Comparison of Path Planning between Improved Informed and Uninformed Algorithms for Mobile Robot

Mohamed Amr[1], Azza Ibrahim[4]
Computers and Systems Department
Electronic Research Institute
ERI, Cairo, Egypt

Ahmed Bahgat[2], Hassan Rashad[3]
Power and Machines Department
Cairo University Faculty of Engineering
CUFE, Cairo, Egypt

***Abstract*—This work is concerned with the Path Planning Algorithms (PPA), which hold an important place in Robotics navigation. Navigation has become indispensable to most modern inventions. Mobile robots have to move to a relevant task point in order to achieve the tasks assigned to them. The actions, which are planned in a structure, may restrict the task duration and even in some situations, the mission tends to be accomplished. This paper aims to study and compare six commonly used informed and uninformed algorithms. Three different maps have been created with gradually increasing difficulty levels related to a number of obstacles in the tested maps. The paper provides a detailed comparison between the algorithms under investigation of several parameters such as: Total steps, straight steps, rotation steps, and search time. The promised results were obtained when the proposed algorithms were applied to a case study.**

Keywords*—Mobile robots; informed algorithm; uninformed algorithm; path planning

I. Introduction

Mobile robots [1] are getting more important in daily life as a result of their increasing role in making human's life easier. According to this point of view, it is obvious that Mobile Robots will be an indispensable part of future life [2]. To make Mobile Robots able to perform an assigned task, they have to know their locations [3], how to navigate [4] within their environment [5] and how to comprehend what the world around them looks like in order to choose the optimal path [6] from the start to the end point in order to reach the target. The path selection depends on the selected criteria to evaluate the path (ex: travel time, number of visited nodes, etc.).The optimal route [7] can be obtained by a variety of methods like the graph theory [8], probabilistic or heuristic [9] based optimization methods [10]. At every point in the grid map representing the working area of the robot, the selected algorithm assigns a direction for the robot out of the possible four directions: Right, Left, Up, and Down. In this paper six different algorithms have been studied, categorized as Informed [11] and Uninformed types [12], a differentiation between them in characteristics is applied, how the algorithm itself works, how the steps of work are correctly applied, and finally, there were three different maps that graduate in their difficulty related to a number of obstacles [13] to test those algorithms, testing six different types of path planning algorithms with three different maps was a challenging work that has been done successfully.

The key objectives of the research are the following:

1) Single mobile robot represents as one node mass.

2) Start and End from Fixed node. Easy, Medium, Hard Map depend on the number of obstacles.

3) Modified informed algorithms and adjusted all algorithms to make the Mobile Robot pass in four dimensions; now no longer in eight directions to ensure that modified algorithms move with less rotation and short pass.

4) Compared all algorithms with each other and get results.

This paper is organized as follows: Section II briefly views the general optimal path problems definitions and the methods, which are executed and discussed in this study. Section III presents the Simulation and the results. Section IV presents the conclusion and the future work.

II. Path Planning Problem and Algorithms

A. Path Planning Problem

First, the path planning is traditionally divided into two categories: local [14], and global [15]. The local is used in an unknown environment while the global is a prior knowledge of the work environment. In this classification, path or route planning [16] is defined as a problem in which an agent moves from a start to an endpoint by avoiding the obstacles in order to reach the target with minimum cost. The starting and the endpoints could be the same (loop closure) or even different. It is preferably desired to have the shortest distance [17] between the start/endpoints. However, the definition of the optimal path can be changeable in some situations. For example, if the tasks are successively assigned to the robots, the elapsed time to calculate the optimal path also matters. When the computation time is too long to obtain the optimal path, it will be difficult to provide the task continuity. According to this reason, the most suitable algorithm has to be chosen according to the desired optimal criterion.

Fig. 1 to 8 demonstrates the different types of the environments as following:

1) Creating a grid map environment for all experiments.
2) Fixed dimensions for the grid map shall be (30*40).
3) Making sure to start and end from a fixed node.
4) Categorizing different Maps depending on a number of obstacles to: Easy, Medium, and Hard Maps.
5) Making modifications and adjustments for the informed algorithms by making them work in four dimensions rather than eight directions.
6) Choosing a single mobile robot to represent only one node mass.

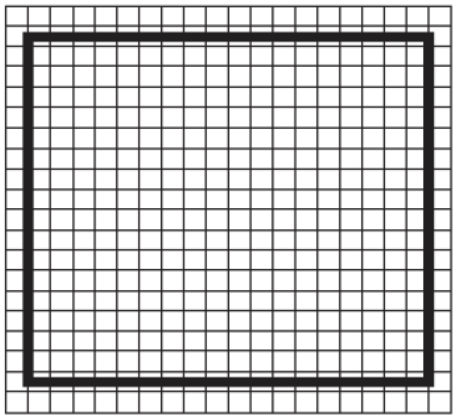


Fig. 1. Grid Map.

