AI Project 3

Jennifer Borucki

November 27, 2017

1 A* Search

1.1 Introduction

There are three informed search algorithms: uniform-cost search, A*, and weighted A*. In order to test the algorithm there were 5 generated 120x160 8 neighbor grids each with 10 different solvable start-goal pairs for a total of 50 distinct grids. The grid visualizations are implemented using JavaFx. Black cells are blocked cells, white cells are unblocked cells, green cells are the highways, gray cells are the hard to traverse, red cell is the start cell, and the blue cell is the goal cell. To see the path once the program is up hit the A key and it will highlight cells in pink to show the path. To see expanded cells hit the E key and the cyan highlighted cells are all the cells that have been expanded.

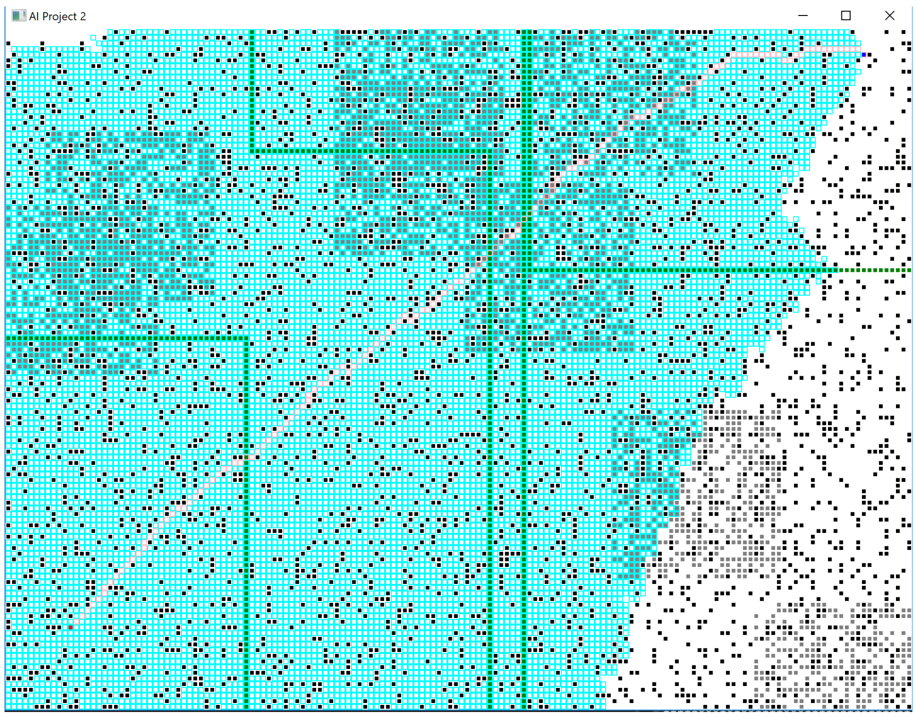


Figure 1: Manhattan Distance with a weight of 0.25

1.2 Algorithm Optimizations

Some optimizations implemented are utilizing Java's Priority Queue class with a specified comparator class. Another one is using a Tile object which holds most of the data. So instead of traversing through a 2D array it just traverses through a 1D array. Once we have the node getting the distance, finding neighbors, checking the type of cell, etc. makes it easily accessible.

1.3 Heuristic Selection

The first heuristic is the Euclidean distance formula divided by 4. The 4 was included to ensure admissibility due to the presence of highways, in order to hasten travel by a factor of 4. This provided the best possible admissible and consistent heuristic for the unweighted A*.

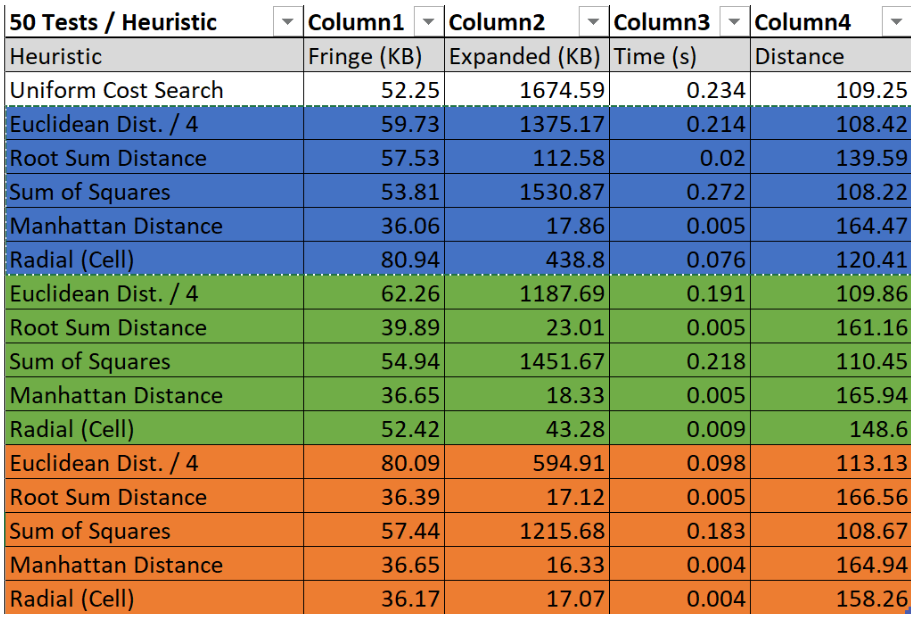
The second heuristic is the Manhattan distance, a alternative to the Euclidean.

