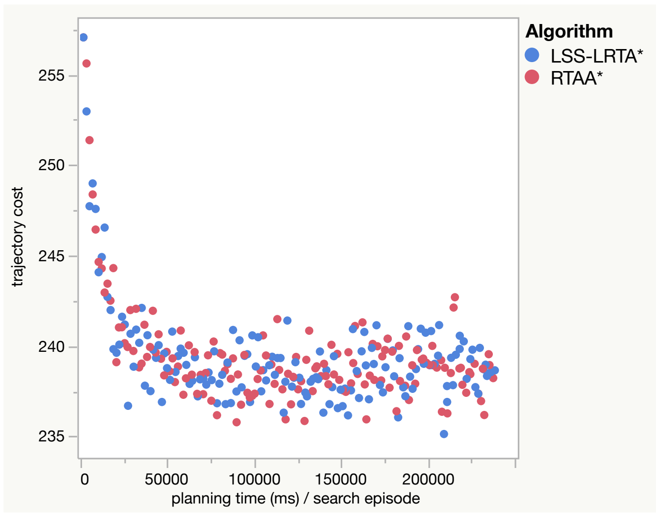
It is easy to see similar patterns in Traffic as in the other domains that I have evaluated: the J-shape in Figures 25a and 25c suggests a diminishing marginal benefit of increasing lookaheads for reasons earlier explained, the inverse relationship between number of search episodes and trajectory cost vs. lookahead, in Figures 25e and 25b respectively, and the square root shape of Figure 25d, suggesting large numbers of local search spaces with a circular shape. The most notable thing about the Traffic domain, however, is the fact that neither algorithm outperforms the other, certainly not as shown by the most important plots, shown in Figures 25g and 25h. This would seem to be due to the randomness of restarts. Although there may be a subtle difference between the algorithms during most of the execution, it is overwhelmed by the noise created by random restarts.



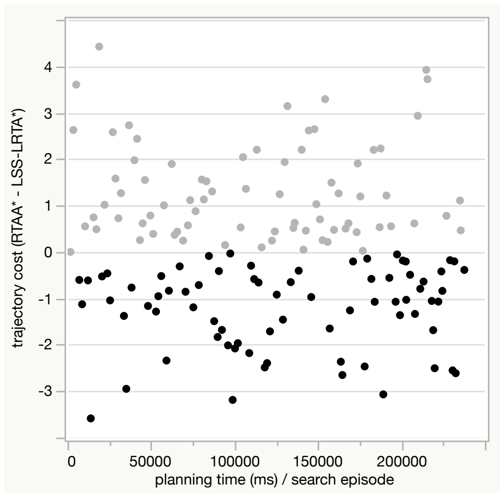
(e) traffic: search episodes vs lookahead



(f) traffic: search episodes (RTAA* - LSS-LRTA*) vs lookahead



(g) traffic: trajectory cost vs planning time / search episodes



(h) traffic: trajectory cost (RTAA* - LSS-LRTA*) vs planning time / search episodes

Figure 25:
traffic, obstacles 50%
(continued)

3 Conclusions

There are several tentative conclusions that can be drawn from this research. First, for grid path-finding, with the increasing lookahead and longer time-limits, relative to LSS-LRTA*, RTAA* performs better and better, eventually outperforming LSS-LRTA* in terms of trajectory cost at the largest lookaheads and longest time-limits, in contradiction to the claims of Koenig & Likhachev. Second, with a lower density of obstacles, RTAA* performs better relative to LSS-LRTA*. It is impossible to tell whether RTAA* or LSS-LRTA* performs better in the Traffic domain, and this seems to be because the occurrence of restarts is basically random and hides any subtle differences between the two.

4 Further Work

New domains to run the algorithms on might include the Sliding Tile Puzzle, Racetrack, and Traffic with different obstacle densities. Domains with non-uniform action costs may also prove to be of interest. A Grand Unified Theory of Heuristic Search would be grand. In this paper, I made many observations which may or may not prove to be generally correct, for example I hypothesized that the shape of the node expansions vs. lookahead plot for orz100d could be related to the shape of the local search space. A detailed analysis with novel metrics could strengthen or weaken this hypothesis. The back-of-an-envelope equations I presented used basic algebra, but there are other mathematical tools which I did not have time to fully apply. For example, I would have liked to examine the relationship between obstacle density and the shape of the plot of actions per search episode vs lookahead using a probability model. Furthermore, by understanding the behavior of real-time heuristic search algorithms generally, we will better be able to understand how and why they differ. Modeling the behavior of these algorithms mathematically will in the long term, I think, prove fruitful in guiding further research.

