CS 405 Homework 2 Final Report
A* with many *s

Cameron K. Titus   Curtis Fortenberry   Laura Lundell
Andrew Parker

April 15, 2019

A report submitted for the undergraduate section.

Figure 1: A star

1 Introduction

In this project we are solving a user created maze using the A* algorithm. The agent uses the number of steps to the current node to the start, and the distance between the current node and the end to determine where to search next. The A* Search algorithm was first mentioned in ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths”[1].

For the group we have Curtis Fortenberry, Andrew Parker and Laura Lundell; all three are junior CS students. As well as Cameron K. Titus a senior CS student.

2 Algorithm

A* starts by searching around for possible places to expand its path. For every expandable square, A* will calculate a certain heuristic, and then put all of the squares as nodes in a binary heap. A node's position in the heap is determined by the chosen heuristic associated with the square. The algorithm will continue this pattern of removing the current (top) node from the heap, expanding this node's adjacent squares, calculating the heuristic, and putting all the new adjacent squares into the heap. The algorithm will continue this pattern until the end of the maze is found.

Algorithm 1 A* algorithm

```
1:  function A*
    Q:= the set of nodes in V, sorted d(v) + h(v)
2:      while Q not empty do
3:          $V \leftarrow Q.pop()$
4:          for all neighbours of u of v, sorted by $d(v) + h(v)$ do
5:              if $d(v) + e(v,u) \leq d(u)$ then
6:                  $d(u) \leftarrow d(v) + e(v,u)$
7:              end if
8:          end for
9:      end while
10: end function
```

- $u \& v$ are 2 nodes connected by 1 edge
- $e(u,v)$ is the edge distance from e to v

3 Proposed Solution

We plan to implement our A* search agent using a binary heap to store our maze environment and having our agent take the next least node in the heap until the end node has been found. Once the end node has been found it will walk backwards through the path to generate the final path. Table 1 contains our PEAS description.

PEAS	Your Approach
Performance Measurement	Does the agent choose the shortest possible path?
Environment	Maze implementation
Actions	The agent chooses nodes that represent the lowest cost function.
Sensors	The agent measures cost by calculating distance from the origin to the current node and the distance from the current to the goal node.

Table 1: The PEAS description for A* Search Algorithm.

4 Proposed Experiment

The optimal path in an A* Search Algorithm is the shortest path so we intend to test our program by running the agent through a variety of test mazes to see if it can find the shortest path between the start and end nodes in each case. We will also compare different heuristics to see which performs best.

5 Analysis

Data was collected by measuring the time for each heuristic used for a single simulation. The path length was also measured for each maze, but each calculated identical path lengths with respect to the maze it analyzes.

Four heuristics were used: Euclidean distance, Manhattan distance, Chebyshev, and Octile distances.

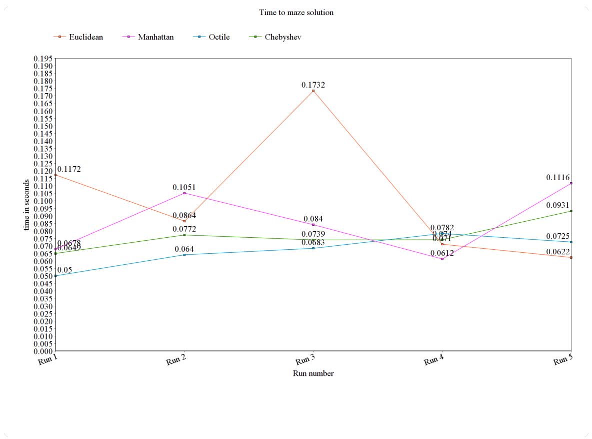

Figure 2: All heuristics measured in time (seconds)

