



**Daffodil**  
*International*  
**University**

# **Improved A\* Search Algorithm for Path Planning of Mobile Robots**

By

**Md. Ishrak Tawsif Nirob**  
**ID: 151-35-882**

A thesis submitted in partial fulfillment of the requirement for the degree of  
Bachelor of Science in Software Engineering

**Department of Software Engineering**  
**DAFFODIL INTERNATIONAL UNIVERSITY**

Spring 2018

Copyright © 2019 by Daffodil International University

# APPROVAL

This Thesis titled “**Improved A\* Search Algorithm for Path Planning of Mobile Robots**”, submitted by **Md. Ishrak Tawsif Nirob, 151-35-882** 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 B.Sc. in Software Engineering and approved as to its style and contents.

## BOARD OF EXAMINERS

-----  
**Dr. Touhid Bhuiyan**  
**Professor and Head**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University

**Chairman**

-----  
**Dr. Md. Asraf Ali**  
**Associate Professor**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University

**Internal Examiner 1**

-----  
**Mohammad Khaled Sohel**  
**Assistant Professor**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University

**Internal Examiner 2**

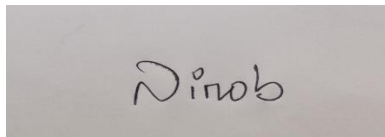
-----  
**Prof Dr. Mohammad Abul Kashem**  
**Professor**  
Department of Computer Science and Engineering  
Faculty of Electrical and Electronic Engineering  
Dhaka University of Engineering & Technology, Gazipur

**External Examiner**

## Thesis Declaration

I hereby declare that, this Thesis Report has been done under the supervision of Md. Anwar Hossain, Senior Lecturer, Department of Software Engineering, Faculty of Science and Information Technology, Daffodil International University. I also declare that this hasn't been submitted elsewhere for award of any degree.

### Submitted By



.....

Md. Ishrak Tawsif Nirob

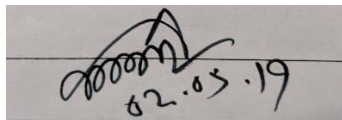
ID: 151-35-882

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

### Certified By



.....

Md. Anwar Hossain

Senior Lecturer

Department of software Engineering

Faculty of Science and Information Technology

Daffodil International University

## **Acknowledgement**

Alhamdulillah, all praises to the Almighty Allah who gives me the ability, motivation, patience and wisdom to complete this research work.

I would like to propagate my gratefulness and gratitude to my respectful supervisor Md. Anwar Hossain. This research would not be possible without him. His excellent guidance, motivation, caring, patience and providing me with an excellent facilities and environment for doing this research. I am also grateful to my parents for their support and care throughout the entire period.

## ABSTRACT

Path planning has a wide range of application in video games, GPS based applications, mobile robotic etc. Autonomous mobile robot is used for doing hazardous task which is dangerous and hard to handle for people. Efficient path planning is basic requirement for mobile robot. Aiming at problem of path finding an improved version of A\* algorithm is proposed. In A\* algorithm the cost of cells is only calculated by the sum of actual cheapest cost of a cell from starting cell and a heuristic value which is estimated cheapest cost from the cell to goal cell. But adding the heuristic value of parent node to cost function can improve the performance of algorithm. Adding heuristic value of parent node with the cost of a cell help the algorithm to find it path more purposefully. A robot can move to any cell adjacent to it. So there are eight possible move from a particular cell. And all of them is not optimal. So by selecting the optimal cell always helps to reach the goal soon. In an improved version of A\* algorithm heuristic value of parent node of a particular node is used to evaluate the cost of a function which perform better than A\* algorithm. In this research an new improved version of A\* algorithm is proposed where priority of optimal cell is reduced to get more priority and heuristic value of parent node of current node is added with the cost of cells. To increase the priority of optimal cell a value is added with the rest of cell. This value is calculated with the help of heuristic value. If a cell is closer to the goal than this value will be lower for it. This method is efficient than A\* algorithm and previous version of improved A\* algorithm. It can find path sooner than the typical A\* algorithm and previous version of improved A\* algorithm.

## Table of Contents

Approval	
Declaration	
Acknowledgement	
Abstract	i
Table of Contents	ii
List of Figures	iii
Chapter 1: Introduction	1 – 3
1.1 Overview	1
1.2 Research Motivation	1
1.3 Problem Statement	1
1.4 Research Question	1
1.5 Research Objectives	2
1.6 Research Scope	2
1.7 Thesis Organization	2
Chapter 2: Background Study	4 – 8
2.1 A* algorithm	4
2.2 Dijkstra Algorithm	5
2.3 Mobile robot	5
2.4 Environment model built by grid method	6
2.5 Path planning of mobile robot	7
Chapter 3: Literature Review	9 - 10
Chapter 4: Methodology	11 – 24
4.1 A* algorithm introducing parent node	11
4.2 Procedure of improved A* algorithm	12
4.3 Demonstration of improved A* algorithm	13
4.4 Modification of improved A* algorithm	19
4.5 Procedure of newly modified improved A* algorithm	23
Chapter 5: Result and Discussion	25 – 31
Chapter 5: Conclusions and Recommendations	32