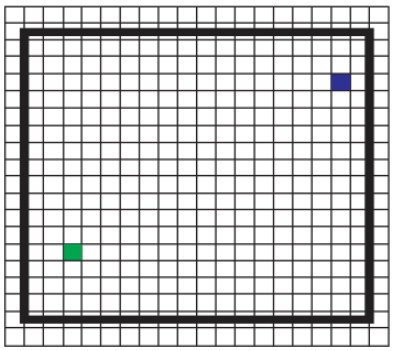


Fig. 2. Fixed Start, End Node.

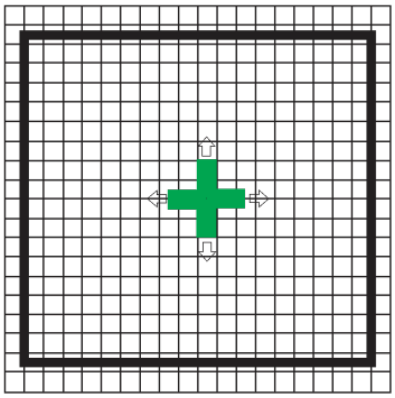


Fig. 3. Four Dimension.

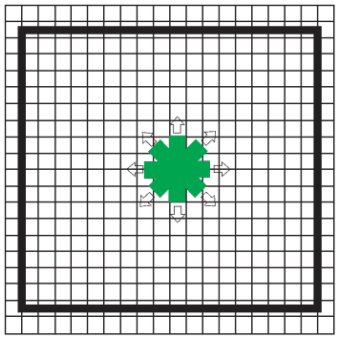


Fig. 4. Eight Direction.

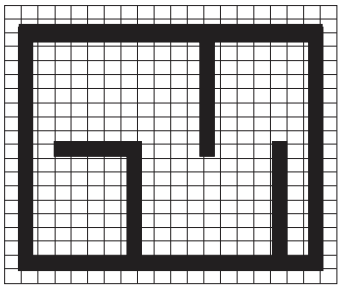


Fig. 5. Easy Map.

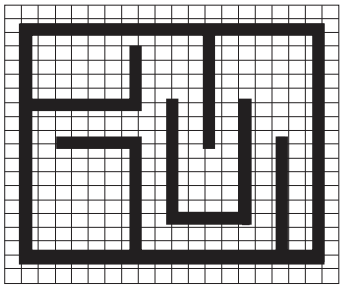


Fig. 6. Medium Map.

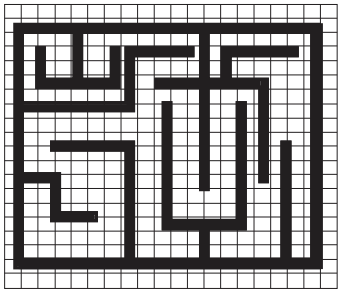


Fig. 7. Hard Map.

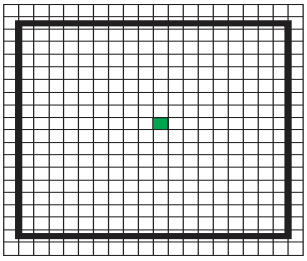


Fig. 8. Single Mobile Robot.

B. Path Planning Algorithm

Optimal path planning algorithm can be classified to informed or uninformed (blind) one. Table I summarizes the algorithms which will be discussed in this paper.

1) Informed algorithms: Informed Algorithms are the algorithms that can be applied if complete information about the working area is available.

TABLE I. INFORMED AND UN-INFORMED ALGORITHMS

Informed	**Un-Informed**
1-Dijkstra [18]	4-Best First Algorithm [19]
2-A star search [20]	5-Breadth First Search [21]
3-Bidirectional A Star [22]	6-Depth First Search [23]

Several algorithms are related to informed algorithms. In this paper, the most common algorithms were selected, which are Dijkstra, A star, and Bidirectional A star.

a) Dijkstra's Algorithm: Dijkstra algorithm is one of the oldest algorithms used in path planning. It was published by Dijkstra in 1959. However, it is still being widely used in several applications [24], [25] [26]. The problem which is handled by Dijkstra Algorithm is represented as follows: when given a graph and a source vertex in that graph, it's obligatory to find the shortest paths from the source to all vertices in the given graph.

