Impact Factor (SJIF): 6.092 E-ISSN: 2583-1615

International Journal of Research in Academic World

Received: 09/October/2024 **IJRAW:** 2024; 3(11):76-78 **Accepted:** 14/November/2024

A Comparative Analysis of Heuristic Approaches in the A* Algorithm for Path Finding in Autonomous Vehicles

[*1]Rishita Tiwari, [2]Ashwani Dwivedi and [3]Shweta Sinha

[*1, 2]Student, National PG College, Lucknow, Uttar Pradesh, India.

[3]Assistant Professor, Department of Computer Science, National PG College, Lucknow, Uttar Pradesh, India.

Abstract

In this article, we will give a more in-depth comparison of different heuristic approaches for A* pathfinding algorithm with respect to their implications to effectiveness calculated by computation and optimality of path. An investigation of four different heuristics—Manhattan, Euclidean, Diagonal and a Custom Hybrid where compared based on important performance metrics: computation time and path length —in simulation using a grid with characteristics representative of problems generally faced in the paths computed for an autonomous vehicle. The paper serves to enlighten the different kinds of heuristics and their effects on A* algorithm efficiency, which basically expresses a trade-off between speed and accuracy. The findings suggest that there is no one-size-fits-all heuristic; rather, it will be a custom solution tuned to the environmental constraints that yields better performance. Such potential implications are exciting for applications in areas such as robotics, gaming and autonomous vehicle navigation.

Keywords: A* path finding algorithm, heuristic comparison, autonomous vehicle navigation, computation time and path length.

Introduction

Pathfinding problem is a fundamental problem of artificial intelligence, especially in autonomous vehicles, robotics and navigation systems. Since these systems will need to make real-time decisions capable of execution on a lot of environmental situations with obstacles and complex terrain, the effectiveness of any algorithm highly relies upon real-life implementations. A* algorithm has emerged to be one of the most practical implementations for pathfinding algorithms as it embraces both the principles of optimality (finding the shortest path) and efficient computation. Since it uses a heuristic to guide the search, this algorithm takes advantage of Dijkstra's optimality guarantee and greedy best-first search at quickly finding a solution [1].

At a high level, A* keeps an open list of nodes to explore further. Then it chooses one node with the lowest cost, which is defined as a sum of the real cost to reach that node and some kind of prediction about how far it is to finish. The A* algorithm performance is reliant on the heuristic function and its ability to give us a cost estimate of reaching the goal given that we are at a node. A wide range of heuristic methods is available; however, each method has its limitations and merits. In this, we use four of the most widely-used heuristics: Manhattan, Euclidean, Diagonal and a Custom Hybrid heuristic. The selected heuristic not only influences search efficiency but also the quality of the path discovered. For example, a few heuristics are Manhattan and diagonal. It will be computationally simpler but lethargic since these may not reach a optimum solution in most conditions, when diagonals are allowed to move. Euclidean on the other hand is even worse because it can get to shorter paths and at some point more computations, but this quickly gets unacceptable for any response time once the grid or precision used is large. The Custom Hybrid heuristic integrates aspects of both the Manhattan and Diagonal heuristics, thereby yielding a solution that demonstrates adaptability across diverse environments. This research undertakes a comparative analysis of four distinct heuristics, aiming to evaluate their appropriateness for various navigation tasks. Emphasis is placed on dynamic and real-time conditions, particularly those experienced by autonomous vehicles, thereby highlighting the relevance of each heuristic in practical applications [9].

Methodology

In this experiment, the A* algorithm was coded using Python in a grid-based simulation context. The grid is a representation of the terrain over which the algorithm will run; every cell in the grid represents a node to traverse or block. The heuristic functions for investigation are:

- **Manhattan Heuristic:** It simply calculates the total of the absolute differences in the x and y-coordinates between the current node and the goal node. Manhattan Heuristic It is strong where the action set permits only strictly horizontal and vertical motion.