## List of Figures

Figure No	Figure Name	Page
2.1	Environment Model built by Grid Model	8 - 9
4.1	Eight different cell from where a cell can be visited	15
4.2	Procedure of improved A* algorithm (step 4)	18
4.3	Procedure of improved A* algorithm (step 2)	19
4.4	Procedure of improved A* algorithm (step 3)	20
4.5	Procedure of improved A* algorithm (planned path)	21
4.6	Demonstration of traversed cell for improved A* algorithm	23
4.7	Demonstration of traversed cell for modified version improved A* algorithm	24

# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

Path planning for mobile robot can be done by various algorithm like dijkstra algorithm, breadth first search (bfs), depth first search (dfs), A\* search. Two technique covers all approaches in robot path planning: (i) global path planning or off-line path planning and (ii) local path planning or on-line path planning[9]. A\* algorithm is a heuristic value based offline path planning algorithm. A\* algorithm doesn't expand cells randomly like bfs algorithm rather it order the available cell based on their cost. Cost of a cell is sum of its actual cost from starting cell and heuristic value which is estimated cheapest cost from this cell to goal cell. A\* algorithm is a popular algorithm for path planning. For path planning of a mobile robot A\* is most popular. A mobile robot can be autonomous or non autonomous. Autonomous robot can move in any given environment by its own. So to select path it need algorithm like A\* algorithm. Mobile robots are mostly used in many hazardous industrial fields which is hard or dangerous for people to handle or control.

### 1.2 Motivation of the Research

Demand of mobile robots are increasing day by day. Autonomous mobile robots are performing an important role in today's world. In the field of automation path planning is crucial. Moreover path planning is fundamental component of many softwares and hardware devices. An effective and efficient path planning improves the performance of a robot. There are numbers of technique for path planning but still there are chances to improve the performance of those technique. Some of those technique may work efficiently for most of the cases but in some critical condition performance of these techniques can be improved. An improved version of an algorithm handle this kind of cases and improve the performance of the algorithm. In this thesis an improved version of A\* algorithm has been proposed.

### 1.3 Problem Statement

The process of finding an accurate and shortest path between two end point is path finding. And path planning for a mobile robot is the process of finding a shortest and accurate path for the robot from the starting point of the robot to the destination. There are several methods for path planning. Grids and artificial potential field are the traditional method and new methods includes fuzzy logic or neural network. The challenge of path planning is to find the shortest path as fast as possible by avoiding obstacles cleverly and choose the path more purposefully.

### 1.4 Research Questions

Path planning is one of the fundamental requirement for autonomous mobile robot. Different path planning algorithm can solve this problem. But,

1. Can these algorithm avoid obstacle more cleverly?



2. Can these algorithm find the shortest path more efficiently?

### 1.5 Research Objectives

- Give more priority to the cells which are more likely to find a shortest path.
- To add a certain value with less likely cells in order to decrease their priority in the queue.

### 1.6 Research Scope

This study mainly focuses on efficient path planning. Path planning is mostly important for automated robot. Path planning is also important for autonomous vehicle. Efficient path planning can increase performance of an autonomous robot or an autonomous vehicle.

There are two kind of technique for path planning that actually cover all other approaches in path planning –

- i) Global path planning or Offline path planning.
- ii) Local path planning or Online path planning.

### 1.7 Thesis Organization

Chapter 1 gives the synopsis of this thesis. Background study contains topics that was needed to learn before start working on this paper. Moreover my motivation of doing this research, my research question, research objective is cleared well in this part.

Chapter 2 contains all the topic that is needed to study before starting my research. What is A\* algorithm, how it works, what is dijkstra algorithm, how it works is written in this part. What is path planning and mobile robot and why these needed is also discussed in this chapter.

Chapter 3 consist the relatable work on path planning of robots. Different paper has different approach to solve path planning problem and some of them is denoted here.

Chapter 4 is the discussion of improved A\* algorithm. How this improved algorithm work and why it performs better than A\* algorithm is discussed here.

Chapter 5 the result of the research. Comparison among proposed algorithm of this paper, a previously proposed improved A\* algorithm and A\* algorithm is shown in this chapter.

## CHAPTER 2

### Background Study

#### 2.1 A\* algorithm

A\* algorithm potentially searches a large area of the map. A\* algorithm is based on Dijkstra algorithm and BFS algorithm and widely used in pathfinding. It is the most known form of BFS. A\* algorithm can avoid unnecessary expensive path and find the optimal path more purposefully. This algorithm uses heuristic function to lead itself in process of pathfinding. A\* algorithm uses a function that calculate the cost of a node and according to this cost it evaluate the quality of nodes. The algorithm always choose the minimum costly node first. Following function calculate the cost for A\* algorithm –

$$f(n) = g(n) + h(n) \quad \dots\dots\dots (1)$$

In this formula if n is node on the path then –

- ▶ ***g(n)*** refers to the cost of the path from start node to node n.
- ▶ ***h(n)*** refers to the estimated cheapest cost of the path from node n to goal node.
- ▶ ***f(n)*** refers to estimated cheapest cost of path from start node to goal node through node n.

A\* algorithm always choose the node with lowest *f(n)*. The heuristic value of a node *h(n)* always changes according to the path choice.