References

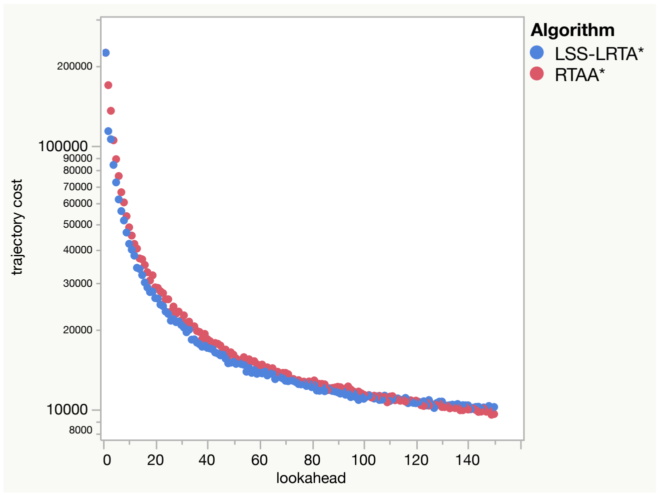
[1] Sven Koenig and Xiaoxun Sun. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems*, 18(3):313–341, June 2009.

[2] Sven Koenig and Maxim Likhachev. Real-time adaptive a*. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 281–288, New York, NY, USA, 2006. ACM.

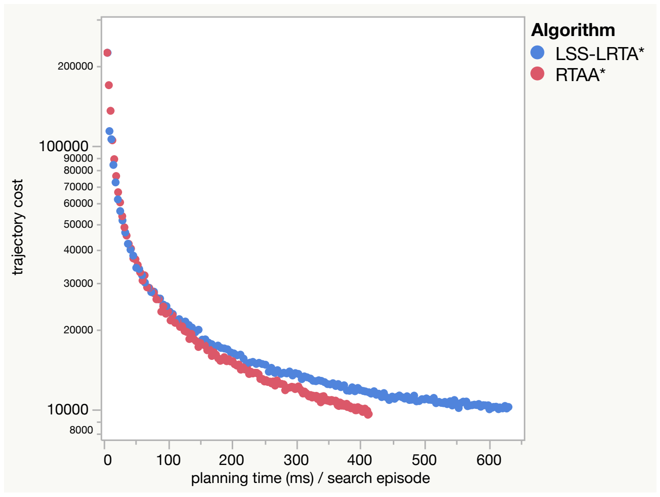
[3] N. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012.