The algorithm can be summarized by using the following flowchart:

Algorithm 1: Dijkstra's Algorithm

Initialization:
for every vertex X in Graph:
Dst[X] = infinity // initial distance from source to vertex X is set to infinite
previous[X] = undefined // Previous node in optimal path from source
Dst [source]:= 0 // Distance from source to source
D: = the set of all nodes in Graph // all nodes in the graph are unoptimized - thus are in D

Main Loop
While D is not empty
S = node in D with shortest distance from source
Remove S from D
for each neighbor X of S such that X □D
alt = *Dst*[*S*] + distance between (*S, X*)
if *alt* < *Dst*[*X*] then
Dst[*X*] = *alt*
previous[*X*]:= *S*
End while

The algorithm outputs an array indicating the best previous node (previous) for every node and another array stores the best distance from the source to every node in the graph (dist).

b) A star search (A or A-star or A* search):* A search method [27][28][29] based on a heuristic function, h (n), where n refer to a node n. Every node is associated an estimation h(n) of a route's cost from n to a goal node, while h(n) is equivalent to the actual distance (cost) from n to the goal node.

The other part of this function is g(n), that represents the cost of the path from the beginning node to node n, and f(n), which shows the estimated cost of the path that is being obtained which moves via n to reach to the goal node. f (n) is defined as the total of g (n) with h (n), as in shown Equation:

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

Two lists are mentioned in following flowchart:

Algorithm 2: A Star Algorithm

Initialization:
Let open list have only the starting node
Let closed list empty

Main Loop
While (the destination node has not been reached):
consider the node with the lowest f score in the open list
If (this node is destination node):
we are finished.
Else If: put the current node in the closed list and look at all of its neighbors.
For (each neighbor of the current node):
If (neighbor has lower g value than current and is in the closed list).
replace the neighbor with the new, lower, g value
current node is now the neighbor's parent
Else If (current g value is lower and this neighbor is in the open list).
replace the neighbor with the new, lower, g value
change the neighbor's parent to current node
Else If this neighbor is not in both lists:
add it to the open list and set its g .
End while.

The algorithm outputs an array indicating the best path related to the lowest cost 'g value' for every node and an array stores the best distance from the source to every node in the graph (closed list).

c) Bidirectional A Star Algorithm: The bidirectional search algorithm combines two separate searches. The search that is performed from both the origin and destination simultaneously, and when these two searches meet, then the shortest path can be obtained.

Next are the steps and pseudocode to show how bidirectional A star search works:

- Search forward from the start point.
- Search backward from the goal point.
- Using "fF", "gF" ,"hF" to indicate f, g, and h-costs in the forward search.
- Also using "fB","gB","hB" similarly in the backward search.

Algorithm 3: Bidirectional A Star Algorithm

Initialization:
Open F and Open B
store states generated in the forward and back ward directions,
Finally, g min F, g min B, f min F and f min B

denote the minimal g- and f-values in OpenF and OpenB
d (x, y) denotes the shortest distance between **x** and **y**. **Front-to-end** algorithms use two heuristic functions. **Main Loop**
The **forward heuristic**, **hF** , is forward admissible
If f hF (u) ≤ d (u, goal)
For all u in G and is forward consistent
If f hF (u) ≤ d (u, u') + hF (u')
For all u and u' in G.
The backward heuristic, hB, is backward admissible
If f hB(v) ≤ d (start, v)
For all v in G and is backward consistent
If f hB(v) ≤ d (v', v) + hB (v')
For all v and v' in G.
final f = d (start, goal) is the cost of an optimal solution.
End

After this implementation, some modifications in informed algorithms are needed to improve path length to make them more optimum and straight as possible.

The huge problem in informed algorithms which is related to the cost function search is that they cannot find the least rotation steps in many paths with the same path length. In this paper, informed algorithms are improved by adding the rotation cost estimation method based on a moving direction. Assuming that the node n coordinate is (xn,yn), then the previous node coordinate is (xn-1,yn-1),and the coordinates of the next node is (xn+1,yn+1). Then the rotation evaluation function of node n is k (n).

$$f(n) = g(n) + h(n) + K(n) \quad (2)$$

2) Uninformed algorithms: Blind search is another name for uninformed search formulas. The search algorithm generates the search tree without relying on any domain knowledge, which is a brute force in nature. Uninformed Algorithms lack background information on how to approach the goal or destination that must be reached.

Best First Algorithm "BFS" stands for Best First Search, it's an evaluation function that measures distance to the goal, the general approach of this search algorithm is that the node is selected for expansion based on an evaluation function f(n).

