A COMPARATIVE STUDY AND EVALUATION OF PATH PLANNING ALGORITHMS IN 2D ENVIRONMENT

By

MUHAMMAD MAMUNUR RASHID
(151-35-1045)

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Software Engineering

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

FALL-2018

APPROVAL

This thesis titled on "**A Comparative Study and Evaluation of Path Planning Algorithms in 2D Environment**", submitted by **Muhammad Mamunur Rashid**, **151-35-1045** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

DECLARATION

It hereby declares that this thesis has been done by me under the supervision of Mr. Sheikh Shah Mohammad Motiur Rahman, Lecturer, Department of Software Engineering, Daffodil International University. It also declare that neither this thesis nor any part of this has been submitted elsewhere for award of any degree.

Muhammad Mamunur Rashid
ID:151-35-1045
Batch: 16th
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

Certified by:

Mr. Sheikh Shah Mohammad Motiur Rahman
Lecturer
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

ACKNOWLEDGEMENT

First and foremost, I have to thank my research supervisors, Mr. Sheikh Shah Mohammad Motiur Rahman. Without his assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. I would like to thank you very much for your support and understanding over these past eight Month.

I would also like to show gratitude to my committee, including Kaushik Sarker,Mr. K. M. Imtiaz-Ud-Din and Mr. Khandker M. Qaiduzzaman. Mr. K. M. Imtiaz-Ud-Din was my 9th semester Artificial intelligence Assistant Professor at Daffodil International University. His teaching style and enthusiasm for the topic made a strong impression on me and I have always carried positive memories of his classes with me. I must also thank to my two educator Mr. Khandker M. Qaiduzzaman and Md.Habibur Rahman for their involvement in my thesis.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

ABSTRACT

Background:
There are lots of use of path planning algorithm in search-related applications including robot exploring, game design, artificial intelligence, traffic route navigation, network routing and path analysis is most extensive.

Objective:
Comparison study on three path planning algorithm Dijkstra, Greedy BFS and A-Star algorithm to find out which algorithm gives the shortest path for a particular 2D environment with an obstacle.
Methods: Dijkstra algorithm, Greedy BFS and A-Star algorithm. For distance calculation have used Euclidean Space calculation.

Results:
This literature carries out from simulation result that A* provide best optimal path rather Dijkstra algorithm, Greedy BFS. But sometimes A* and Greedy BFS show same performance.

Conclusions:
A star gives the best performance on more obstacle in static environment. It can avoid the obstacle and travel less node than Dijkstra, Greedy BFS. In less obstacle density Dijkstra , A* and Greedy BFS travel same number of node.

Keywords: Dijkstra, Greedy BFS, A Star, Path Planning, Path Optimization;

CHAPTER 1

INTRODUCTION

1.1 Background

Path planning is the process of creating a path from any starting point to goal point in kind of environment. Now a days it is an essential part of the aerial vehicle. Public Transport Route Planning also use the disjkstra (Alican Bozyiğit, 2017). An Efficient Hardware Architecture algorithm use for Shortest Path Search Engine (Woo-Jin Seo, 2009). path planning algorithm is apply on find out the solution of optimal path (Mackworth, 2017; Pearl, 1984).for radar network optimization also use the path planning algorithm (Fu Xiao-wei, 2010). Wireless Sensor Networks path planning used the optimization algorithm (Xu Yulong, 2016).

1.2 Motivation of the Research

Path planning is the most important part of UAV (Unmanned Arial Vehicle) (KHAILI, 2014). It has so many uses in a different area one of the controls of autonomous mobile robots (Shaimaa Ahmed, 2016). There are so many countries they have used UAV for observation and gather intelligence. For combat aircraft has great significance effect for automated flight paths design as without increasing workload for pilots and improve the mission effectiveness and also allows the possibility of updated flight paths being generated in response to the receipt of new intelligence concerning the mission (Tianyou Chen, 2018).

1.3 Problem Statement

One of the fundamental problems for path planning is navigating a terrain and finding the shortest path to a Goal location (Nadira Jasika, 2012). There are many algorithms that can solve this problem, most common and widely known like the Dijkstra search, greedy BFS, and A Star search algorithm. To reach goal, robot has to traverse the terrain. To identify all the nodes that surround it, a brute force approach would be to start with a node. Move to one of the adjacent nodes and repeat this process until the goal node is found. There will be so many obstacles and also have a starting and target node algorithm have to find out the path and path should be optimal.

1.4 Research Questions

There are so many algorithms that can plan for path planning, but they are not always optimal. Optimal is the most sufficient for UAV control problem.so have to find out which algorithm is best for optimal, though optimality depends on the environment.

Why A-Star algorithm is best for find out the shortest pat?

1.5 Research Objectives

To find out the Best Algorithm among Dijkstra search, greedy BFS, and A Star search algorithm. Established that A Star is best algorithm for path optimization.

1.6 Research Scope

A Star is best algorithm for optimization, but it is not always best. Sometimes it gives the same result of dijkstra and greedy BFS. The scope of this paper is compare three algorithms (dijkstra, greedy BFS and A*) and all the algorithms are applied on only 10X10 grid environment. Where agent can move left, right, up, down and also diagonal. Algorithms are work on a static environment and it will not give a good or optimal solution in the dynamic environment. It will not work in properly if the obstacle or goal is moving.

1.7 Contribution:

Establishing A* algorithm is best path planning algorithms in 2D static environment among dijkstar, pure heuristic and A* algorithms.