Appendix A $\log_{10}$ Scale Plots

Mazes: Unknown Terrain

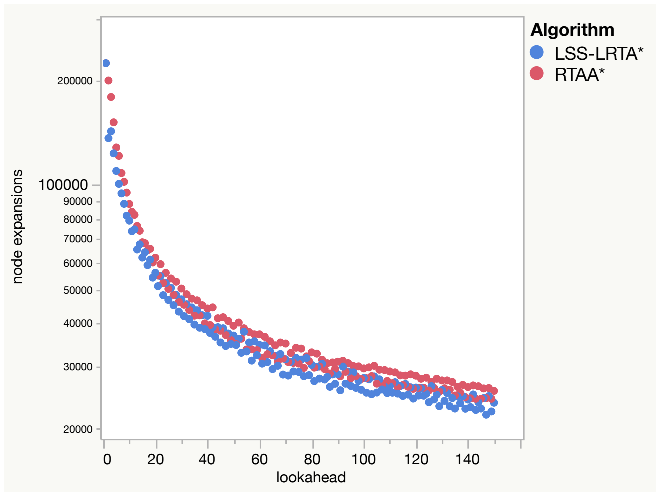


trajectory cost vs lookahead

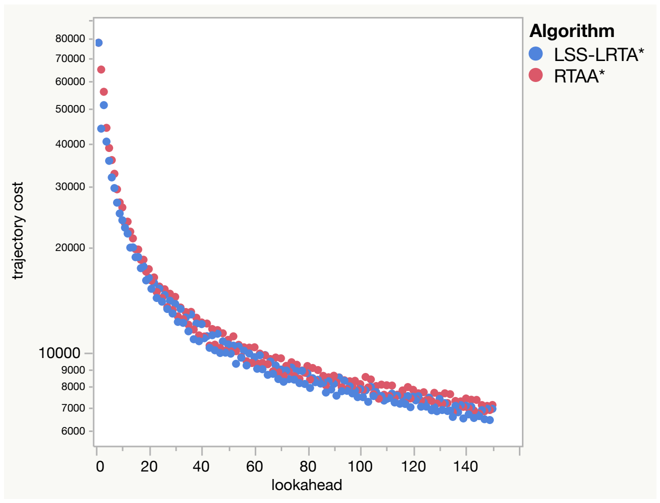


trajectory cost vs planning time per search episode (

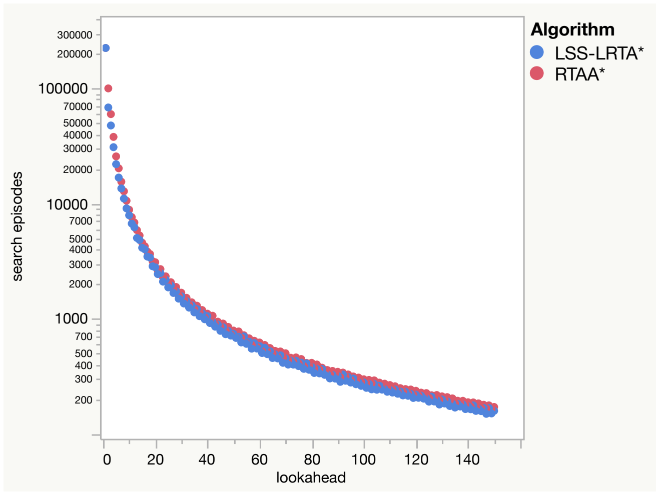
Mazes: Known Terrain



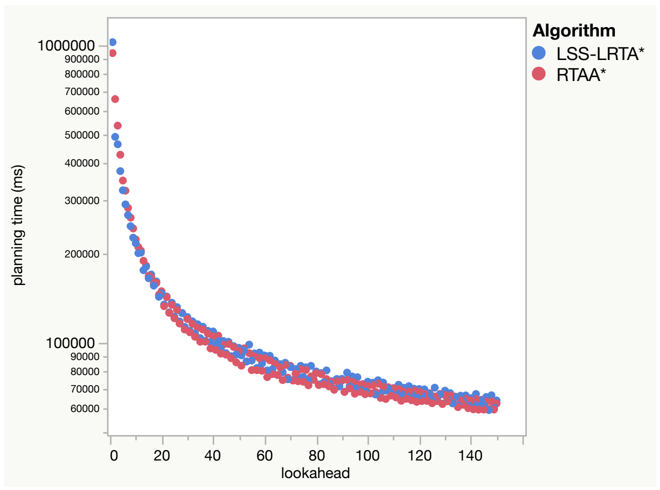
node expansions vs lookahead



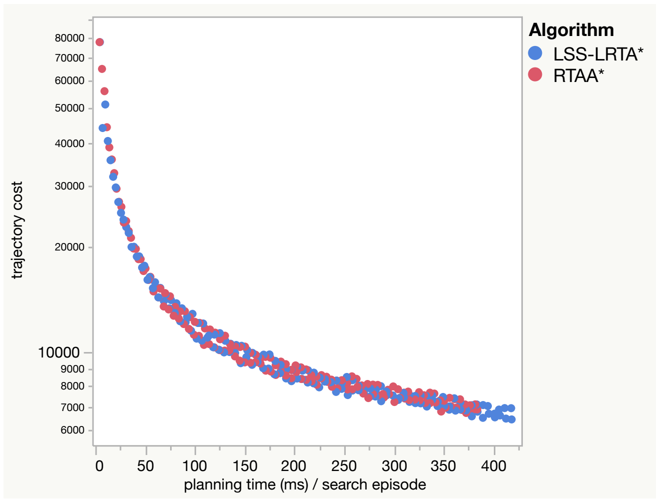
trajectory cost vs lookahead



search episodes vs lookahead

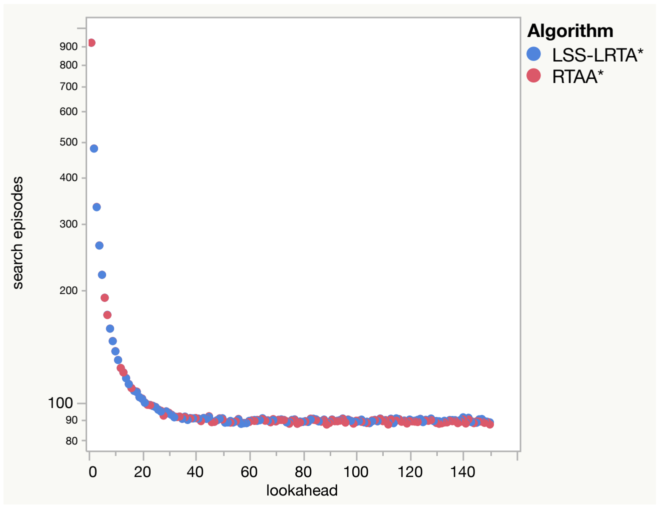


planning time (ms) vs lookahead