It combines the advantages of both DFS and BFS in a single method; in DFS not all combining branches have to be expanded; on the other hand, the BFS are not trapped on dead-end paths. Combining the two to follow a single path at some time, but switching between paths whenever there is some competing path looks more promising than the current one.

a) Best first search algorithm: At every step of the BFS [30] search process, The most promising nodes are selected that have been generated thus far, applying some appropriate heuristic function to each of them, and then making an expansion to the chosen node by using the specified rules to generate its successors, which is known as an OR-graph, because each of its branches representing an alternative problem-solving path, and storing the nodes in the to Visit Nodes data structure using a queue.

Algorithm4: Best First Search Algorithm

Initialization:
open list containing the start state.
CLOSED list empty.
BEST =start state.
lets s= arg max e(x).
(get the state from OPEN with the highest evaluation)
remover S from OPEN and add to CLOSED
if e(S)>=e(BEST),
then BEST =S

Main Loop
for each child t of s that is not in the OPEN or CLOSESD list, evaluate and add to OPEN
if BEST changed in the last set of expansions
goto step four
Return BEST

b) Breadth first search: There are different ways to traverse graphs, this algorithm means by graph traversal visiting every vertex and edge exactly one time in a well-defined order, as using a certain graph algorithm, each vertex of the graph must be visited one time exactly, and the order in which the vertices are visited are so important as it may depends on the algorithm or the problem that is needed to be solved.

Next are steps and pseudo code explaining BFS methodology.

First step: move horizontally then visit all the nodes of the present layer.

Second step: Move to the following layer

Algorithm 5:Breadth First Search

Initialization
Set all nodes to "not visited";
q = new Queue ();
q. enqueue (initial node);

Main Loop
While (q ≠ empty)
do{ x = q. dequeue ();
If (x has not been visited)
{visited[x] = true; // Visit node x
for (every edge (x, y) // we are using all edges
If (y has not been visited)
q. enqueue(y); // Use the edge (x, y)
End While

c) Depth first search: DFS stands for Depth First Search, it is an algorithm for traversing or searching a tree or graph data structure, it uses a stack data structure for implementation. In DFS, one starts at the root and explores as far as possible along each branch before backtracking. Nest are steps and pseudocode explaining DFS methodology.

- Initializing nodes with status =1(ready data).
- Put starting node in the stack and change status to status=2(waiting state).
- Loop: repeat the next two steps until stack gets empty or algorithm reaches goal node.
- Remove front node N from stack, process them and change the status of N to status=3(processed stat).
- Add all the neighbors of N to the stack and change status to status=2(waiting status).

Algorithm 6:Depth First Search

Initialization:
procedure DFS-iterative (G, v) is
let S be a stack
S.push (v)

Main Loop
While S is not empty do
v = S.pop()
If v is not labeled as discovered then
label v as discovered
For all edges from v to w in G.adjacent Edges (v)
do S.push(w)
End while

III. Simulation and Results

To demonstrate the benefits of the modification for the algorithms in terms of search speed, visited nodes, number of rotations, path selection, and path length, Three separate experimental locations have been created, basic and modified algorithms are compared then comparing those algorithms with uninformed ones, The experiments aim to establish a starting point and a target point environment.

Fig. 9(a) to 11(b) shows the difference between Basic and Modified Dijkstra is in three different maps.

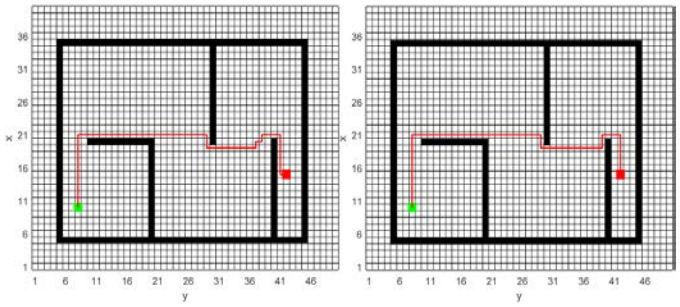
A. Dijkstra Algorithm

In this environment, it shows the search methods and gives a comparison between the two algorithms (Dijkstra algorithm and modified Dijkstra algorithm) in three environments.

Fig. 9(a) to 11(b) shows the difference between Basic and Modified Dijkstra in three different maps.

Also, Tables II to IV compare Basic and Modified Dijkstra in total steps, straight steps, rotation steps, visited nodes, and search time on three different maps.

1) Dijkstra and modified Dijkstra algorithm in Easy Map.