1.8 Thesis Organization:

Chapter 1 describe the Introduction of the thesis Chapter 2 will describe the related works or what is the latest work on algorithm, Chapter 3 represent the methodology which function and algorithm have used for simulation, Chapter 4 will show the simulation result and Chapter 5 represent recommendation and conclusion of the thesis area.

CHAPTER 2

LITERATURE REVIEW

Enacting literature review will describe the existing work that have been published in the past based on path planning algorithms which relate to my thesis area. In the past publication they have been discuss how they applied and compare the algorithms.

4.1 Route Planning

Based on the basic theory of A-star algorithm compares its advantages and disadvantages with genetic algorithm and ant colony algorithm. On the other hand (Alongside) construct a battleground environment model of unmanned aerial vehicle (UAV) for path planning. And also, A-star algorithm has applied on that environment with least fuel and least danger multiple constraints (Tianyou, 2018).

2.2 USV Avoiding Underwater Obstacles

The literature narrates the path re-planning techniques and also obstacle avoidance for an unmanned surface vehicle (USV) underwater based on multi-beam forward looking sonar (FLS). In underwater obstacles computing a numerical solution procedure based on an A* algorithm has used for near-optimal paths in static and dynamic environments. To regenerate the optimal path for several updated frames in the field of view of the sonar with a proper update frequency of the FLS has (Phanthong, 2014) (Guohao, 2018) (Gopikrishnan, 2011) (Chimanga, 2016) (Valenzano, 2014) been tested with real-time path re-planning algorithm. The proposed method was able to avoid both a single stationary obstacle, multiple stationary obstacles and moving obstacles with the Global Positioning System (GPS) of the USV (Phanthong, 2014).

2.3 Performance comparison

According to (B.Moses Sathyaraj, 2008) performance comparison in variant environment if the number of nodes is enhancement then complexity of the search algorithm will increase. But A_star was giving best performance rather than Dijkstra's, Bellman Ford's, Floyd-Warshall's.

2.4 Demand control

From the problem of a critical power deficit due to low water levels in Zambia that has oppressed the economy. That is the primary source of electricity. The government also making efforts to overcome by diversifying to renewable solar energy but it was not an optimal solution and also the current photovoltaic (PV) systems are not to capable to produce sufficient energy. In this literature has proposed a system to demand control in PV systems by using the Best First Search algorithm. As a result, the proposed system is efficient to optimize user priorities and limits the power consumed by finding the best combination of household appliances. It has achieved the goal by using Artificial Intelligence heuristic search. To determine the best combination of these household appliances has used the Best First Search algorithm with appropriate heuristic value (it's actually an A* search). And it was also optimal rather than Breadth First Search algorithm to find out the best combination of household appliances (Chimanga, 2016).

2.5 A* algorithm in Traffic Navigational System

A* algorithm is used for achieving high efficiency of vehicular navigation by analyzing capabilities of maturity, optimality, time complexity, and space complexity in variants evaluation function (Jingang, 2010).

2.6 Evaluation of multiple algorithm

Comparison study based on computational time and solution optimality in variant obstacle information. Recently the researchers have attracted to (UAVs) unmanned aerial vehicles due to their much potential civilian applications. However, current robot navigation technologies need further improvement for diligent application to several scenarios. One of the key issues for robot navigation is, the robot has to “Sense and Avoid” capability. The most challenging part of the navigation is avoiding the noise while accounting for sensor noise, uncertainties in operating conditions, and real-time applicability. The purpose of this paper is the evaluation of widely known UAVs path planning and obstacle avoidance algorithm based on heuristic and non-heuristic method on three different global and local obstacle scenarios. The literature carrying out computational time and solution optimality of the algorithms. And it will provide a clear vision to the reader for which algorithm is best for which obstacle scenarios. Cause environment and obstacle play a huge role in the overall performance (Radmanesh, 2018).

2.7 Summery

In the above literature review there is lots of uses of dijkstar, greedy heuristic and A* algorithms in different environment and different area for evaluating which algorithm gives the best or optimal solution, but every algorithm have a significant role based on the environment and area. For traffic navigation system A* is more optimal rather than any algorithm.

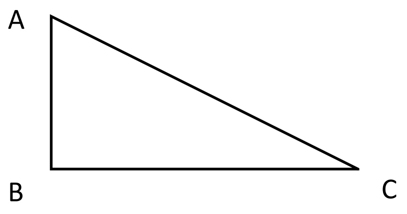
CHAPTER 3

RESEARCH METHODOLOGY

In the above paper will discuss three path planning algorithm and will apply one distance space calculating methodology for finding adjacent node.

3.1 Heuristic Function

Heuristic: The practice of heuristics was progressed by psychologists Amos Tversky and Daniel Kahneman among 1970s to 80s to help human decision making (Wikipedia, 2018). Though it was basically proposed by Novel laureate Herbert A. Simon. The heuristic is a process to perform a practical method for problem-solving. It also logical, rational, and sufficient to arrive an immediate goal but not guaranteed to be optimized. It is used for making a process speedy to find out the optimal solution [2]. Basically, heuristic function is a method to instruct of an algorithm about the goal direction [3]. Usually, we use two types of heuristic value calculation formula.one is the Manhattan distance or taxicab geometry calculation and another is the straight-line distance (SLD) or Euclidean Space calculation (Amos Tversky, 1982; ZeFang He, 2017). But of course, heuristic value must be less than actual distance (Krause, 1975).



