

HEURISTIC SEARCH

[Assignment One]

Cindy Lin, Amielyn Musa, & Sanidhi Borale

Introduction to AI

Professor Troy McMahon

10/20/2020

Optimizations:

The purpose of optimization is to achieve the best, most efficient design. Adhering to the criteria and constraints outlined by the assignment, we optimized our algorithms by using various data structures, calculating h , g , and f values, and testing our project as we worked on it. We considered factors such as productivity, runtime, efficacy, space, and more.

Proposed Heuristics:

Admissible/consistent heuristic: Manhattan Distance

The Manhattan distance is the safest option to calculate the heuristic value. If the value was overestimated, it would pose a problem in accuracy. Since it is counted block by block, the value will not be overestimated.

Inadmissible heuristics:

1. Euclidean Distance- Can be underestimated, but is still a good measurement.
2. Chebyshev distance- Provides a maximum distance.
3. Average of Euclidean and Manhattan Distance- Manhattan can potentially overestimate, and Euclidean can underestimate. If we average them, a more accurate heuristic value can be found.
4. Manhattan/2 - Rough estimate which is an okay starting place.

Experimental Results:

(5)

Average of Euclidean and Manhattan

	Runtime	Path Length	Nodes Expanded	Memory
A Star	3479.92	180.5163281381 4905	435.44	2054.08
Uniform Cost	85339.1	100.6639346624 3366	10765.74	32934.04
Weighted AStar(1.25)	1440.4	199.3348672614 2996	180.56	1222.34
Weighted AStar(2.0)	984.3	208.6058414570 998	123.54	1032.8

Chebyshev

	Runtime	Path Length	Nodes Expanded	Memory
A Star	22623.6	110.2950094990 893	2839.18	9460.38
Uniform Cost	85339.1	100.6639346624 3366	10765.74	32934.04
Weighted AStar(1.25)	16443.88	125.2060760419 1908	2061.54	7131.88
Weighted AStar(2.0)	3724.7	178.2932841437 0298	466.72	2112.56

Euclidian

	Runtime	Path Length	Nodes Expanded	Memory
A Star	13961.82	112.9802948343 9484	1752.2	6212.88
Uniform Cost	85339.1	100.6639346624 3366	10765.74	32934.04
Weighted AStar(1.25)	7333.44	129.8374452510 8995	918.42	3665.18
Weighted AStar(2.0)	1286.4	179.7789535066 5239	160.86	1157.32

Manhattan

	Runtime	Path Length	Nodes Expanded	Memory
A Star	9230.88	133.4316248947 7584	1157.94	4402.28
Uniform Cost	85339.1	100.6639346624 3366	10765.74	32934.04
Weighted AStar(1.25)	6333.02	165.3598997571 2164	793.6	3200.14
Weighted AStar(2.0)	1317.5	207.6181998726 367	165.4	1169.96

Manhattan/2

	Runtime	Path Length	Nodes Expanded	Memory
A Star	35739.58	112.0009925142 368	4495.24	14433.5
Uniform Cost	85339.1	100.6639346624 3366	10765.74	32934.04
Weighted AStar(1.25)	24924.38	116.0145141525 2235	3133.52	10376.5
Weighted AStar(2.0)	10096.74	132.0194284584 6065	1267.68	4798.64

(6)

Different heuristic functions affect the behavior of each algorithm in terms of all four presented measurements. Uniform Cost Search has the shortest path length, and all categories are constant for all heuristics. Manhattan/2 has the shortest path lengths, but the largest memories. Chebyshev has the second shortest path lengths, and the second largest memory stats. The other three have relatively similar path lengths, and proportional memories. Each algorithm performs relative to the heuristics used.

(7)

Sequential Heuristic A Star

	Runtime	Path Length	Nodes Expanded	Memory
Case 1: w1=1.25 w2=2.0	5377.56	189.87	969.66	3686.84
Case 2: w1=2.0 w2=1.25	4386.92	188.7	834.5	3650.76
Case 3: w1=2.0 w2=2.0	3872.74	200.88	736.44	3339.56

Sequential Heuristic A star utilizes all the heuristics, admissible and inadmissible, while A star only takes into consideration the Manhattan distance heuristic. When we look at Sequential Heuristic A Star, case one and case two have very similar path lengths, but case one's runtime, nodes expanded, and memory usage are larger than case two. Case 3 has the longest path length, but the least runtime, nodes expanded, and memory usage. When we compare these results to the A star algorithm using Manhattan Distance as the heuristic, we can see that A star produces the shortest path length but has much greater values for runtime, nodes expanded, and memory usage. From this data, we can conclude that the A star algorithm is more efficient than Sequential A star because the path length is significantly shorter.