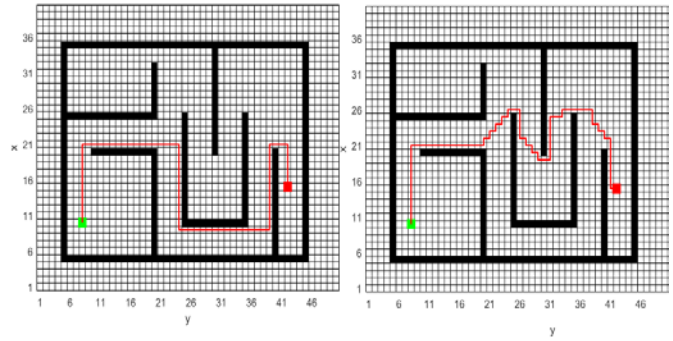
(a) (b)

Fig. 9. (a) Modified Dijkstra for Easy Map (b) Basic for Easy Map.

TABLE II. Comparison between Basic and Modified Dijkstra Algorithm in Easy Map

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic Dijkstra	56	47	9	986	4.9
Modified Dijkstra	56	50	6	1010	5.3

2) Dijkstra algorithm in Medium Map



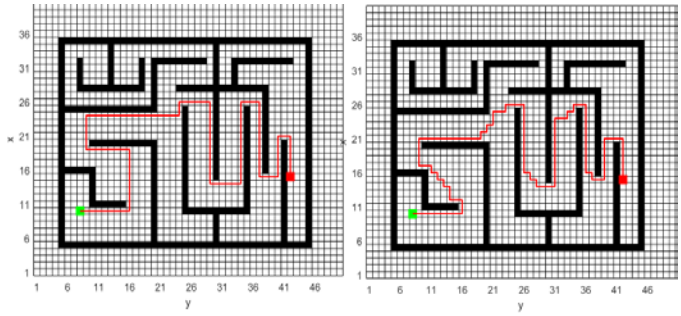
(a) (b)

Fig. 10. (a) Modified Dijkstra for Medium Map (b) Basic Dijkstra for Medium Map.

TABLE III. Comparison between Basic and Modified Dijkstra Algorithm in Medium Map

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic Dijkstra	76	45	31	974	4.29
Modified Dijkstra	76	70	6	976	4.32

3) Dijkstra Algorithm in Hard Map.



(a) (b)

Fig. 11. (a) Modified Dijkstra for Hard Map (b) Basic Dijkstra for Hard Map.

TABLE IV. COMPARISON BETWEEN BASIC AND MODIFIED DIJKSTRA ALGORITHM IN HARD MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic Dijkstra	112	73	39	812	3.95
Modified Dijkstra	112	97	15	814	3.96

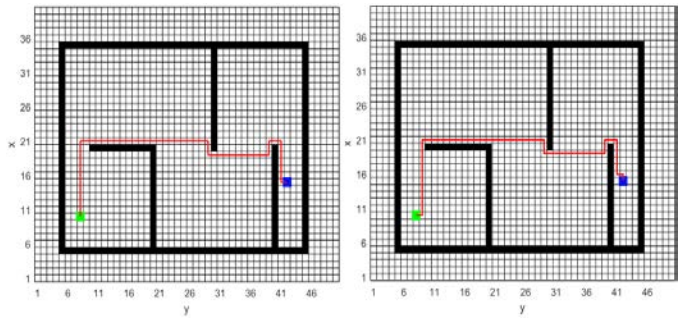
As shown in the Fig. 9(a) to 11(b) and Tables II to IV, both of the Dijkstra algorithm and the improved Dijkstra algorithm can search for symmetric paths of the same length, but the search path of the improved Dijkstra algorithm has more straight length, the path trajectory is different, and the number of rotation points in the paths are also different. It is obvious that the improved Dijkstra algorithm search straighter than the Dijkstra algorithm by equal to 3 steps in easy map, 25 steps in medium map and 19 steps in hard map.

B. A Star Algorithm

In this environment, Fig. 12(a) to 14(b) shows the difference between basic and modified A Star in three different maps.

In addition, Tables V to VII gives a comparison between the two algorithms in total steps, straight steps, rotation steps, visited nodes, and search time on three different maps.