The third heuristic is the square root of the Manhattan distance, which will weight nodes closer to the goal stronger than the normal Manhattan distance.

The fourth heuristic is the square of the Euclidean distance. This is slightly easier to calculate than standard Euclidean and will increase the variation in heuristic values when comparing nodes in the fringe.

The fifth heuristic is the radial distance, which considers distance from a node to the goal in terms of the circle of radius d centered at the goal.

1.4 Results



50 Tests / Heuristic	Column1	Column2	Column3	Column4
Heuristic	Fringe (KB)	Expanded (KB)	Time (s)	Distance
Uniform Cost Search	52.25	1674.59	0.234	109.25
Euclidean Dist. / 4	59.73	1375.17	0.214	108.42
Root Sum Distance	57.53	112.58	0.02	139.59
Sum of Squares	53.81	1530.87	0.272	108.22
Manhattan Distance	36.06	17.86	0.005	164.47
Radial (Cell)	80.94	438.8	0.076	120.41
Euclidean Dist. / 4	62.26	1187.69	0.191	109.86
Root Sum Distance	39.89	23.01	0.005	161.16
Sum of Squares	54.94	1451.67	0.218	110.45
Manhattan Distance	36.65	18.33	0.005	165.94
Radial (Cell)	52.42	43.28	0.009	148.6
Euclidean Dist. / 4	80.09	594.91	0.098	113.13
Root Sum Distance	36.39	17.12	0.005	166.56
Sum of Squares	57.44	1215.68	0.183	108.67
Manhattan Distance	36.65	16.33	0.004	164.94
Radial (Cell)	36.17	17.07	0.004	158.26

Figure 2: Weight = 1 for the heuristics in blue. Weight = 1.5 for heuristics in green. Weight = 3 for heuristics in orange.

1.5 Discussion

Uniform cost search, even though its complete and optimal, it does not do well with space and time complexity which it inherits from breadth first search like behavior. The first heuristic considered is both admissible and consistent, it helped created an optimal path with a slightly better time and space complexity. The complexity gains increased with weighted A*. Manhattan distance was not as accurate as some other heuristics since it did not consider much of the grid's terrain. The root of the Manhattan did not do as well probably because the reduced variation in values calculated. The sum of the squares did well as the weighting factor increased because of the increased variation provided from the squares. The radial distance did not do as well which could be because of a variety of things. One such reason could be the 8-neighbor nature of the grid and the lack of terrain consideration.

2 Sequential A* Search

Incomplete, since the search does not actually provide the most optimal path. This can be due to calculation cost and how the the Queue is prioritizing the Tiles.

2.1 Proof

We want to show that g-value of any state s, when it is the minimum anchor key in the iteration of the Sequential Heuristic A* algorithm, is bounded by w_1 times the cost of the optimal path to the goal.

Show that

$$f_0(s) \leq w_1 * c^*(s_goal)$$

g-value for s is at most w_1 suboptimal

$$g_0(s) \leq w_1 * c^*(s)$$

Anchor search for the algorithm requires an admissible heuristic to be used. Let $h_0(s)$ be the admissible heuristic used in anchor search.

$$c^*(s) + h_0(s) \leq c^*(s_goal)$$

The heuristic never overestimates the true path cost from s to the goal. The evaluation function $f_0(s)$ of the weighted A* anchor search is:

$$f_0(s) = g_0(s) + w_1 * h_0(s)$$

Applying $f_0(s)$ and $g_0(s)$ equations we get:

$$f_0(s) \leq w_1 * (c^*(s) + h_0(s))$$

Then applying the other two equations we haven't used we get:

$$f_0(s) \leq w_1 * c^*(s_goal)$$

Which satisfies the general case where $s \neq s_goal$ exists. In the case where it does equal then $h_0(s_goal) = 0$ and $c^*(s) = c^*(s_goal)$. In the case where no path then $c^*(s_goal) = \infty$ so it will always hold.