Manhattan distance =>AC=AB+BC

Taxicab geometry is the sum of the absolute differences two Cartesian coordinates (Krause, 1975).

Euclidean Space calculation=>AC=$\sqrt{AB^2 + BC^2}$

Basically, its follow the Pythagorean Theorem to calculate distance to two Cartesian coordinates (Judith D.Sally & Paul J.Sally, 2007). Here has used the Euclidean distance formula for this particular path planning problem-solving. Denoted by h(n), where n is node pointer that describes the location.

3.2 The Dijkstra's algorithm

The Dijkstra is an uninformed search algorithm. It was envisioned by Computer Scientist the Edsger Wybe Dijkstra in 1956 (Dijkstra, 1959). the algorithm stand in many diverse, original divers is the shortest path finding between two nodes in a graph (Jehn-Ruey Jiang, 2014) (Dijkstra, 1959).it has another variant that is called shortest path finding from single source node to all other nodes in a graph.it can also be used to finding a shortest path from a single source node to a single destination node from a graph with put an end, the algorithm when once the shortest path of the destination is found (Risald, 2017). We apply here single source to a single destination shortest path method. The whole environment is demonstrated in the grid. So that search environment is simply a two-dimensional array. Each point of the array is rectangle or node and traversable or not traversable (Fan, 2010). The most important thing in the algorithm is to find out all adjacent rectangle or square that are connected to each other with an edge (KHAILI, 2014), fixed a node as an initial node=current node and its path distance is 0 just because it has not traveled yet. Expand the node and mark all the adjacent node of the initial node (Wang, 2011). And have to find out which adjacent node will be determined, when once determine the traverse adjacent node, the traveler for sure move from the center of the rectangle (node) to the next node and have to fix that current node as a visited node (Mohamed, 2014). It will stop when the current node is the final destination node (Kang, 2008). At every current node of the path, searching has to check the adjacent node and also have to ignore the obstacle terrain node. The adjacent node will store in open list and have to choose an adjacent node from the open list, the process will continue until the destination will found (Kang, 2008). But which adjacent node has to select from the open list that proceeding with the algorithm? That is appointed by the function f (n), which must be minimized to take a low-cost path.

$$f(n) = g(n).$$

Here, g(n) = real path-cost or distance between two pairs node which is directly connected.

Dijkstra's Pseudo code:

1) First have to set beginning node in the OPEN list than compute the cost function f (n). [Where g (n) =path cost between the goal and the start node, f (n) = g (n).]
2) Find the adjacent node with the least cost f(n) on the open list
3) Shift to the CLOSED list from the OPEN List which node has the smallest cost function f (n).
4) Incase two or more nodes have the same cost function f (n), then it will chose randomly resolve ties.
5) Algorithm will terminate if n is the goal node and to obtain the solution optimal path use the pointers. Otherwise, will continue until find the goal.
6) Determine all the adjacent nodes of n and calculate the cost function f (n) for every adjacent, not on the CLOSED list.
7) Associate with every adjacent, not on list OPEN list or CLOSED list the cost calculated and shift these on the OPEN list, placing pointers to n (n is the parent node).
8) Associate with any adjacent already on OPEN the smaller of the cost values just calculated and the previous cost value. (min (new f(n), old f(n))).
9) Repeat from step 2.

3.3 Greedy Best First Search or pure heuristic search Algorithm

Greedy Best First Search is the type of inform search, that operates based on estimated cost of real path distance that’s call heuristic value, denoted by h(n) (Marinescu, 2010). The greedy BFS is extending nodes based on their heuristic function, it’s a searching method, in hopes to find out an optimal solution in every step by selecting the best local choice (Lu, 2013). But does not think about seriously how the current solution is optimal (Chimanga, 2016). To estimate the distance from each local choice to goal point uses the heuristic value (Mahmud, 2012). By estimating the distance with search space for one step ahead from any node to goal node define the heuristic function h(n). after that select a minimizing heuristic function.

h(n)=Manhattan distance or Euclidean Space.

But here use the Euclidean Space for estimating the heuristic cost. If the heuristic function h(n), for all n, is always less than or equal to the actual path(cost) then it will always find out a solution.

Greedy BFS Pseudo code:

1. First have to set beginning node in the OPEN list than compute the cost function f (n). [Where h (n) = 0 and f (n) = h(n).]
2. Find the adjacent node with the least cost f(n) on the open list
3. Shift to the CLOSED list from the OPEN List which node has the smallest cost function f (n).
4. Incase two or more nodes have the same cost function f (n), then it will chose randomly resolve ties.
5. Algorithm will terminate if n is the goal node and to obtain the solution optimal path use the pointers. Otherwise, will continue until find the goal.
6. Determine all the adjacent nodes of n and calculate the cost function f (n) for every adjacent, not on the CLOSED list.

7. Associate with every adjacent, not on list OPEN list or CLOSED list the cost calculated and shift these on the OPEN list, placing pointers to n (n is the parent node).
8. Associate with any adjacent already on OPEN the smaller of the cost values just calculated and the previous cost value. (min (new f(n), old f(n))).
9. Repeat from step 2.

.