***Corresponding Author:** Rishita Tiwari

h_Manhattan = |x_goal – x_current| + |y_goal – y_current|

- **Euclidean Heuristic:** In Euclidean heuristic, the distance between two points is given as a straight line with diagonal movements. It is more precise but in case of much computation since it involves square roots.

h_Euclidean = √((x_goal – x_current)^2 + (y_goal – y_current)^2)

- **Diagonal Heuristic:** The Diagonal heuristic considers both horizontal/vertical and diagonal move directions. It balances the between computational efficiency and path accuracy, so it is particularly useful for environments which also allow diagonal movements.

h_Diagonal = min(|x_goal – x_current|, |y_goal – y_current|) * √2 + (|x_goal – x_current| - |y_goal – y_current|)

- **Custom Hybrid Heuristic:** This is essentially a combination of both the Manhattan and Diagonal heuristics, with weights adjusted based on the environment's constraints. It would thus be balanced properly between computation speed and possible path optimality.

The experiments were carried out over a 10x10 grid. For all heuristics, the same conditions were employed. The performance of the algorithm was measured along the following criteria:

Computing Time: The entire time it takes the algorithm to find a path.

Path Length: Number of Nodes traversed by the algorithm to reach the goal.

The evaluation was done under a number of trials to ensure statistical significance. Results averaged to be able to compare [7, 8].

Implementation

The A* algorithm was implemented in the Python programming language, using standard data structures like lists and priority queues to maintain the open and closed lists. The heuristic functions are implemented as stand-alone functions that are called during the execution of the algorithm to compute the cost estimates. Thus, the overall workflow of the A* algorithm was:

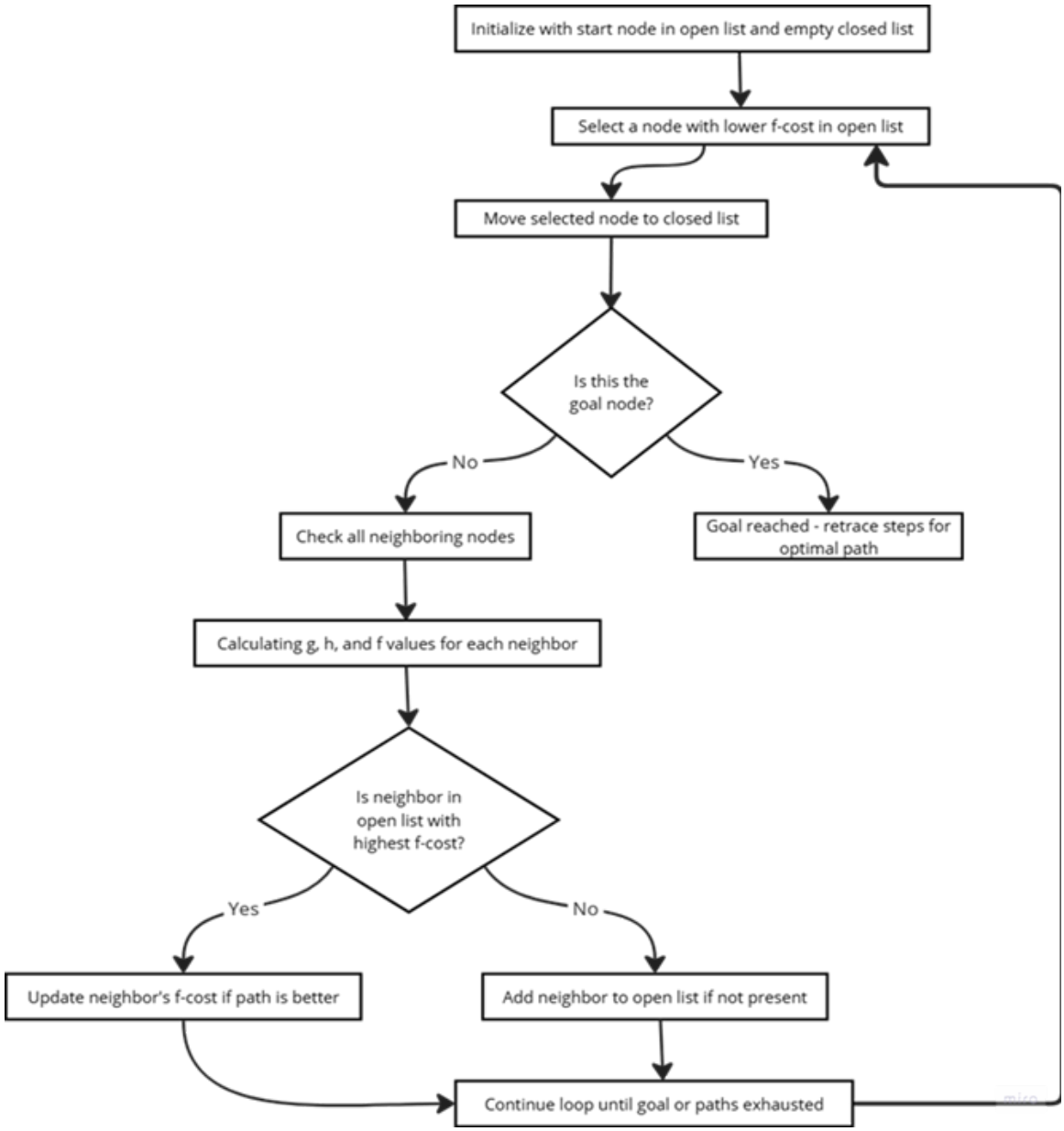

Fig 1: Flowchart of Pathfinding Algorithm Using Heuristic Cost Evaluation

Results

The results are analysed based on the two major factors, which are computation time and path length.

Table 1: Comparison of Heuristic Algorithms Based on Computation Time and Path Length

Heuristics	Average Computation Time (Seconds)	Path Length (Nodes)
Manhattan	0.0020	14
Euclidean	0.0030	12
Diagonal	0.0025	11
Custom Hybrid	0.0030	13

Compare Computation Times: The Manhattan heuristic had the shortest time of computation followed by Diagonal. Euclidean and Custom Hybrid had similar though slightly higher times due to more complex operations [5, 6].

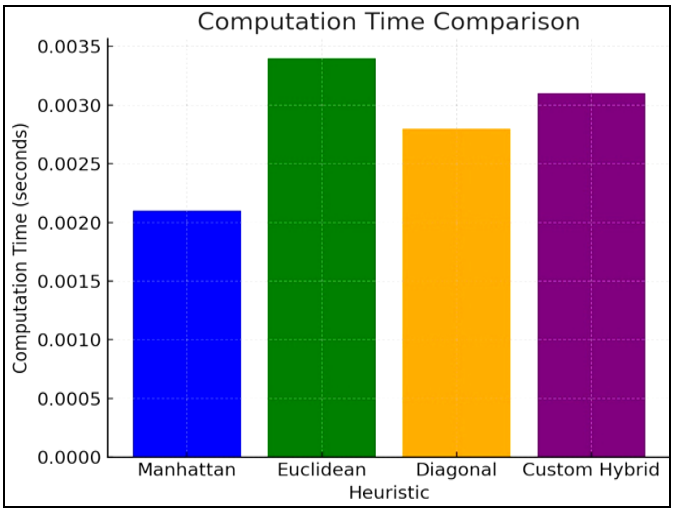

Fig 2: Computation Time Comparison

Path Length Comparison: The Euclidean heuristic always output the shortest paths, whereas the Diagonal heuristic threw a relatively short path with better performance than the Manhattan approach. Custom Hybrid heuristic performed comparable to Diagonal heuristic but with slightly higher computation time.