There is no fixed method to calculate heuristic value of a node. The heuristic value of a node can be calculated by several formula. For example –

a) Manhattan Distance:

The formula for Manhattan distance is:

$$dis (x1,x2 \text{ and } y1,y2) = ( (abs(x1 - x2) + (abs(y1-y2) ) \quad \dots\dots\dots (2)$$

b) Euclidean Distance:

The formula for Euclidean distance is:

$$dis (x1,x2 \text{ and } y1,y2) = \sqrt{(x1-x2)^2 + (y1-y2)^2} \quad \dots\dots\dots (3)$$

Both of these formula reflects the estimated cheapest cost of the path from node n to goal node.

In searching process A\* algorithm usually uses two lists- open list and closed list. open list contains those nodes which has not been accessed yet and closed list tracks which nodes has been discovered. A\* traverse by expanding all adjacent nodes (in 4 directions or in 8 directions).

The searching process of A\* algorithm:

- 1) Add start node to open list.
- 2) Take the optimal node from open list. Let it be current node. If this node is goal node go to step 5.  
Else remove current node from open list and add it to closed list.
- 3) Expand all adjacent nodes of current node. If this node contains an obstacle then ignore it else calculate their cost by using formula (1).Add them to open list.
- 4) Go to step 2.
- 5) Stop searching.

## 2.2 Dijkstra Algorithm:

Dijkstra algorithm is another algorithm that hugely used to find shortest path between nodes in a given graph or map. This algorithm actually can solve single path shortest path problem means it can only find path from a particular source or node to any other nodes. This algorithm regularly maintain a set of nodes with their cost, whose final shortest path weights from source have been determined. And then this algorithm repeatedly select node minimum cost and expand all it's adjacent nodes.

The searching process of dijkstra algorithm:

- 1) Initialize the source
- 2) Set initial cost of source node.
- 3) Initialize a list and add source node along with it cost in the list.
- 4) If list is not empty then take the node with minimum cost from the list.
- 5) Expand all child of taken node and update their value and add to the list.
- 6) If list is not empty go to step 4.

## 2.3 Mobile robot:

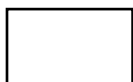
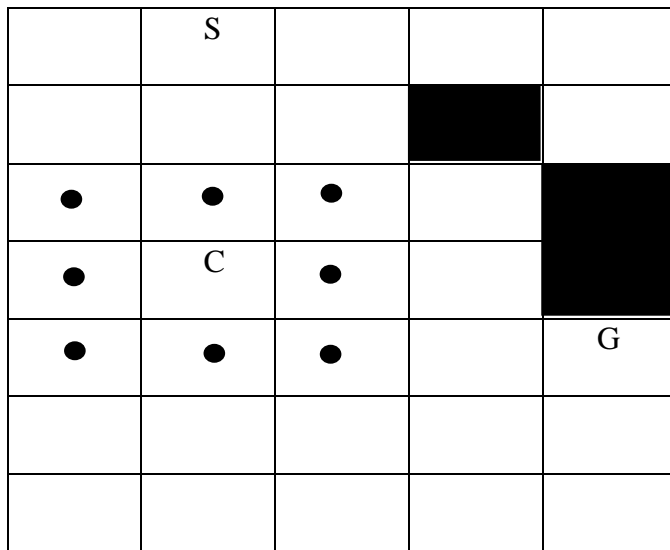
A mobile robot can be described as a machine that can move automatically and has the capability to move around any given environment. So a mobile robot is not fixed to one physical environment. Mobile robot require both artificial intelligence and physical robotics. Mobile robots can be autonomous or non-autonomous –

- a) Autonomous mobile robot: An autonomous mobile robot can move in an unknown environment without any external guidance (physical or electro mechanical).
- b) Non autonomous mobile robot: A non-autonomous mobile robot external guidance. It need any kind of guidance to move in an environment.

## 2.4 Environment model built by grid method:

In 1968, W. E. Howden first proposed the grid method. This is the most well developed environment modeling. This model is widely used because it is brief and effective. This method is applied to many algorithm. Main idea of this method is to divide the a finite area into small square sized grid. If the horizontal direction is divided into N interval and vertical direction divided into M interval then the whole area can be described as environment map by the  $N \times M$  grids. This model assumed robot will move in this finite area of  $N \times M$  grids with finite numbers of obstacle.

Following figures demonstrate the grid method environment model-



Feasible Grid.



Obstacle Grid.

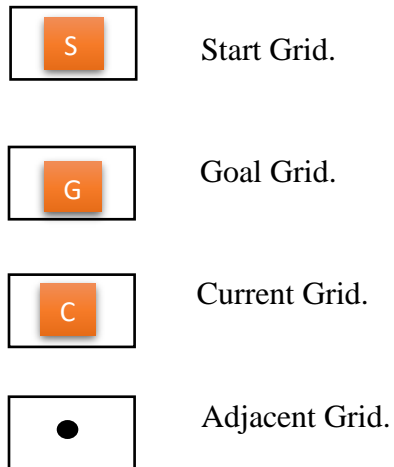


Fig 2.1 Environment model built by grid method.

## 2.5 Path planning of mobile robot:

Path planning is one of the basic requirement for mobile robot. A robot's path planning is a sequence of rotation and translation from the source position to destination position by avoiding cells with obstacle.

There are two kind of technique for path planning that actually cover all other approaches in path planning

–

- i) Global path planning or Offline path planning.
- ii) Local path planning or Online path planning.

Global path planning work on known environment and do not work well in unknown or dynamic environment. But this approach is very efficient to find optimal path.