3.4 A* Algorithm

This algorithm first exposition by Peter Hart, Nils Nilsson and Bertram Raphael in 1968 (Peter E.Hart, 1968) (wikipedia, 2018).it's the combination of Dijkstra and Greedy best first search. It's actually a inform search, just because it has knowledge about the solution for every particular problem that is use for those particular problem There are many algorithms that can find out shortest path, A Star search algorithm most common and. To reach goal robot has to traverse the terrain (Guohao, 2018). To identify all the nodes that surround it, a brute force approach would be to start with a node (Haifeng Wang, 2014). Move to one of the successor nodes and repeat this process until the goal node is found. A Star search algorithm is best for finding the shortest path. However, this method does not always guarantee the best path to the target and computationally intensive. The main key is identifying the appropriate adjacent node (Jingang, 2010). To make an educated guess has given some information regarding the location of the target (I. S. AlShawi, 2012). For example, if the robot knows the target lies to the west, explore nodes to the west of your current location (Iram Noreen, 2016). If the robot knows the direction of the target robot could always try to move to the node in that heading. To determine which node needs to be selected, A* uses the distance between the current node and the target node and moves to the node that has the smallest distance among adjacent nodes. It evaluates nodes by combining h (n), the estimated (cost) to that node that’s call heuristic value and g (n), the distance (cost) to get from that node to the goal node. The total cost f (n) = g (n) +h (n) is calculated for each adjacent node and the node with the smallest cost f (n) is selected as an adjacent (Risma Septiana, 2016). To determined heuristic value the distance between two nodes is simply determined by calculating the straight-line distance between the two nodes. It’s actually the Euclidean distance.

Though this might not the true distance. It can be shown that as long as the cost is never overestimated, it never overestimates the actual distance. The algorithm is acceptable i.e. it generates the optimal path.

1,5	2,5	3,5	4,5	5,5
1,4	2,4	3,4	4,4	5,4
1,3	2,3	3,3	4,3	5,3
1,2	2,2	3,2	4,2	5,2
1,1	2,1	3,1	4,1	5,1

Let consider the case for a simple 5X5 Matrix The start position is (1, 1) and goal node is (4, 1). The adjacent node only one, in this case, is (1, 2). There is no doubt, because (2,1),(2,2),(2,3),(2,4),(1,5) those are obstacle point .until the Robot reaches node (2, 4). Here there are two nodes (3, 4) and (3, 5). The adjacent node can be determined by evaluating the cost to the target node from both the nodes.

f (n) for node (3,4)

g (n)= 1+1+1+1=4 (assuming each square is 1X1 units)

h (n)=sqrt((4-3)^2 + (1-4)^2) = 4.16

f (n) = 4.16 + 4

f (n) for node (3, 5)

g (n) = 1+1+1+1+1=5 (assuming each square is 1X1 units)

h (n) = sqrt((4 - 3) ^2 + (1-5)^2) = 4.123

f(n)= 4.123+5

So, f(n) for (3,4) has been found to be the smallest of the two, hence the adjacent node is f(n). The robot can now move to the node (3, 4) and continue expanding the adjacent nodes as above until the goal node is reached.

Dead Lock:

1,5	2,5	3,5	4,5	5,5
1,4	2,4	3,4	4,4	5,4
1,3	2,3	3,3	4,3	5,3
1,2	2,2	3,2	4,2	5,2
1,1	2,1	3,1	4,1	5,1

If the robot runs into a dead end? Consider the terrain start with (1, 2) and goal is (5, 3).

From adjacent nodes what we have learned so far node (2, 2) will be chosen as the adjacent node instead of node (1, 3). The robot will traverse the route until it ends up at the node (5, 2).

If the robot runs into dead end than need to add a mechanism,

1) Explores the alternate routes once it lands on at a dead node.

2) Avoid to traversing paths that it knows leads to a dead node.

To overcome this problem, we can use two store lists one is OPEN and another is CLOSED. The OPEN list stores all consecutive paths that are yet to be explored while CLOSED list stores all paths that have been explored.

The OPEN list also stores all the parent node of each node. To trace the path from the Goal to the Start node it is used at the end, in this way generating the optimal path.

Consider the figure below. The start (1, 2) node has four adjacent nodes (1, 1), (2, 1), (1, 3) and (2,2). From the beginning calculation (2, 2) is chosen and the robot travels along that node, however ones it reaches the dead end, it discards the node (1, 1), (2, 1), (2, 2) and takes the adjacent (1, 3) and explores that path. Tracked back to the start node to get the complete path once the goal node is reached the parent nodes are found.

So, the optimal path is n (5,4) -> n (4,5) -> n (3,4)->n (2,4)->n (1,3)->n (1,2) the shortest path.

Algorithm. A∗ algorithm for path-planning.

1. First have to set beginning node in the OPEN list than compute the cost function f (n). [Where h (n) = 0 and g (n) =path cost between the goal and the start node, f (n) = g (n).]
2. Find the adjacent node with the least cost f(n) on the open list
3. Shift to the CLOSED list from the OPEN List which node has the smallest cost function f (n).
4. Incase two or more nodes have the same cost function f (n), then it will chose randomly resolve ties.
5. Algorithm will terminate if n is the goal node and to obtain the solution optimal path use the pointers. Otherwise, will continue until find the goal.
6. Determine all the adjacent nodes of n and calculate the cost function f (n) for every adjacent, not on the CLOSED list.
7. Associate with every adjacent, not on list OPEN list or CLOSED list the cost calculated and shift these on the OPEN list, placing pointers to n (n is the parent node).

8. Associate with any adjacent already on OPEN the smaller of the cost values just calculated and the previous cost value. (min (new f(n), old f(n))).
9. Repeat from step 2.