1) A Star Algorithm in Easy Map



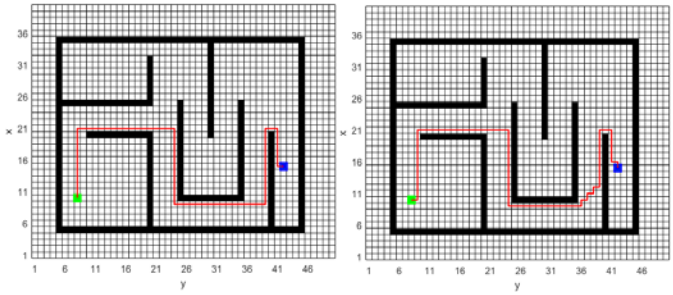
(a) (b)

Fig. 12. (a) Modified a Star for Easy Map, (b) Basic a Star for Easy Map

TABLE V. COMPARISON BETWEEN BASIC AND MODIFIED A STAR ALGORITHM IN EASY MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic A Star	56	47	9	569	2.96
Modified A Star	56	49	7	726	3.15

2) A Star Algorithm in Medium Map.



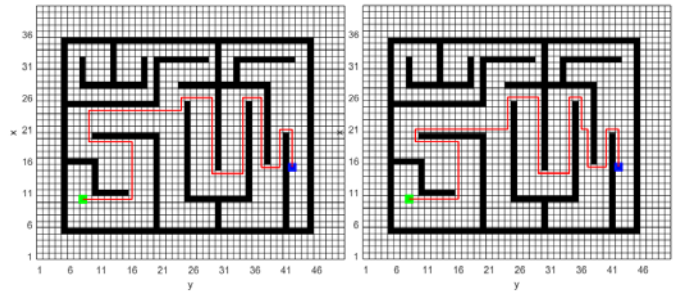
(a) (b)

Fig. 13. (a) Modified a Star for Medium Map, (b) Basic a Star for Medium Map.

TABLE VI. COMPARISON BETWEEN BASIC AND MODIFIED A STAR ALGORITHM IN MEDIUM MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic A Star	76	61	15	758	3.19
Modified A Star	76	69	7	768	3.21

3) A Star Algorithm in Hard Map



(a) (b)

Fig. 14. (a) Modified a Star for Hard Map, (b) Basic a Star for Hard Map.

TABLE VII. COMPARISON BETWEEN BASIC AND MODIFIED A STAR ALGORITHM IN HARD MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic A Star	112	95	17	679	2.96
Modified A Star	112	97	15	698	2.97

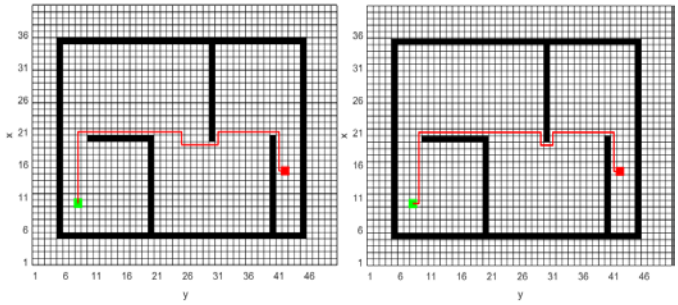
As shown in above Fig. 12(a) to 14(b) and Tables V to VII, both A Star algorithm and the improved A Star algorithm can search for symmetric paths of the same length, but the search path of the improved A Star Algorithm has more straight length, the path trajectory is different, and the number of rotation points in the paths are also different. And it is obvious that the improved A Star Algorithm search straighter than A Star Algorithm by equal to 3 steps in easy map, 8 steps in medium map and 2 steps in hard map.

C. Bidirectional a Star

In this environment, Fig. 15(a) to 17(b) shows the difference between Basic and modified Bidirectional A Star in three different maps.

In addition, Tables VIII to X give a comparison between the two algorithms in total steps, straight steps, rotation steps, visited nodes, and search time on three different maps.

1) Bidirectional A Star algorithm in Easy Map:



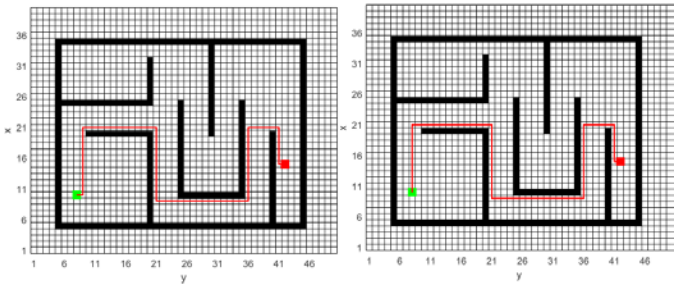
(a) (b)

Fig. 15. (a) Modified Bidirectional a Star for Easy Map (b) Basic Bidirectional a Star for Easy Map.