Local path planning does not require any information about environment beforehand and finds a possible optimal path based on available information that it can gather on its way to goal.

There different type of path planning algorithm for mobile robot-

### 1) Classical Algorithm

- i) Cell decomposition.
- ii) Potential field.
- iii) Sampling based method.
- iv) Sub goal network.

2) Heuristic based algorithm

- i) Neural Network (NN).
- ii) Fuzzy Logic. (FL)
- iii) Nature Inspired Algorithm.
- iv) Hybrid Algorithm.

## Chapter 3

### LITERATURE REVIEW

Path planning is basic requirement for different game, different GPS application, robotics etc. Though there are a lot of algorithm or technique that can solve this problem but still there are a lot of scope to work on it.

In a research paper František Duchon et al. (2014) compared four different algorithm that can solve path planning problem on a particular environment [2]. Those four algorithms are A\* algorithm, Basic Theta\*, Focused D\* and jump point search (jps) algorithm. And they found jps as the superior of other three algorithm.

Thi Thoa Mac et al. (2016) discussed different algorithm of different technique that can solve path planning problem [3]. Their study covered Neural network (NN), Fuzzy Logic (FZ) and some most common nature inspired algorithm which are GA,PSO,ACO in robot path planning applications. They also showed the pros and cons of those algorithm.

M.S.Ganeshmurthy et al. (2015) proposed a simulated annealing based solution for dynamic environment [4]. Their proposed method is their algorithm will first generates a path to goal from starting cell off line. Then the algorithm will try to update the path until a certain period. But if an obstacle appear in the planned path then the algorithm will generate a new path excluding that cell.

Mingxiu Lin et al. (2017) proposed an improved A\* algorithm [1]. They updated the formula of calculating cost of A\* algorithm. The added the heuristic value of parent node of current node. And they also introduced weight to heuristic function.

Akshay Kumar Guruji et al. (2016) introduces a A\* algorithm which is Time-Efficient [5]. At first a photo of robots environ is taken then they process this image to find the start point, goal point and cells with obstacle. Depending on the distance between starting and goal cell the algorithm can either use A\* algorithm or a process called switching phase.



Ke Da1 et al. (2017) introduces variable step length in A\* algorithm [6]. From any cell unlike typical A\* algorithm this algorithm doesn't always use length 1 to step forward. Rather they take a length and let this length be radius of a circle then try to jump to any cell that seems efficient.

Daniel Drake et al. (2018) proposed a method that can solve path planning problem with moving goal[7]. The algorithm start it process from goal and assign id to all expanded cells. And by following this process it initiate a path. If sometime the goal changed then it try to find the goal around the previous goal location and try to initiate a new path.

**CHAPTER 4**  
**RESEARCH METHODOLOGY**

**4.1 A\* algorithm introducing parent node:**

An A\* algorithm can be improved by adding heuristic value of parent node of current node[1]. Usual cost function of A\* algorithm is –

$$f(n) = g(n) + h(n)$$

In the usual formula of calculating the cost of current node only the actual cost from start node to current node and estimated cheapest cost from current node to destination is added. But they also add the heuristic value of the parent node of current node. So their proposed formula is

$$f(n) = g(n) + h(n) + h(p) \dots\dots\dots (4)$$

A robot on grid can move to eight direction. So from a particular cell there are eight possible cells where robot can go in the next move. Consequently there are eight possible cell from where a cell can be visited by robot. That means there are eight possible parent cell for a particular cell.

#### 4.2 Procedure of previous improved A\* algorithm:

```
function Astar  
  
  initialize open list and closed list  
  add starting point into open list  
  
  while open list not empty  
  
    choose a node from open list  
    delete the node  
    if it is the terminal point then  
    get the path  
  
    for each child point Y  
      if Y is not in either list then  
        find the value of Y  
        add Y into open list  
      else if Y is in open list then  
        compare the value of Y  
        update the list  
      else if less than value of closed list then  
        update the value of closed list  
        remove Y to open list  
  
      end if  
    end if  
    end if  
  
    end for  
    add x to closed list  
    rank the nodes in open list  
  
  end while  
  
end of function Astar
```

### 4.3 Demonstration of improved A\* algorithm:

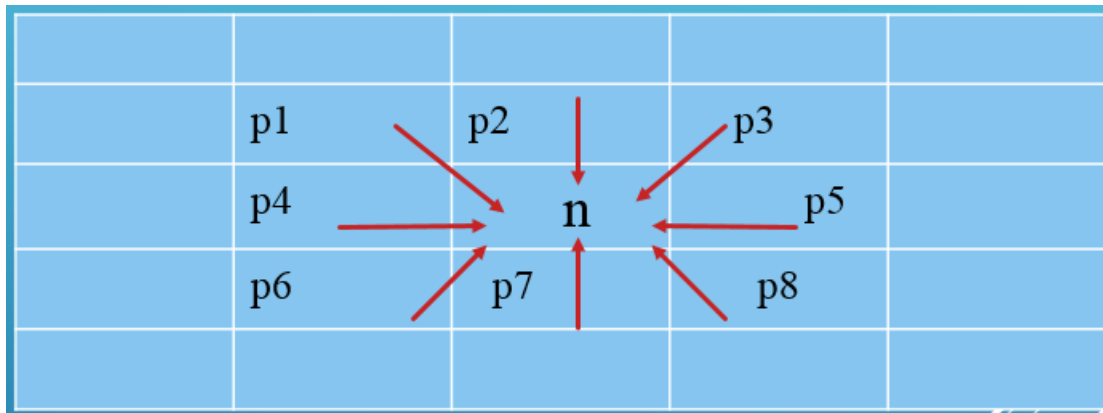


Fig 4.1 Eight different cell from where a cell can be visited.