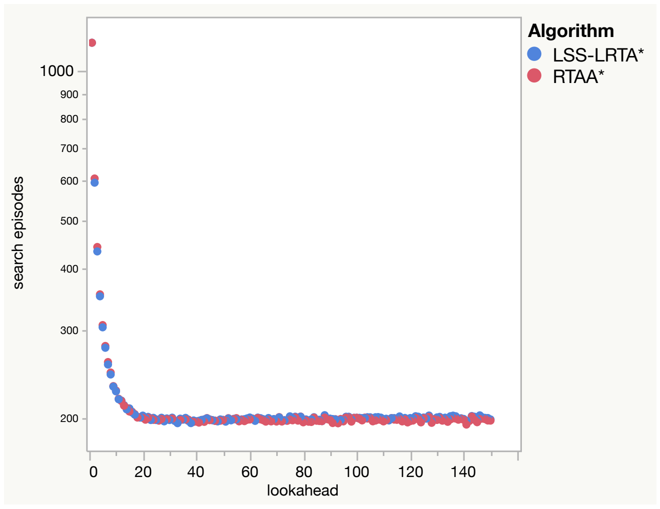


trajectory cost vs planning time (ms) / search episode

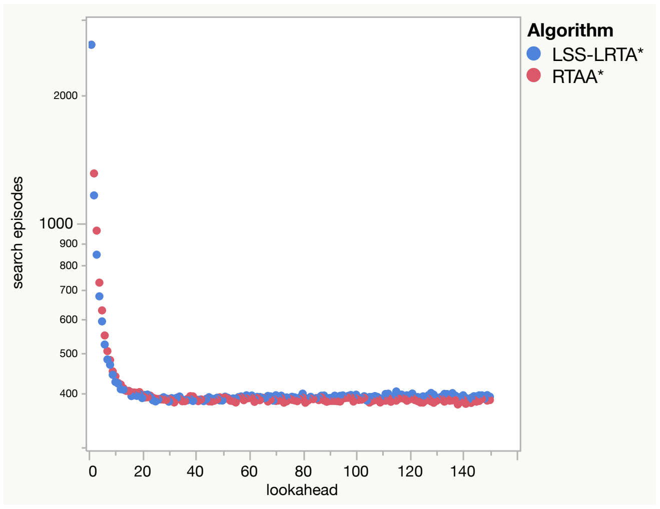
Random Maps: Unknown Terrain



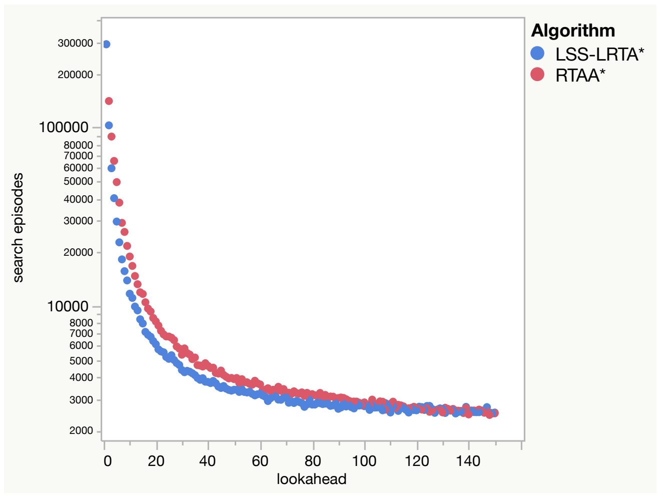
search episodes vs. lookahead: obstacles 10%



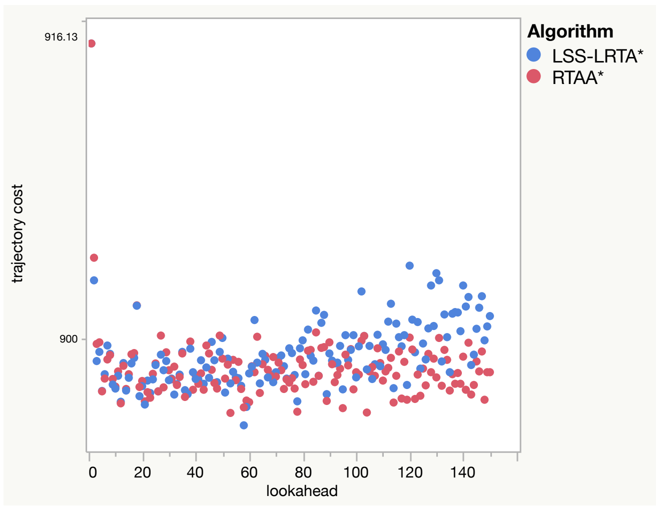
search episodes vs. lookahead: obstacles 20%



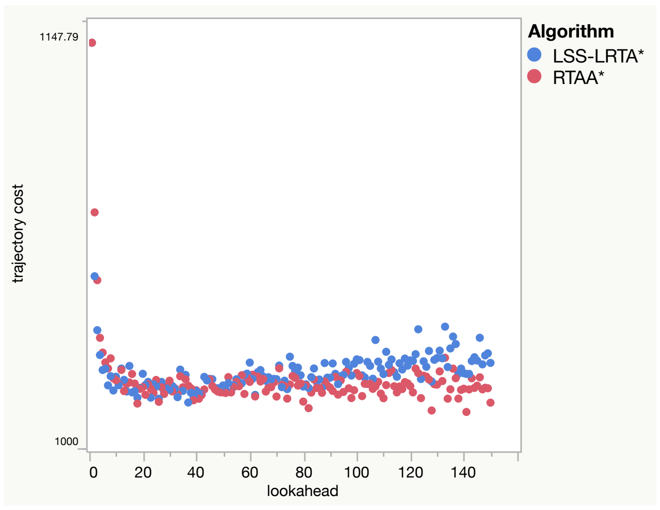
search episodes vs lookahead: obstacles 30%



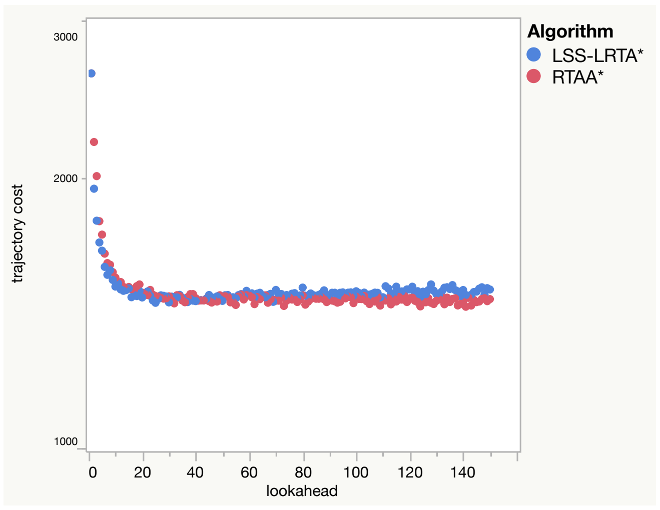
search episodes vs lookahead: obstacles 40%