(8)

Our implementation is efficient because it utilizes various heuristics and calculates a path length that is close to the distance formulas employed. Our algorithm also uses transition cost, as well as the f and g values to calculate this path. In terms of shortest path length/solution quality, relative to the other methods, uniform cost performed the best in every table. A Star was second, followed by Weighted A Star(1.25), Weighted A Star(2.0), and lastly Sequential Heuristic A Star. In terms of computation time, Weighted A Star(2.0) performed the best, followed by Weighted A Star(1.25), A Star, and Uniform cost. From these results, we can observe and deduce that our implementation is accurate because across the board each search algorithm performed the same relative to the heuristic measurement used. We can also conclude that Uniform cost search is the most effective search algorithm because it is consistent across all heuristics.

(9)

Prove (part i) show that the anchor key of any state s , when it is the minimum anchor key in an iteration of Algorithm 2, is also bounded by w_1 times the cost of the optimal path to the goal. In other words, show that for any state s for which it is true that $Key(s, 0) \leq Key(u, 0) \forall u \in OPEN_0$, it holds that $Key(s, 0) \leq w_1 g(s)$

Proof by contradiction. Assume that $key(s, 0) = g_0(s) + w_1 h_0(s) > w_1 g(s_{goal})$. In addition, let's call the least cost path to be $L = (s_0 \text{ or } s_{start}, \dots, s_k \text{ or } s_{goal})$. From this path, we pick the first state s_i , that has yet to be expanded to by the anchor search. In addition, as hinted, s_i in the $OPEN_0$ queue, which is located along a least cost path from s_{start} to s_{goal} does always exist in the queue (s_0 is put in $OPEN_0$ at line 18, when state $s_j \in L$ is expanded in the anchor search $s_{j+1} \in L$ is always inserted in $OPEN_0$, and finally, s_k is never expanded by the anchor search, or it terminates if s_k has the least key (Line 30).

Examining $g_0(s_i)$, if $i = 0$, we have $g_0(s_i) = 0 \leq w_1 * g(s_i)$. if $i \neq 0$, we see that s_{i-1} has been expanded in the anchor search. With s_i as a successor of s_{i-1} , we have . . .

$$g_0(s_i) \leq g_0(s_{i-1}) + c(s_{i-1}, s_i) \leq w_1 * (g(s_{i-1}) + c(s_{i-1}, s_i))$$

$$s_{i-1}, s_i \in optimal = w_1 * g^*(s_i)$$

Thus, $g_0(s_i) \leq w_1 * g^*(s_i)$. Using this, we get..

$$key(s_i, 0) = g_0(s_i) + w_1 * h_0(s_i)$$

$$\leq w_1 * g^*(s_i) + w_1 * c^*(s_i, s_{goal})$$

$$= w_1 * g^*(s_{goal})$$

Now, as $s_i \in OPEN_0$ and $key(s_i, 0) \leq w_1 * g^*(s_{goal}) \leq key(s, 0)$, we have the contradiction to our assumption that $key(s, 0) \leq key(u, 0), \forall u \in OPEN_0$

Prove (part ii) that when the Sequential Heuristic A Star terminates in the i^{th} search, that $g_i(s_{goal}) \leq w_1 * w_2 * c(s_{goal})$. In other words, prove that the solution cost obtained by the algorithm is bounded by a $w_1 * w_2$ sub-optimality factor.

The Sequential Heuristic A Star Algorithm can terminate successfully in lines 30 (anchor search) or 22 (inadmissible search), or it can terminate without a solution in line 19.

If the anchor search terminates at line 30,

Given $\rightarrow \text{key}(s, 0) \leq \text{key}(u, 0), \forall u \in \text{OPEN0}$

$$\begin{aligned} g0(s) &\leq w_1 * c * (s) \\ &\leq w_1 * w_2 * c * (s) \rightarrow w_2 \text{ must be } \geq 1.0 \end{aligned}$$

If an inadmissible i^{th} search terminates in line 22, then

$$\begin{aligned} gi(s) &\leq w_2 * \text{OPEN0.Minkey()} \\ &\leq w_2 * w_1 * c * (s) \end{aligned}$$

In both the anchor search and an inadmissible search, we have proven that the solution cost is bound within the $w_1 * w_2$ sub-optimality function.