TABLE VIII. COMPARISON BETWEEN BASIC AND MODIFIED BIDIRECTIONAL A STAR ALGORITHM IN EASY MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic Bidirectional A Star	56	48	8	437	1.32
Modified Bidirectional A Star	56	49	7	462	1.57

2) Bidirectional A Star Algorithm in Medium Map



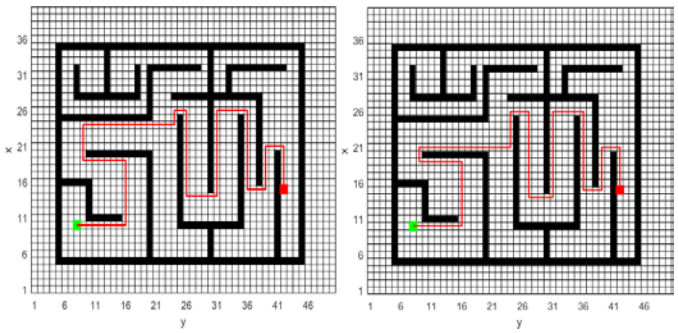
(a) (b)

Fig. 16. (a) Modified Bidirectional a Star for Medium Map, (b) Basic Bidirectional a Star for Medium Map.

TABLE IX. COMPARISON BETWEEN BASIC AND MODIFIED BIDIRECTIONAL A STAR ALGORITHM IN MEDIUM MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic Bidirectional A Star	76	68	8	651	2.71
Modified Bidirectional A Star	76	69	7	659	2.73

3) Bidirectional A Star Algorithm in Hard Map.



(a) (b)

Fig. 17. (a) Modified Bidirectional a Star for Hard Map (b) Basic Bidirectional a Star for Hard Map.

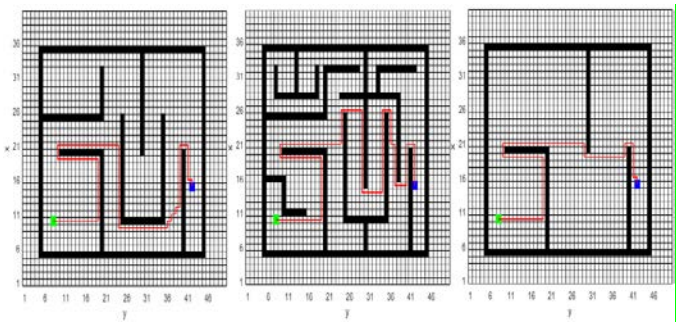
TABLE X. COMPARISON BETWEEN BASIC AND MODIFIED BIDIRECTIONAL A STAR ALGORITHM IN HARD MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
Basic Bidirectional A Star	112	97	15	747	3.16
Modified Bidirectional A Star	112	97	15	756	3.19

As shown in Fig. 15(a) to 17(b) and Tables VIII to X, it can be seen that both the Bidirectional A Star algorithm and the improved Bidirectional A Star algorithm can search for symmetric paths of the same length, but the search path of the improved Bidirectional A Star algorithm has more straight length, the path trajectory is different, and the number of rotation points in the paths are also different. In addition, it is obvious that the improved Bidirectional A Star algorithm search straighter than Bidirectional A Star algorithm can reach to one-step in easy map one-steps in medium map and the same steps in hard map.

D. Best First Search Algorithm

1) Best first search algorithm in easy map: The three different maps of BFS are demonstrated in three figures [Fig. 18(a) to 18(c)].



(a) (b) (c)

Fig. 18. (a) BFS Easy Map, (b) BFS Medium Map, (c) BFS Hard Map.

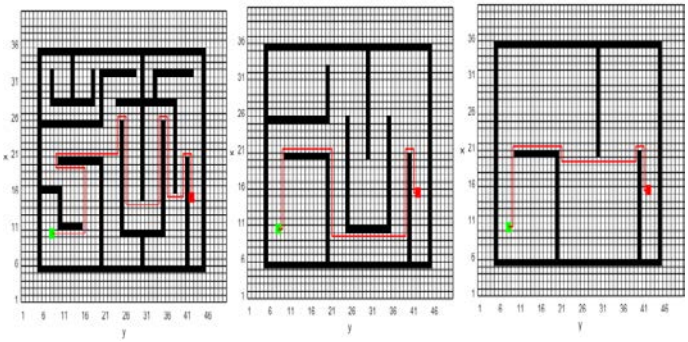
Tables XI gives the result of BFS in total steps, straight steps, rotation steps, visited nodes, and search time on three different maps.