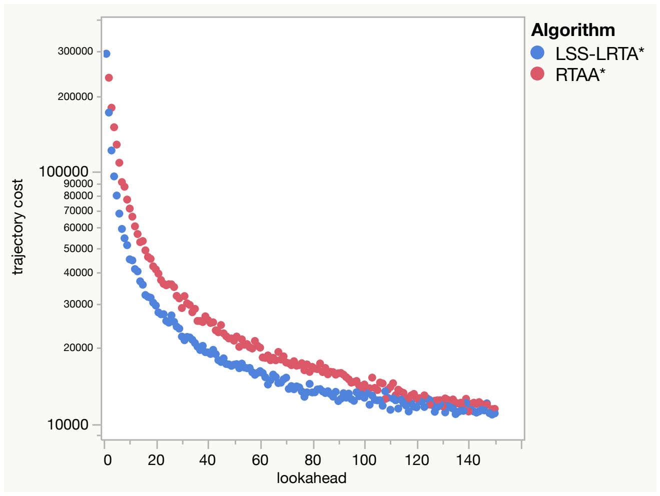
trajectory cost vs lookahead: obstacles 10%



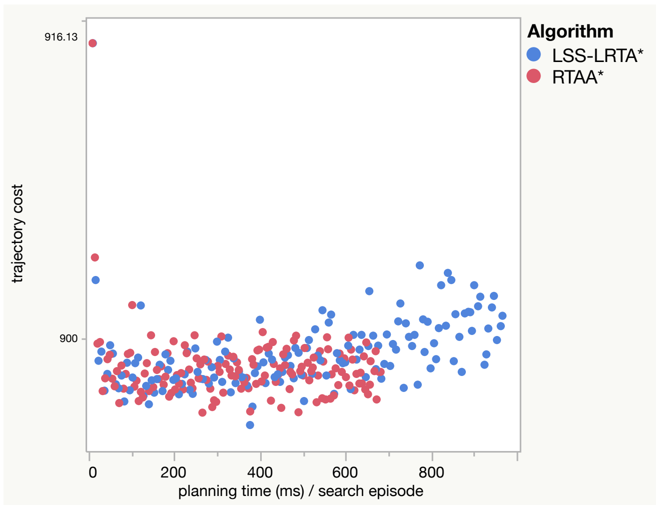
trajectory cost vs lookahead: obstacles 20%



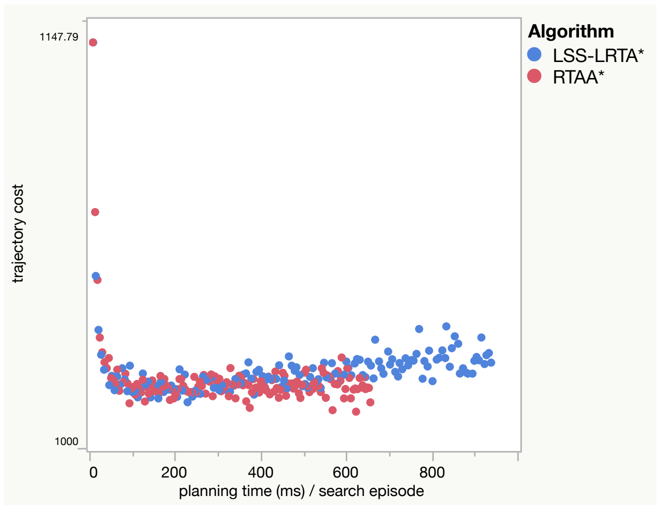
trajectory cost vs lookahead: obstacles 30%



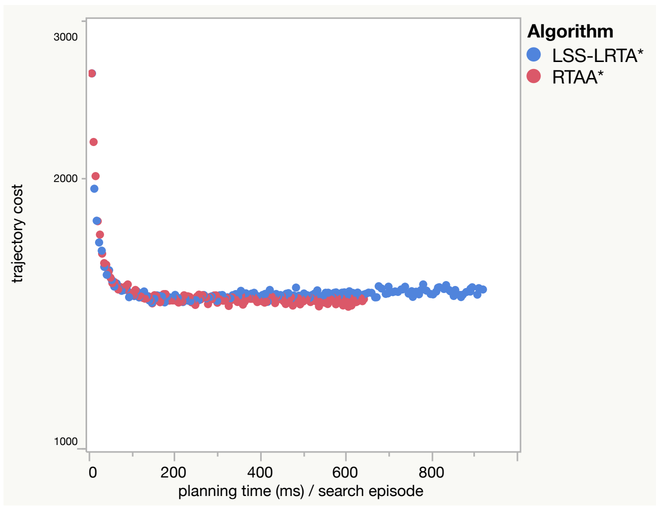
trajectory cost vs lookahead: obstacles 40%