Figure 2 shown all eight possible cells from where a cell n can be visited. So it can be said that for a given time there can be multiple information for a particular cell n. For example –

- 1) Current cell is n, parent cell is p1 and cost is c1.
- 2) Current cell is n, parent cell is p2 and cost is c2.
- 3) Current cell is n, parent cell is p3 and cost is c3.
- 4) Current cell is n, parent cell is p4 and cost is c4.
- 5) Current cell is n, parent cell is p5 and cost is c5.
- 6) Current cell is n, parent cell is p6 and cost is c6.
- 7) Current cell is n, parent cell is p7 and cost is c7.
- 8) Current cell is n, parent cell is p8 and cost is c8.

It is clear that in the list there can be more than one information for a particular cell. If heuristic value of parent is not added then this multiple information can be expanded in some random order because cost of different information can be same. And this will lead to unnecessary cell expansion.

So they proposed to add heuristic value of parent node of the current cell to give more priority to the information which may reach to goal sooner than other. But this makes the heuristic value large than usual as two heuristic value ( $h(n)$  and  $h(p)$ ) added. This leads to another problem that large heuristic value may lead the algorithm to local optimal solution. So they multiply the summation of heuristic value ( $h(n) + h(p)$ ) with a fraction value.

Let's consider the following grid –

13,14 Start cell	13,15 obstacle	13,16 obstacle	13,17 obstacle
14,14	14,15	14,16	14,17
15,14	15,15 Current cell	15,16	15,17
16,14	16,15	16,16	16,17

In this grid robot's starting cell is (13,14), current position is (15,15). And cells with obstacles are (13,15); (13,16);(13,17). And goal cell is (50,50).

Following grid represents  $g(n)$  [actual cheapest cost to reach a cell from starting cell] for every adjacent cells of current cell -

13,14 Start cell	13,15 obstacle	13,16 obstacle	13,17 obstacle
14,14 $g(n) = 1$	14,15 $g(n) = 2$	14,16 $g(n) = 2$	14,17
15,14 $g(n) = 2$	15,15 $g(n) = 2$	15,16 $g(n) = 3$	15,17
16,14 $g(n) = 3$	16,15 $g(n) = 3$	16,16 $g(n) = 3$	16,17

Following grid represents  $h(n)$  [estimated cheapest cost to reach goal from a cell  $n$ ] for every adjacent cells of current cell –

13,14	13,15 obstacle	13,16 obstacle	13,17 obstacle
14,14 $h(n) = 72$	14,15 $h(n) = 71$	14,16 $h(n) = 70$	14,17
15,14 $h(n) = 71$	15,15 $h(n) = 70$	15,16 $h(n) = 69$	15,17
16,14 $h(n) = 70$	16,15 $h(n) = 69$	16,16 $h(n) = 68$	16,17

As we know there can be eight possible parent for a certain cell. The following table demonstrates the cost for our current cell (15,15) for all possible parent –

parent of (15,15)	$h(p)$ heuristic value of parent cell	$g(n)$ (cost from start cell to current cell (15,15) )	$h(n)$ (heuristic value of cell (15,15) )	$h(n)+g(n)$	$h(n)+g(n)+h(p)$
(14,14)	72	2	70	72	144
(14,15)	71	2	70	72	143
(14,16)	70	2	70	72	142
(15,14)	71	2	70	72	143
(15,16)	69	2	70	72	141
(16,14)	70	2	70	72	142
(16,15)	69	2	70	72	141
(16,16)	68	2	70	72	140

So in the table above we can see that  $h(n) + g(n)$  is same for all possible parent of current cell. That means algorithm gives same priority to each of these cells. That's why it can lead the algorithm to expand unnecessary cells.

But if we add value of  $h(p)$  with the previous value of  $h(n) + g(n)$  then some cell will get more priority than other. And algorithm may find the shortest path by expanding less cells than the original A\* search algorithm.

Following grids will demonstrate how this algorithm expand cells-

Lets consider a grid where start cell is (2,1) and the goal is on cell (5,3). There are obstacle on cell (1,3); (2,2) and (3,3).

**Initiation of process:**

Calculate the cost of starting cell (2,1) and add it to openlist along with it's cost.

There will be only one information in closed list-

Cell – (2,1) and cost =  $g(2,1) + h(2,1) + h(p) = 0 + 5 + 0 = 5$ .

**Step 1:**

Row/column	1	2	3	4	5
1	↑		obstacle		
2	S	obstacle			
3	↓		Obstacle		
4					
5			G		

Fig 4.2 Procedure of improved A\* algorithm (step 1)

In step 1 all free (without obstacle) adjacent cell of starting cell will be discovered and they will be added in open list with their respective cost which will be calculated by formula  $f(n) = h(n) + h(p) + g(n)$ .

Start cell (2,1) will be removed from open list and will be added in closed list.