3.5 Environment Setup

Simulation perform in MATLAB R2018a, the whole environment is demonstrated in the (10x10) two-dimensional grid.

1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10
1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9
1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8
1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	10,7
1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	10,6
1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5
1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4
1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,3
1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2
1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1

Simulation perform on that environment with different scenario and robot can move diagonally. Red circle is obstacle, green circle is target point and blue point is starting.

.

CHAPTER 4

RESULTS AND DISCUSSION

Each and every node distance is same (1), robot move in the center of every node and also move diagonally on the grid. Left blue is the source node and all red circle is an obstacle.

4.1 Scenario Number: 1

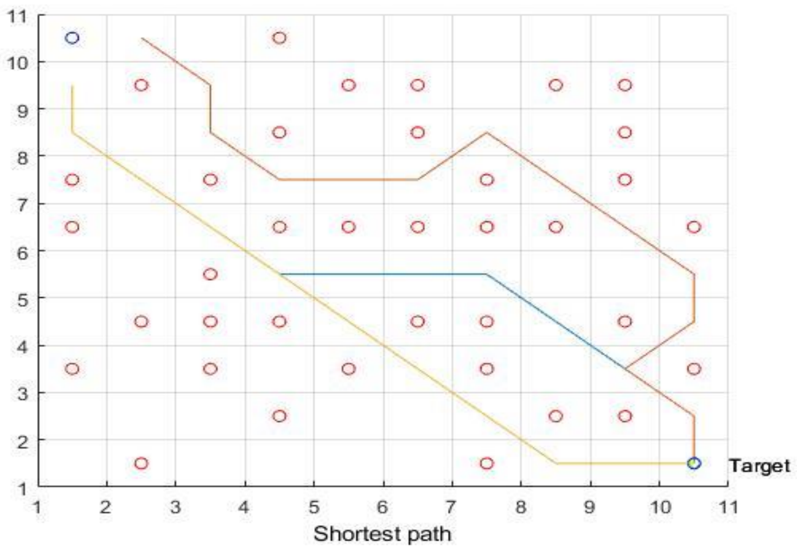


Figure 4. 1: Output with 20 obstacles

NB: Dijkstra=blue, Greedy BFS=radish, A*=yellow.

Table 4. 1: Environment with 37 obstacles

Algorithm	Source	Obstacle point	Goal	No of nodes travels
Dijkstra	1,10	(1,3),(1,6),(1,7),(2,1),(2,4),(2,9),(3,3),(3,4),(3,5),(3,7),(4,2),(4,4),(4,6),(4,8),(4,10),(5,3),(5,6),(5,9),(6,4),(6,6),(6,8),(6,9,(7,1),(7,3),(7,4),(7,6),(7,7),(8,2),(8,6),(8,9),(	10,1	12

		9,2),(9,4),(9,7),(9,8),(9,9),(10,3),(10,6)		
Greedy BFS	1,10	Same obstacle point	10,1	14
A*	1,10	Same obstacle point	10,1	11

From the table no: 1 and figure: 1 let obstacle point is [(1,3),(1,6),(1,7),(2,1),(2,4),(2,9),(3,3),(3,4),(3,5),(3,7),(4,2),(4,4),(4,6),(4,8),(4,10),(5,3),(5,6),(5,9),(6,4),(6,6),(6,8),(6,9,(7,1),(7,3),(7,4),(7,6),(7,7),(8,2),(8,6),(8,9),(9,2),(9,4),(9,7),(9,8),(9,9),(10,3),(10,6)] every algorithm has same source point (1,10) and same goal point(10,1) but here result show that A* is more optimal than Dijkstra and Greedy BFS cause A* travel 11 [(1,9),(1,8),(2,7),(3,6),(4,5),(5,4),(6,3),(7,2),(8,1), (9,1),(10,1)] node to reach goal but Dijkstra travel [(2,10),(3,9),(3,8),(4,7),(5,7),(6,7), (7,8),(8,7),(9,6),(10,5),(10,4),(9,3),(10,2),(10,1)] 12 and Greedy BFS 14 node [(1,9),(1,8), (2,7),(3,6),(4,5),(5,5),(6,5), (,7,5),(8,4),(9,3),(10,2),(10,1)].

4.2 Scenario Number: 2

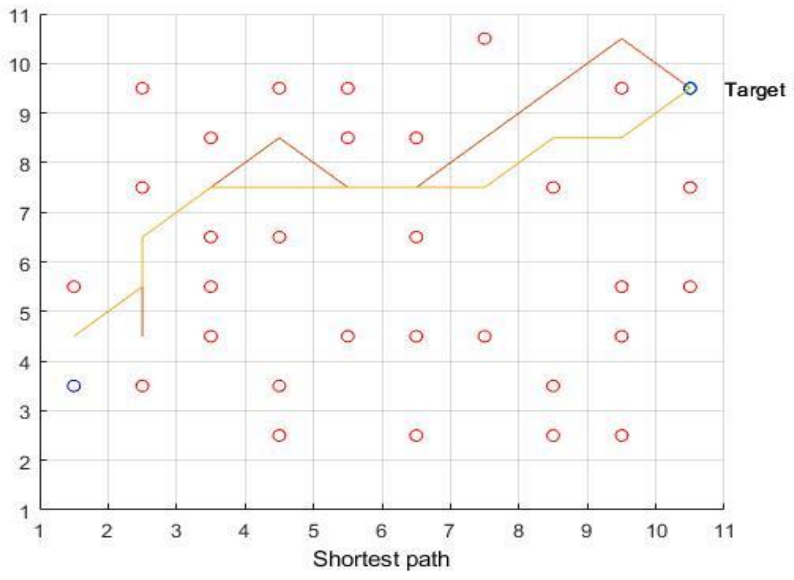


Figure 4. 2: Output with 30 obstacles

NB: Dijkstra= radish, Greedy BFS, A*=yellow.