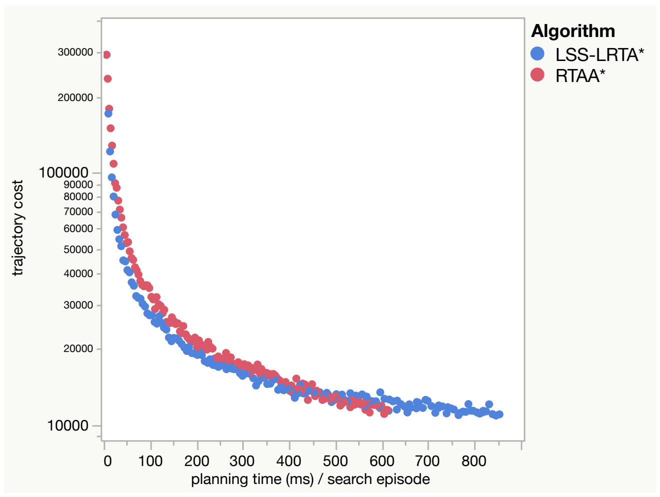
trajectory cost vs planning time / search episode: obstacles 10%



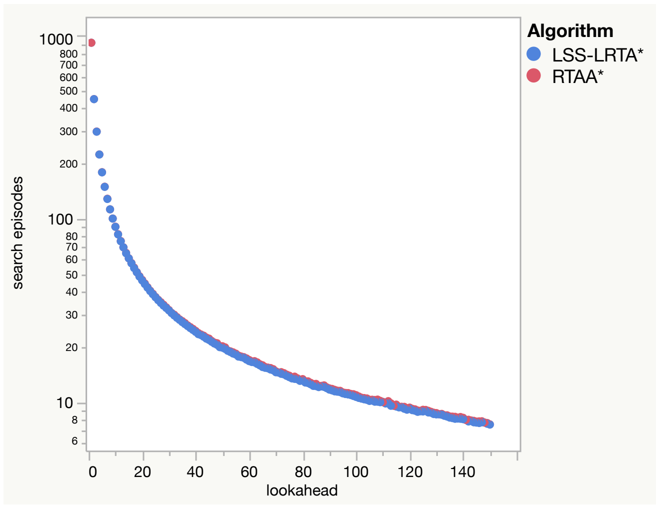
(a) trajectory cost vs planning time / search episode: obstacles 20%



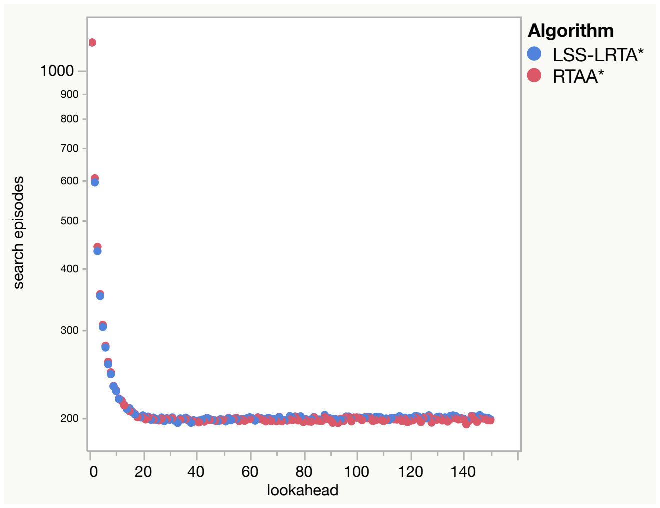
trajectory cost vs planning time / search episode: obstacles 30%



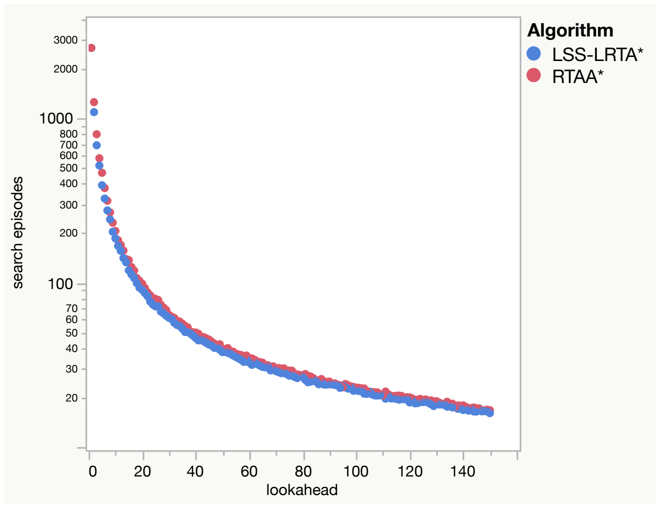
(b) trajectory cost vs planning time / search episode: obstacles 40%