After step 1 there will be 4 information in closed list and also will be ordered in increasing order of cost of cells –

cell (3,2), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 3 + 5 = 9$ .

cell (3,1), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 4 + 5 = 10$ .

cell (1,2), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 5 + 5 = 11$ .

cell (1,1), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 6 + 5 = 12$ .

**Step 2:**

In step 2 cell with lowest cost from the open list will be taken as current cell which is in this case cell (3,2).

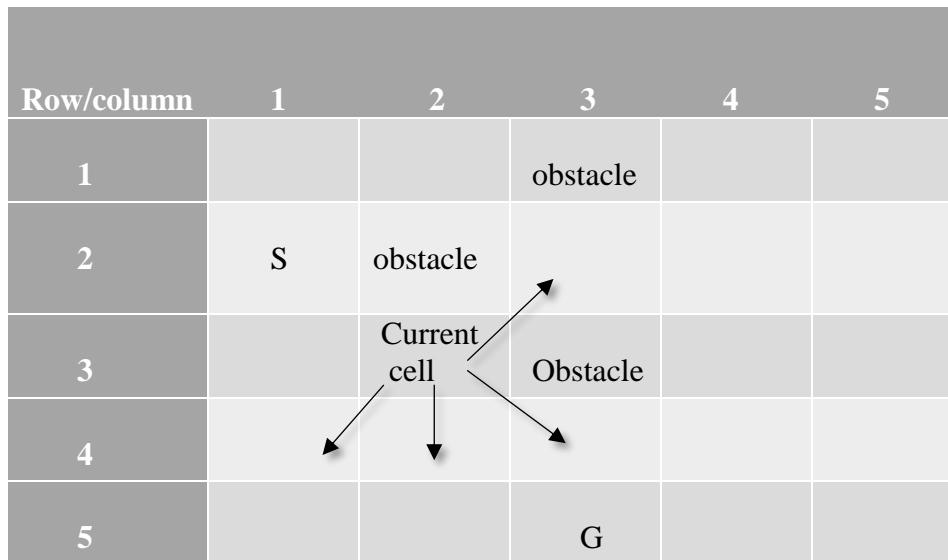


Fig 4.3 Procedure of improved A\* algorithm (step 2)

All free (without obstacle) adjacent cell of current cell will be discovered and they will be added in open list with their respective cost which will be calculated by formula  $f(n) = h(n) + h(p) + g(n)$ .

Current cell (2,1) will be removed from open list and will be added in closed list.

After step 2 there will be 7 information in closed list and also will be ordered in increasing order of cost of cells –

cell (4,3), parent = (3,2) and cost =  $g(n) + h(n) + h(p) = 2 + 1 + 3 = 6$ .

cell (4,2), parent = (3,2) and cost =  $g(n) + h(n) + h(p) = 2 + 2 + 3 = 7$ .



cell (4,1), parent = (3,2) and cost =  $g(n) + h(n) + h(p) = 2 + 3 + 3 = 8$ .  
 cell (2,3), parent = (3,2) and cost =  $g(n) + h(n) + h(p) = 2 + 3 + 3 = 8$ .  
 cell (3,2), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 3 + 5 = 9$ .  
 cell (3,1), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 4 + 5 = 10$ .  
 cell (1,2), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 5 + 5 = 11$ .  
 cell (1,1), parent = (2,1) and cost =  $g(n) + h(n) + h(p) = 1 + 6 + 5 = 12$ .

**Step 3:**

In step 3 cell with lowest cost from the open list will be taken as current cell which is in this case cell (4,3).

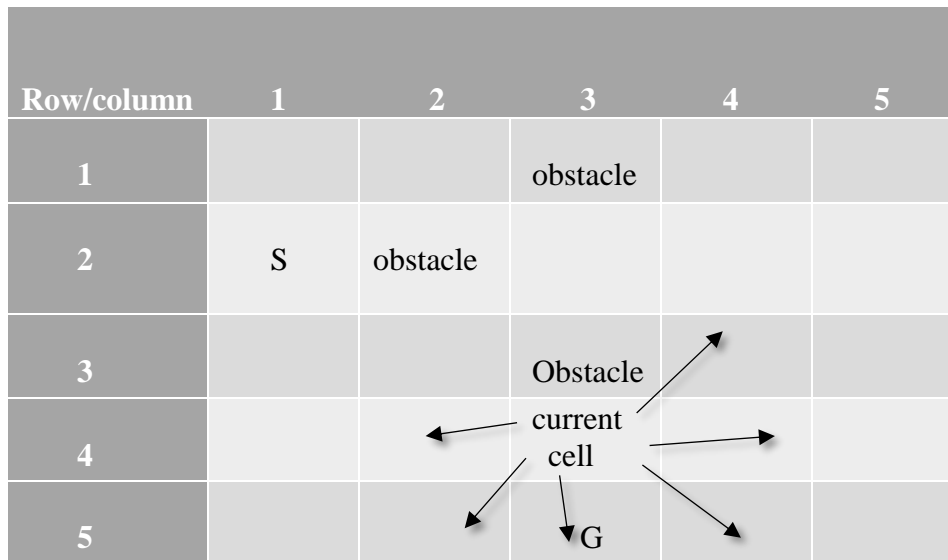


Fig 4.4 Procedure of improved A\* algorithm (step 3)

All free (without obstacle) adjacent cell of current cell will be discovered and they will be added in open list with their respective cost which will be calculated by formula  $f(n) = h(n) + h(p) + g(n)$ .