Table 4. 2: Environment with 30 obstacles

Algorithm	Source	Obstacle point	Goal	No of nodes travels
Dijkstra	1,3	(1,5),(2,3),(2,7),(2,9),(3,4),(3,5),(3,6),(3,8),(4,2),(4,3),(4,6),(4,9),(5,4),(5,8),(5,9),(6,2),(6,4),(6,6),(6,8),(7,4),(7,10),(8,2),(8,3),(8,7),(9,2),(9,4),(9,5),(9,9),(10,5),(10,7)	10,9	11
Greedy BFS	1,3	Same obstacle point	10,9	11
A*	1,3	Same obstacle point	10,9	11

From the table no: 2 and figure: 2 let obstacle point is[(1,5),(2,3),(2,7),(2,9),(3,4),(3,5),(3,6),(3,8),(4,2),(4,3),(4,6),(4,9),(5,4),(5,8),(5,9),(6,2),(6,4),(6,6),(6,8),(7,4),(7,10),(8,2),(8,3),(8,7),(9,2),(9,4),(9,5),(9,9),(10,5),(10,7)]every algorithm has same source point (1,3) and same goal point(10,9), here result show that A*, Dijkstra and Greedy BFS provide same result, cause Dijkstra, A* and Greedy BFS travel same number of (11 node) node to reach goal but A* travel this minimum way[(2,4),(2,5),(2,6), (3,7),(4,7),(5,7),(6,7),(7,7),(8,8),(9,8),(10,9)] where Dijkstra, Greedy BFS and (1,4),(2,5),(2,6),(3,7),(4,8),(5,7),(6,7),(7,8)(8,9),(9,10),(10,9).

4.3 Scenario Number: 3

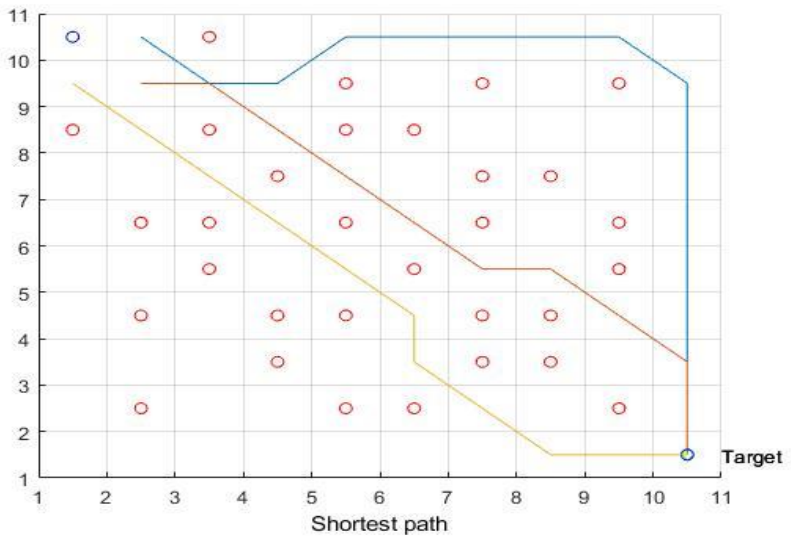


Figure 4. 3: Output with 31 obstacles

NB: Dijkstra=blue, Greedy BFS=red, A*=yellow.

Table 4. 3: Environment with 31 obstacles

Algorithm	Source	Obstacle point	Goal	No of nodes travels
Dijkstra	1,10	(1,8),(2,2),(2,4),(2,6),(3,5),(3,6),(3,8),(3,10),(4,3),(4,4),(4,7),(5,2),(5,4),(5,6),(5,8),(5,9),(6,2),(6,5),(6,8),(7,3),(7,4),(7,6),(7,7),(7,9),(8,3),(8,4),(8,7),(9,2),(9,5),(9,6),(9,9)	10,1	17
Greedy BFS	1,10	Same obstacle point	10,1	11
A*	1,10	Same obstacle point	10,1	11

From the table no: 3 and figure: 3 let obstacle point is

[(1,8),(2,2),(2,4),(2,6),(3,5),(3,6),(3,8),(3,10),(4,3),(4,4),(4,7),(5,2),(5,4),(5,6),(5,8),(5,9),(6,2),(6,5),(6,8),(7,3),(7,4),(7,6),(7,7),(7,9),(8,3),(8,4),(8,7),(9,2),(9,5),(9,6),(9,9)] every algorithm has same source point (1,10) and same goal point(10,1) but here result show that A*, Greedy BFS is more optimal than Dijkstra cause Greedy BFS [(2,9),(3,9),(4,8),(5,7),(6,6),(7,5),(8,5),(9,4),(10,3),(10,2),(10,1)], A* [(1,9),(2,8),(3,7),(4,6),(5,5),(6,4),(6,3),(7,2),(8,1),(9,1),(10,1)] travel same number of (11) node to reach goal but Dijkstra travel 17 node [(2,10),(3,9),(4,9),(5,10), (6,10),(7,10), (8,10),(9,10),(10,9),(10,8),(10,7),(10,6),(10,5),(10,4),(10,3),(10,2),(10,1)].