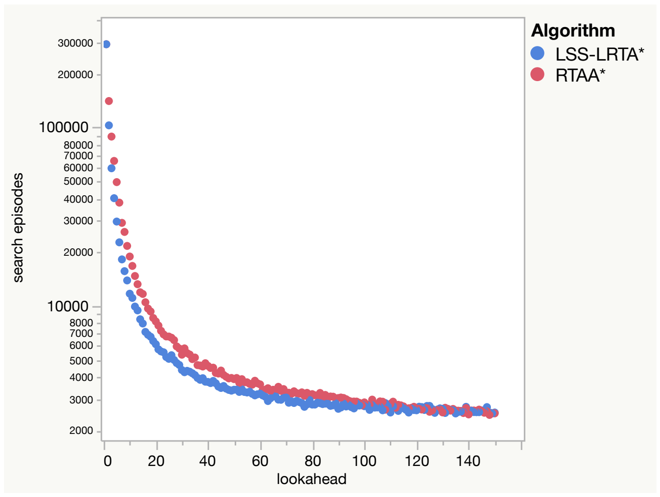
search episodes vs. lookahead: obstacles 10%



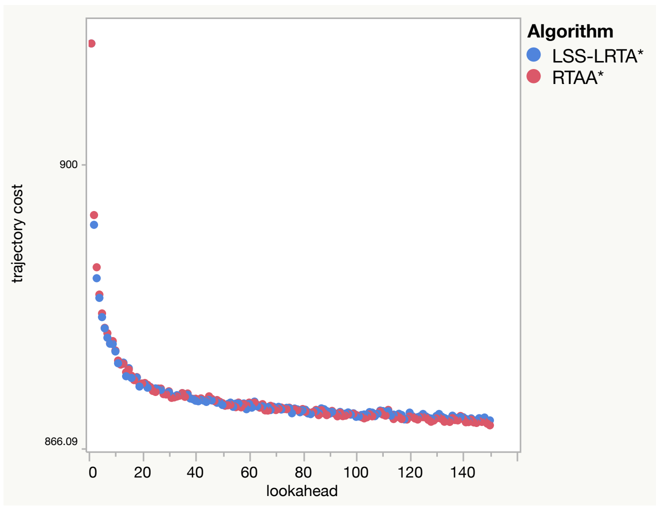
search episodes vs. lookahead: obstacles 20%



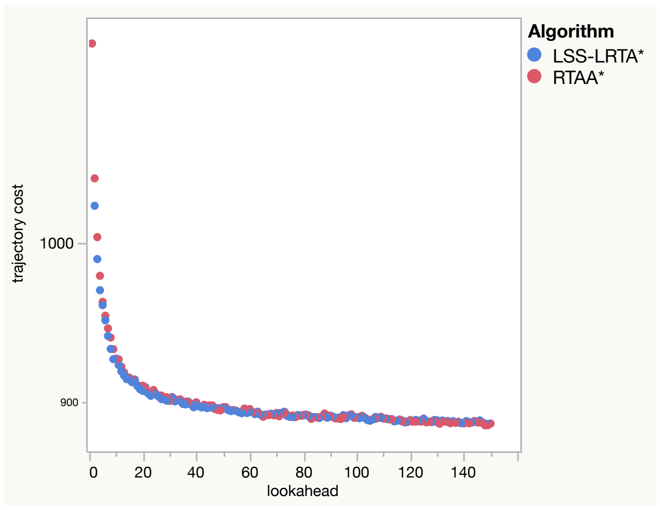
search episodes vs lookahead: obstacles 30%



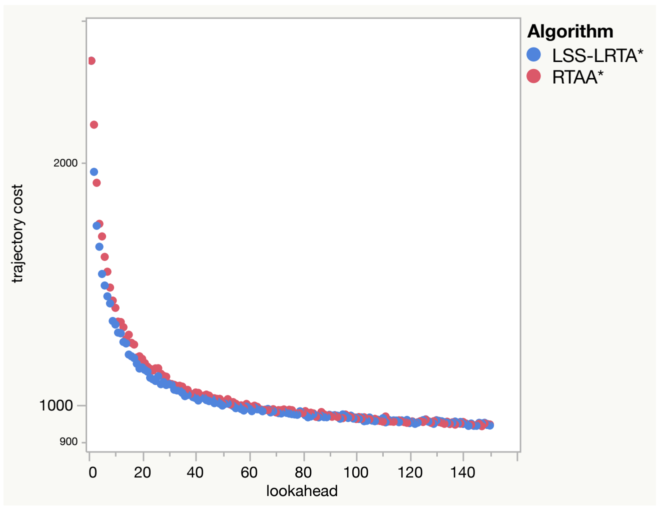
search episodes vs lookahead: obstacles 40%



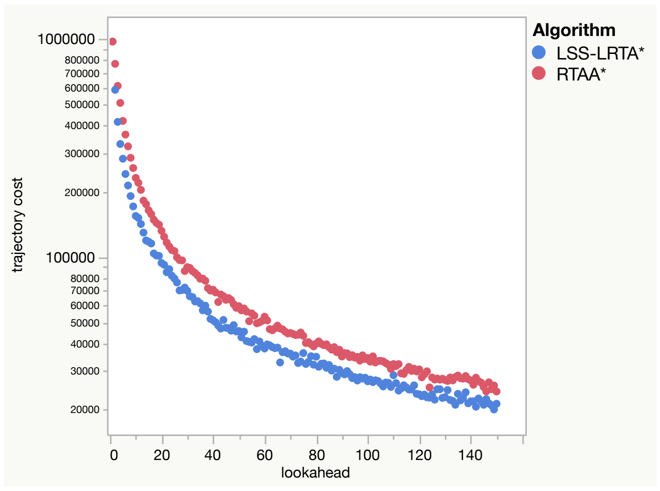
trajectory cost vs lookahead: obstacles 10%