But in the mean time goal cell (5,3) which is adjacent cell of current cell (4,3) will be found and the path from starting cell to goal cell will be stored.

Path from starting cell to goal cell for this case is –

(2,1) -> (3,2) -> (4,3) -> (5,3)

Following grid will demonstrate the path-

Row/column	1	2	3	4	5
1			obstacle		
2	S	obstacle			
3			Obstacle		
4					
5			G		

Fig 4.5 Procedure of improved A\* algorithm (planned path)

But when heuristic value of parent cell is added with the cost of a cell there still remain some chance to improve the performance of the algorithm. A\* algorithm is sort of breadth first search. So this algorithm expand cell by level. And also consider the cost. Sometime this approach leads to expand unnecessary cells too.

#### 4.4 Modification of improved A\* algorithm:

Even if heuristic value helps to find shortest path more purposefully. But as grid contains obstacle some optimal cell can have higher  $g(n)$  and low heuristic value but still can have low priority than the cell which have low  $g(n)$ . As a result that non optimal cell can be expanded unnecessarily. And throughout the whole process number this kind of cell can be a lot.

To reduce this problem we can increase the priority of cells which have more possibility to reach to the goal. Of course heuristic value also suppose to do the same task. But sometime higher value of  $g(n)$  can decrease the priority of the optimal cells.

To increase the priority of the optimal cell when we expand the adjacent cell of a certain cell we will check which of those cell have more possibility to reach the goal. And we will add a certain value (let's call it V) with all adjacent cell of current cell without the optimal one. As a result all cell without the optimal one will have lower priority. And value V will be added with all adjacent cells without that optimal cell all cell. So if this cell can't reach the goal all other cell will be operate as normal A\* algorithm.

Value V which will be added with the cells to change the order of priority of cells is not a constant value. Rather this value will change according to the cell's position. A cell which has low heuristic value or in other word is close to the goal, it's value V for it's adjacent cell will be lower than the cell which have higher heuristic value. Because cell which are closed to the goal may reach to the goal sooner. So adjacent cell will get higher priority than the cell which have more distance. This value can calculated by several method.

For this paper we calculate value of V for different cells by following method-

We take the row and column of current cell and row and column of starting cell. Then we calculate the ratio of heuristic value of current cell and starting value of current cell. And we multiply the ration with the product of number of row and number of column.

$$V = [ h(\text{current cell}) / h(\text{starting cell}) ] * (\text{number of row} * \text{number of column}).$$

So the value of V of staring cell will be (number of row \* number of column). And V will decrease when heuristic value of current cell decrease. This V will be added with all adjacent cell of current cell without the optimal one.

This method will expand less cell than previous improved algorithm because it will always try check the optimal cells before the other cells. And if it finds an obstacle then it will try to expand the other cells with previous priority.

Following pictures demonstrate the expanded cell of previous improved algorithm and expanded cells of new modified algorithm for a specific environment –





#### 4.5 Procedure of newly modified improved A\* algorithm:

*function calculate*

*find the ration of heuristic value of current cell and heuristic value of starting cell  
multiply the ration with product of number of row and number of column*

*end of function calculateV*

*function Astar*

*initialize open list and closed list  
add starting point into open list  
while open list not empty  
    choose a node from open list  
    delete the node  
    if it is the terminal point then  
        get the path*

*find the optimal child P for current cell  
calculate the value V for current cell by function calculateV*

*for each child point Y  
    if Y is not in either list then  
        find the value of Y  
        if Y is not P then add value V with value of Y  
        add Y into open list  
    else if Y is in open list then  
        compare the value of Y  
        update the list  
    else if less than value of closed list then  
        update the value of closed list  
        remove Y to open list  
    end if  
    end if  
    end if*

*end for  
add x to closed list  
rank the nodes in open list  
end while*

*end of function Astar*

The core idea of this newly modified algorithm is to always try to select the cell from list which has more possibility to reach goal. So the algorithm will always increase the priority of this kind of cell and will give other cells their regular priority. So if the optimal cell fails to reach goal rest of the cell will be expanded in regular basis. And then algorithm will try to find the optimal cells from rest of the cells.

**CHAPTER 5**  
**RESULTS AND DISCUSSIONS**

**Test 1: (2)**

Number of row – 20.

Number of row – 20.

Size of grid – 400.

Start cell – (1,7).

Goal cell – (19,19).

Obstacle density: 3%.

**Performance comparison for Test 1:**

<b>Comparison based on</b>	<b>Previous improved algorithm</b>	<b>Newly modified algorithm</b>	<b>A* algorithm</b>
Time	0.0000122	0.00000131	0.0000139
Path Length	18	18	18
Number of traversed grid	222	131	388

**Test 2: (3)**

Number of row – 40.

Number of row – 45.

Size of grid – 1800.

Start cell – (2,3).

Goal cell – (36,30).

Obstacle density: 9%.



Performance comparison for Test 2:

<b>Comparison based on</b>	<b>Previous improved algorithm</b>	<b>Newly modified algorithm</b>	<b>A* algorithm</b>
Time	0.0000165	0.0000141	0.0000247
Path Length	35	35	34
Number of traversed grid	649	406	1466

**Test 3: (1)**

Number of row – 20.

Number of row – 20.