4.4 Scenario Number: 4

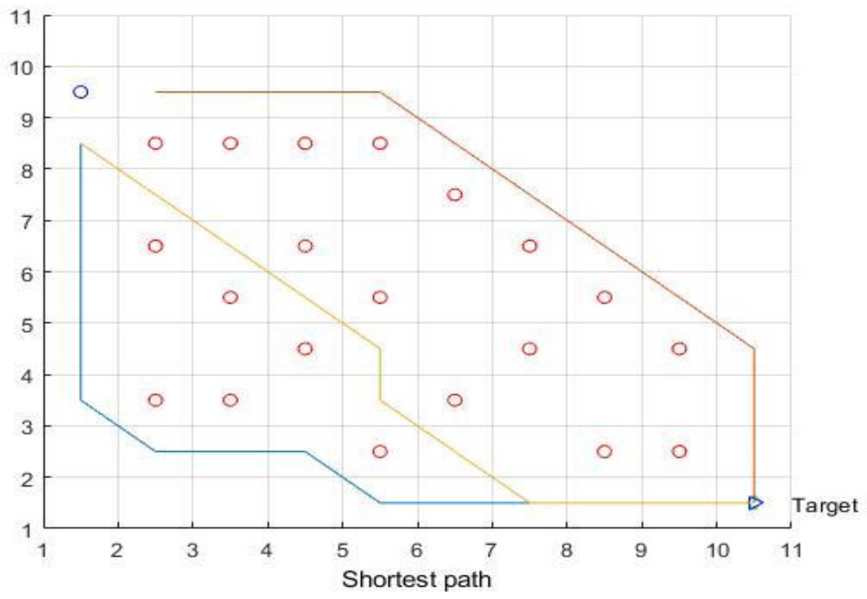


Figure 4. 4: Output with 20 obstacles

NB: Dijkstra=blue, Greedy BFS=red, A*=yellow.

Table 4. 4: Environment with 20 obstacles

Algorithm	Source	Obstacle point	Goal	No of nodes travels
Dijkstra	1,10	(2,3),(2,6),(2,8),(3,3),(3,5),(3,8),(4,4),(4,6),(4,8),(5,1),(5,5),(5,8),(6,3),(6,7),(7,3),(7,4),(7,6),(8,2),(8,5),(9,2),(9,4)	10,1	15
Greedy BFS	1,10	Same obstacle point	10,1	12
A*	1,10	Same obstacle point	10,1	11

From the table no: 4 and figure: 4 let obstacle point is [(2,3),(2,6),(2,8),(3,3),(3,5),(3,8),(4,4),(4,6),(4,8),(5,1),(5,5),(5,8),(6,3),(6,7),(7,3),(7,4),(7,6),(8,2),(8,5),(9,2),(9,4)]every algorithm has same source point (1,10) and same goal point(10,1) but here result show that A* is more optimal than Dijkstra, Greedy BFS cause Dijkstra travel 15 [(1,8),(1,7),(1,6),(1,5),(1,4),(1,3),(2,2), (3,2),(4,2),(5,1), (6,1),(7,1),(8,1), (9,1), (10,1)] node Greedy BFS travel 12 node [(2,9),(3,9),(4,9), (5,9),(6,8), (7,7),(8,6), (9,5),(10,4), (10,3), (10,2),(10,1)], but A* travel 11 [(1,8),(2,7),(3,6),(4,5), (5,4),(5,3), (6,2),(7,1),(8,1),(9,1),(10,1)] node to reach goal.

4.5 Discussion:

In the above figure and result proved that while obstacle is less every algorithm will travel same number of node but while obstacle density is more A-star algorithm will show the more optimal than dijkstra and greedy BFS.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Findings

There are so many real-life problems that can manage by the path planning algorithm. For automated piloting use the map planning algorithm, but all algorithm is not optimal, optimization depends on environment most of the time. Sometimes dijkstra, Greedy BFS and A-star show the same result depend on less obstacle density and computation time is not same. In every critical scenario A-star gives the best performance.

5.2 Recommendations for Future Works

In UAV system, traffic navigation path planning problem is solved with optimization algorithm. Find out the work station where A-star gives the best performance. And also find out a new solution for dynamic environment with an algorithm. Basically extending of A* algorithm for dynamic environment.

REFERENCES

Alican Bozyiğit, G. A. (2017). Public Transport Route Planning: Modified Dijkstra's Algorithm. *(UBMK'17)2nd International Conference on Computer science and engineering* (p. 502). IEEE.

Amos Tversky, D. K. (1982). *Judgment under Uncertainty: Heuristics & Biases.* Cambridge, UK: Cambridge University Press.

B.Moses Sathyaraj, L. &. (2008). Multiple UAVs path planning algorithms:A scomparative study.

Chimanga, K. M. (2016). Application of best first search algorithm to demand control. *IEEE.* Livingstone.

Dijkstra. (1959). A Note on Two problems in connexion with Graphs. In E.W.Dijkstra, *Numerische Mathematik 1* (pp. 269-271).

Fan, S. (2010). Improvement of Dijkstra’s Algorithm and Its Application in Route Planning. *IEEE Conferences.* IEEE.

Fu Xiao-wei, L. Z.-g. (2010). Path planning for UAV in radar network area*. *IEEE Conferences* (p. 260). Wuhan: IEEE.