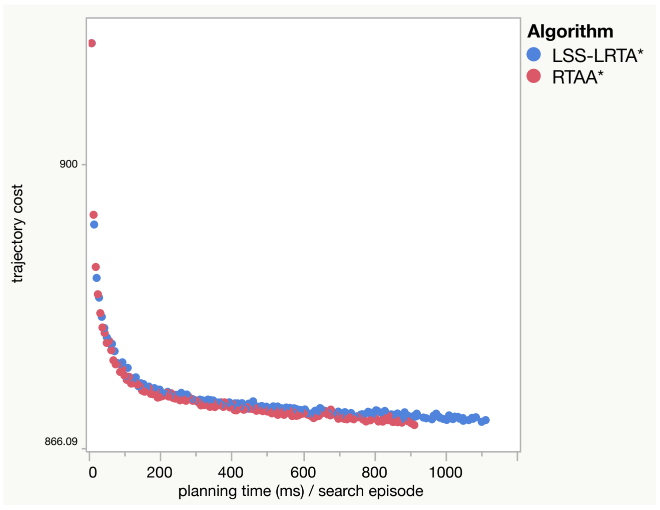
trajectory cost vs lookahead: obstacles 20%



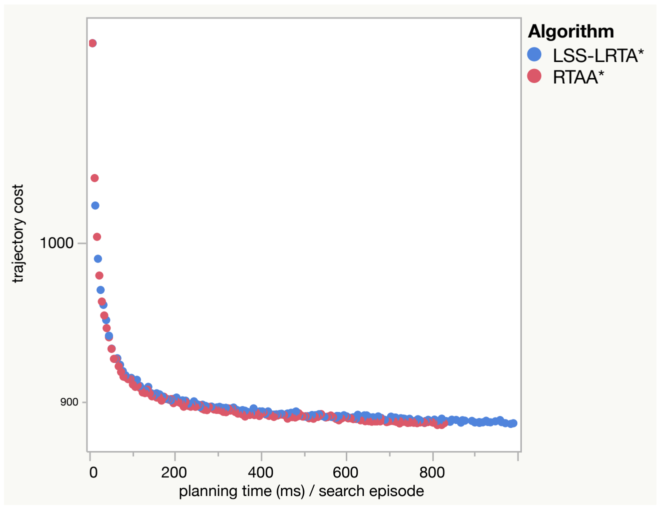
trajectory cost vs lookahead: obstacles 30%



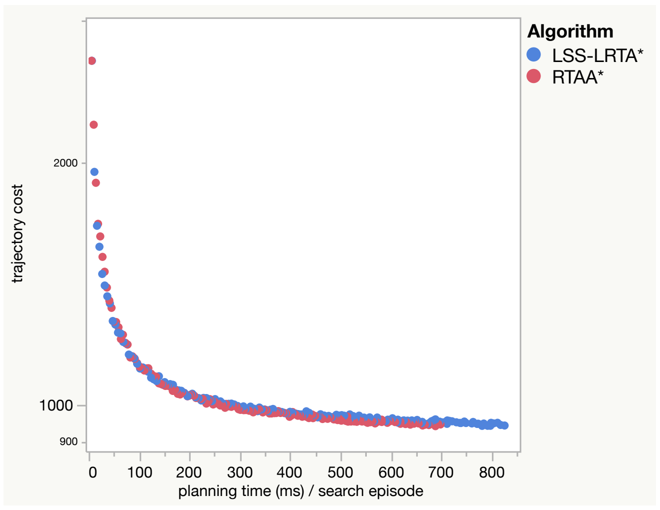
trajectory cost vs lookahead: obstacles 40%



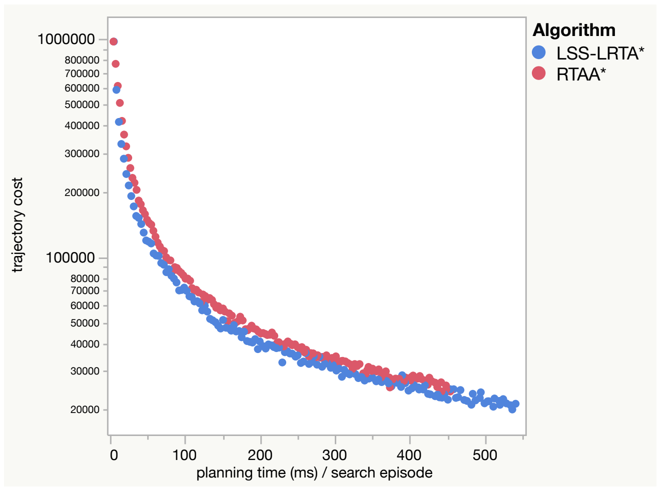
trajectory cost vs planning time / search episode: obstacles 10%



(a) trajectory cost vs planning time / search episode: obstacles 20%



trajectory cost vs planning time / search episode: obstacles 30%



(b) trajectory cost vs planning time / search episode: obstacles 40%