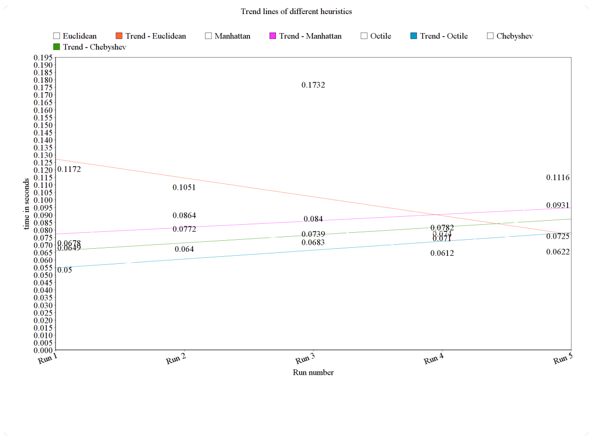

Figure 3: Trendlines for all heuristics

5.1 Euclidean

Andrew analyzed the Euclidean distance heuristic to measure time elapsed for the A* Search algorithm. It performed the worst in time measurement, with an average of 0.102 seconds. The first run it performed very poorly at 0.1172 but improved by the fifth and final test, with oscillating times in the middle three tests.

$$\text{sqrt}(xDiff^2 + yDiff^2) = \text{sqrt}(9^2+3^2) = 9.48$$

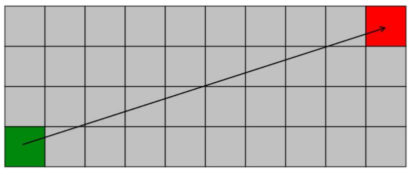

Figure 4: Euclidean distance, from [2] Rabin, 2017

5.2 Manhattan

Laura analyzed the Manhattan distance heuristic to measure time elapsed for the A* Search algorithm. It performed at an average of 0.08594 seconds to find the shortest path, which was the third best average time of the four heuristics tested.

$$xDiff + yDiff = 9 + 3 = 12$$

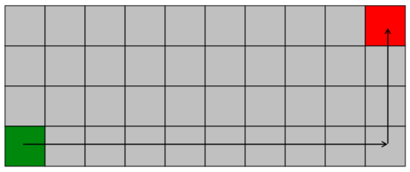

Figure 5: Manhattan distance, from [2] Rabin, 2017

5.3 Chebyshev

Cameron used the Chebyshev distance heuristic to measure time elapsed for the A* Search algorithm. It performed second best of the four tested heuristics with an average time 0.07662 seconds.

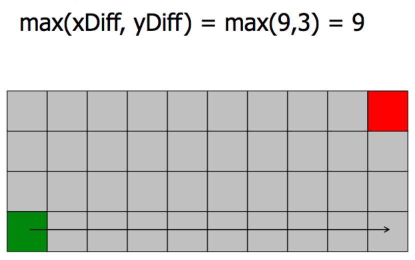

Figure 6: Chebyshev distance, from [2] Rabin, 2017

5.4 Octile

Curtis will use the Octile distance heuristic to measure the time elapsed for the A* Search algorithm. This calculation performed best of all four heuristics, with an average of 0.0666 seconds to perform the given maze.

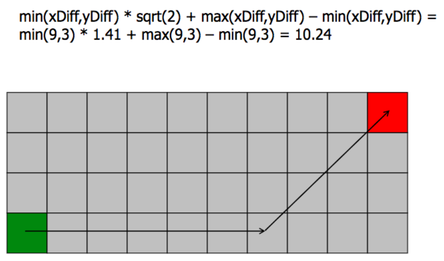
min(xDiff,yDiff) * sqrt(2) + max(xDiff,yDiff) – min(xDiff,yDiff) =
min(9,3) * 1.41 + max(9,3) – min(9,3) = 10.24

Figure 7: Octile distance, from [2] Rabin, 2017

6 Conclusion

Overall, the Octile distance heuristic performed the best of the four heuristics: Euclidean, Manhattan, Chebyshev, and Octile.

6.1 Positive outcomes

This experience strengthened us as a team, as collection of students and as future professional collaborators. We learned how to analyze and improve the pre-existing code that Andrew had written for a framework. We took his agent pattern from a previous personal project, and updated it to CS405 standards. We then each had a great jumping off point to implement our respective heuristics. These can be found primarily in the constructor for the Node class.

6.2 Future improvements

To expand on the current project, we would consider implementing additional measurements to the system. For example, each number of operations and searched nodes could be measured and compared relative to the other heuristics.

More tests might also be run to get a broader picture of how each heuristic performed relative to the others.

References

[1] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

[2] Steve Rabin. Artificial intelligence in games, 2017. Last accessed 14 April 2019.