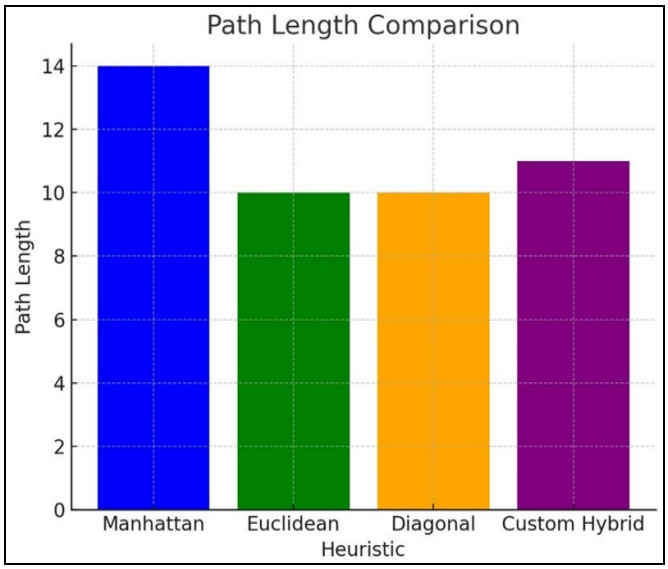

Fig 3: Path Length Comparison

Analysis

From results, it is clear that the selection heuristic influences both running time and optimality of A*. The MHT strategy is simple and therefore takes a shortest computation time at the expense of longer paths. EHT, on the other hand, computes the shortest distances for the paths at the expense of being more computationally intense especially for large grids. The Diagonal heuristic balances computation time with the accuracy of paths, which should make it really attractive when diagonal movement is allowed in an environment. The Custom Hybrid heuristic provides a flexible solution but is not significantly better than the competition and thus probably has use only in targeted scenarios where movements can vary [3, 4].

Conclusion and Future Work

This experimentation also indicates how important it is to choose the right heuristic for pathfinding operations, especially when we talk about autonomous vehicle navigation application. The first would be used for environments characterised by time and simplification, while the second is useful in terms of precision when paths must be defined accurately. In fact, the diagonal heuristic itself is kind of a good middle ground between these two heuristics. The Custom Hybrid Heuristic allows adaptability over different environments [10, 2].

The future work can build on the adaptive heuristics that will be developed which vary in real time with information obtained about the environment, e.g. static/dynamic layout of grid or obstacles. For autonomous systems, these can be tested in more complex environments like larger city-scale maps or obstacle-infested terrains and see the extent of generalization achieved from them. Additionally, by using past experiences and the current environmental features, these machine learning based techniques can allocate or tune heuristics dynamically on particular instances to increase the robustness of pathfinding algorithms in practice.

References

1. Hart PE, Nilsson NJ & Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics. 1968; 4(2):100-107.
2. Koul A, Greydanus S & Fern A. Learning Finite State Representations of Recurrent Policy Networks. arXiv preprint arXiv:1811.12530, 2018.
3. Schaeffer J *et al*. Pathfinding in Games and Simulation Using A with Multiple Heuristics*. AI Game Programming Wisdom, Charles River Media, 2001.
4. Thrun S, Burgard W & Fox D. Probabilistic Robotics. MIT Press, 2005, 253-261.
5. Cui X & Shi H. A Pathfinding Algorithm Optimized for Large Environments*, 2011.
6. Nash A *et al*. Any-Angle Path Planning. AIIDE, 2007, 23-28.
7. Gonzalez J. Evaluating A Heuristics for Optimal Pathfinding in Dynamic Environments*, 2018.
8. Karaman S & Frazzoli E. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*. 2011; 30(7):846-894.
9. Wang L & Kim J. *Heuristic Path Planning for Autonomous Vehicles in Urban Traffic Scenarios*. Journal of Advanced Transportation, 2020.
10. Silver D & Huang A. *Mastering the Game of Go with Deep Neural Networks and Tree Search*. Nature. 2016; 529(7587):484-489.