Gopikrishnan, S. &. (2011). Path Planning Algorithms: A comparative study. *National Conference on Space Transportation Systems.* Thiruvananthapuram.

Guohao, &. L.-T. (2018). A New Path Planning Algorithm Using a GNSS Localization Error Map for UAVs in an Urban Area. *Journal of Intelligent & Robotic Systems*, 1-17.

Haifeng Wang, J. Z. (2014). HAS:Hierarchical A-Star algorithm for big map navigation in special areas. *IEEE Conferences* (p. 222). Guangzhou: IEEE.

I. S. AlShawi, L. Y. (2012). LIFETIME ENHANCEMENT IN WIRELESS SENSOR NETWORKS USING FUZZY APPROACH AND A-STAR ALGORITHM. *IET Conference on Wireless Sensor Systems (WSS 2012).* London: ITE.

Iram Noreen, A. K. (2016). Optimal Path Planning for Mobile Robots Using Memory Efficient A*. *2016 International Conference on Frontiers of Information Technology (FIT).* Islamabad: IEEE.

Jehn-Ruey Jiang, H.-W. H.-H.-Y. (2014). Extending Dijkstra’s Shortest Path Algorithm for Software Defined Networking. *IEEE Conferences* (p. 1). IEEE.

Jingang, Y. (2010). Application of A* algorithm in traffic Navigational system. *IEEE conference.* IEEE.

Judith D.Sally & Paul J.Sally, J. (2007). Chapter 3: Pythagorean triples. In *Roots to research: a vertical development of mathematical problems* (p. 63). American Mathematical Society.

Kang. (2008). Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm. *IEEE Conferences* (p. 1002). IEEE .

KHAILI, M. E. (2014). Path Planning in a Dynamic Environment by the approach of the sliding on edge. *International Journal of Advanced Computer Science and Applications, 5*(8).

Krause, E. F. (1975). *Taxicab Geometry: An Adventure in Non-Euclidean Geometry.* New york: Dover Publication.

Lu, L. (2013). Polog with best first search. *IEEE Conference.* IEEE.

Mackworth, D. P. (2017). *Artificial Intelligence: Foundations of Computational Agents.* Cambridge University Press.

Mahmud. (2012). A Greedy Approach in Path Selection for DFS Based Maze-map Discovery Algorithm for an Autonomous Robot. *IEEE Conferences* (p. 546). Chittagong: IEEE.

Marinescu. (2010). Best-First vs. Depth-First AND/OR Search for Multi-objective Constraint Optimization. *IEEE Conference* (p. 439). Arras: IEEE.

Mohamed. (2014). Path Planning in a Dynamic Environment by the approach of the sliding on edge. *International Journal of Advanced Computer Science and Applications, 5*(8), 86.

Nadira Jasika, N. A. (2012). Dijkstra's shortest path algorithm serial and parallel execution performance analysis. *2012 Proceedings of the 35th International Convention MIPRO* (p. 1811). Opatija: IEEE.

Pearl, J. (1984). *Intelligent Search Strategies for Computer Problem Solving.*

Peter E.Hart, N. J. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 100 - 107.

Phanthong, T. T. (2014). Application of A* algorithm for real-time path re-planning of an unmanned surface vehicle avoiding underwater obstacles. *Journal of Marine Science and Application, 13*(1), 105-116.

Radmanesh. (2018). Overview of path planning and obstacle Avoidance Algorithms for UAVs:A Comparative Study. *World Scientific Conference.* World Scientific.

Risald, A. E. (2017). Best Routes Selection Using Dijkstra And Floyd-Washall Algorithm. *2017 International Conference on Information & Communication Technology and System (ICTS)* (p. 155). Surabaya: IEEE.

Risma Septiana, I. S. (2016). Evaluation Function Effectiveness in Wireless Sensor Network Routing using A-star Algorithm. *IEEE Conferences* (pp. 1-5). Bandung: IEEE.

Shaimaa Ahmed, A. M. (2016). Energy Efficient Path Planning Techniques for UAV-based Systems with Space Discretization. *2016 IEEE Wireless Communications and Networking Conference.* Doha: IEEE.

Tianyou Chen, G. Z. (2018). Unmanned Aerial Vehicle Route Planning Method Based on A Star Algorithm. *IEEE Conferences* (p. 1510). IEEE.

Tianyou, G. J. (2018). Unmanned aerial vehicle route planning method based on a star algorithm. *IEEE.* Wuhan.

Valenzano, S. S. (2014). *Adding Exploration to Greedy Best-First Search.* Toronto: Affiliation: University of Alberta.

Wang. (2011). Application of Dijkstra algorithm in robot path planning. *IEEE.* Hohhot.

wikipedia. (2018, 11 10). Retrieved from https://en.wikipedia.org/wiki/A*_search_algorithm#cite_note-nilsson-3

Wikipedia. (2018, 11 13). Retrieved from https://en.wikipedia.org/wiki/Heuristic#cite_note-6

Woo-Jin Seo, S.-H. O.-H. (2009). An Efficient Hardware Architecture of the A-star Algorithm for the Shortest Path Search Engine. *IEEE Conferences* (p. 1499). Seoul: IEEE.

Xu Yulong, W. X. (2016). Comparative Study on the Optimal Path Problem of Wireless Sensor Networks. *IEEE Conference* (p. 2234). Harbin: IEEE.

ZeFang He, L. Z. (2017). The comparison of four UAV path planning algorithms based on geometry search algorithm. *IEEE Conferences* (p. 33). IEEE.