Size of grid – 400.

Start cell – (0,0).

Goal cell – (19,19).

Obstacle density: 22%.

Performance comparison for Test 3:

<b>Comparison based on</b>	<b>Previous improved algorithm</b>	<b>Newly modified algorithm</b>	<b>A* algorithm</b>
Time	0.0000115	0.0000113	0.0000126
Path Length	26	26	26
Number of traversed grid	146	130	260

**Test 4: (4)**

Number of row – 10.

Number of row – 10.

Size of grid – 100.

Start cell – (0,0).

Goal cell – (1,9).

Obstacle density: 10%.

Performance comparison for Test 4:

<b>Comparison based on</b>	<b>Previous improved A*</b>	<b>Newly modified A*</b>	<b>A* algorithm</b>
Time	0.0000107	0.0000106	0.0000108
Path Length	16	17	16
Number of traversed cell	67	58	79

**Test 5: (6)**

Number of row – 45.

Number of row – 50.

Size of grid – 2250.

Start cell – (2,47).

Goal cell – (21,0).

Obstacle density: 28%.

Performance comparison for Test 5:

<b>Comparison based on</b>	<b>Previous improved algorithm</b>	<b>Newly modified algorithm</b>	<b>A* algorithm</b>
Time	0.0000189	0.0000175	0.000259
Path Length	58	58	57
Number of traversed grid	892	745	1593

**Test 6: (7)**

Number of row – 50.

Number of row – 55.

Size of grid – 2750.

Start cell – (3,23).

Goal cell – (27,9).

Obstacle density: 56%.

Performance comparison for Test 6:

<b>Comparison based on</b>	<b>Previous improved algorithm</b>	<b>Newly modified algorithm</b>	<b>A* algorithm</b>
Time	0.0000107	0.0000106	0.0000109
Path Length	29	29	29
Number of traversed grid	74	64	88

**Test 7: (5)**

Number of row – 50.

Number of row – 55.

Size of grid – 2750.

Start cell – (0,54).

Goal cell – (27,0).

Obstacle density: 20%.

Performance comparison for Test 7:

<b>Comparison based on</b>	<b>Previous improved algorithm</b>	<b>Newly modified algorithm</b>	<b>A* algorithm</b>
Time	0.0000206	0.0000175	0.0000301
Path Length	54	54	53
Number of traversed grid	1055	754	2013

Based on these 7 cases demonstrated above we calculate the average time decreased, average path length increased and average number of traversed cell decreased –

**1) Search time**

- ▶ Average search time decreased by 19.49 % than A\* algorithm.
- ▶ Average search time decreased by 6.82 % than previous improved A\* algorithm.

**2) Path length**

- ▶ Average path length increased by 0.4286 % than A\* algorithm.
- ▶ Average path length increased by 0.1428 % than previous improved A\* algorithm.

**3) Number of traversed grid**

- ▶ Number of traversed grid decreased by 21.493 % than A\* algorithm.
- ▶ Number of traversed grid decreased by 8.82 % than improved A\* algorithm.

## CHAPTER 6

# Conclusions and Recommendations

### 5.1 Findings and Contribution

In this paper a previously proposed improved A\* algorithm is improved further. Previously proposed improved algorithm was based on the introduction of parent node and the change of heuristic cost weight was proposed. And the newly proposed algorithm was an improved version of previous improved algorithm. In this newly improved algorithm a cell called optimal cell among eight adjacent cell of a certain cell is selected and its priority is increased. To increase the priority a value is added with the cost of every adjacent cell of that cell. This newly improved algorithm reduces time of path generation and number of traversed grid. But for some environment it may generate bit larger path.

### 5.2 Recommendation for Future Work

Although newly proposed improved algorithm can find path in any type of environment but for some environment it may generate a longer path. But this path length is not must larger than the optimal path length.

## REFERENCES

- [1] Mingxiu Lin, Kai Yuan, Chengzhi Shi , Yutong Wang. Path Planning of Mobile Robot Based on Improved A\* Algorithm,2017.
- [2] František Duchon, Andrej Babineca, Martin Kajana, Peter Beno, Martin Floreka, Tomáš Ficoa, Ladislav Jurišica. Real-time path planning for the robot in known environment,2014.
- [3] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, Robin De Keyser. Heuristic approaches in robot path planning: A survey, 2016.
- [4] M.S.Ganeshmurthy, Dr.G.R.Suresh. Path Planning Algorithm for Autonomous Mobile Robot in Dynamic Environment,2015.
- [5] Akshay Kumar Guruji, Himansh Agarwal, D. K. Parsediya. Time-Efficient A\* Algorithm for Robot Path Planning, 2016.
- [6] Ke Da1, Liu Xiaoyu1, Zhang Bi1. Variable-Step-Length A\* Algorithm for Path Planning of Mobile Robot, 2017.
- [7] DANIEL DRAKE, SCOTT KOZIOL, EUGENE CHABOT. Mobile Robot Path Planning with a Moving Goal, 2017.
- [8] Behrang Mohajer, Kouros Kiani, Ehsan Samiei, etc. A New Online Random Ppapers Optimization Algorithm for Mobile Robot Path Planning in Dynamic Environments [J].Mathematical Problems in Engineering, 2013.
- [9] P. Raja, S. Pugazhenthii. Optimal path planning of mobile robots: A review, 2012.