TABLE XI. RESULTS OF BFS IN EASY, MEDIUM AND HARD MAP

Algorithm	Path Length	Straight Steps	Rotation Steps	Visited nodes	Search time
BFS Easy Map	76	65	11	462	1.54
BFS Medium Map	96	78	17	659	2.75
BFS Hard Map	118	101	17	756	3.17

E. Breadth First Search

1) Breadth first algorithm in easy map: The three different maps of BFS is demonstrated in three Fig. 19(a) to 19(c).



(a) (b) (c)

Fig. 19. (a) BFS Easy Map, (b) BFS Medium Map, (c) BFS Hard Map.

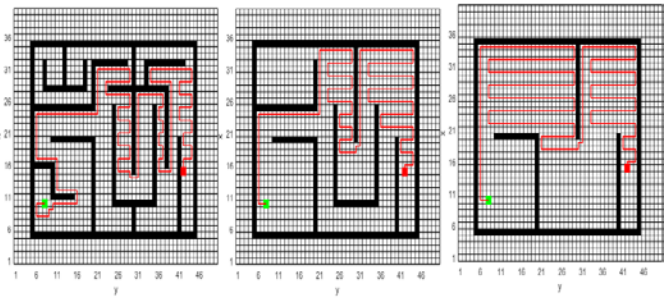
The Table XII gives the result of BRFS in total steps, straight steps, rotation steps, visited nodes, and search time on three different maps.

TABLE XII. RESULTS OF BRFS IN EASY, MEDIUM AND HARD MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
BRFS Easy Map	56	48	8	678	2.95
BRFS Medium Map	76	68	8	673	2.97
BRFS Hard Map	112	97	15	805	3.81

F. Depth First Search

1) Depth First algorithm in Easy Map: The three different maps of DFS is demonstrated in three figures below [Fig. 20(a)-(c)].



(a) (b) (c)

Fig. 20. (a) DFS Easy Map, (b) DFS Medium Map, (c) DFS Hard Map.

The Table XIII gives the result of DFS in total steps, straight steps, rotation steps, visited nodes, and search time on three different maps.

TABLE XIII. RESULTS OF DFS IN EASY, MEDIUM AND HARD MAP

Algorithm	Total Steps	Straight Steps	Rotation Steps	Visited nodes	Search time
DFS Easy Map	332	289	41	625	2.68
DFS Medium Map	220	177	43	526	2.61
DFS Hard Map	224	159	65	526	2.35

2) Comparison between modified informed algorithm and informed algorithms: In all informed and modified informed algorithms path length are equal in all maps, but trajectory path and straight length in selected path is improved as shown in Table XIV.

TABLE XIV. COMPARISON OF STRAIGHT LENGTH IN UNINFORMED ALGORITHMS

Algorithm	Easy Map straight Length	Medium Map Path Length	Hard Map Path Length
BFS	65 /76	78/96	101/118
BRFS	48/56	68/76	97/112
DFS	289/332	177/220	159/224

G. These Experiments Show us Best Uninformed Algorithm in all Maps is BRFS

1) Comparison between modified informed algorithm and informed algorithms: In Table XV all informed and modified informed algorithms path length are equal in all maps, but trajectory path and straight length in selected path is improved, in easy map path equal 56 steps, in medium map 76 steps, in hard map 112 steps.

TABLE XV. COMPARISON OF STRAIGHT LENGTH IN MODIFIED INFORMED AND INFORMED ALGORITHMS

Algorithm	Easy Map straight Length	Medium Map straight Length	Hard Map straight Length
Dijkstra	47 /56	45/76	73/112
Mod Dijkstra	50/56	70/76	97/112
A star	47/56	61/76	95/112
Mod A star	49/56	69/76	97/112
Bidirectional A star	48/56	68/76	97/112
Mod Bidirectional A star	49/56	69/76	97/112

Table XV shows us that:

- Dijkstra algorithm is improved by more than 5% in Easy Map, more than 32% in medium Map, more than 19% in Hard Map.
- A Star Algorithm is improved by more than 3% in Easy Map, more than 10% in medium Map, more than 1% in Hard Map.
- Bidirectional A Star Algorithm is improved by more than 1% in Easy Map, more than 2% in medium Map.
- Modified Dijkstra has higher effect in results more than other two algorithms.

2) Comparison between modified informed algorithm and uninformed algorithms: According to informed and uninformed algorithms experiment data as shown in Table XVI, the obtained path trajectories are not the same for all algorithms, which indicates that the Modified Dijkstra algorithm has Minimum straight length path in all maps, in other side modified A Star, modified Bidirectional A Star have same straight length easy and medium map, also all of three modified informed algorithms have Minimum straight length path Comparing the search time easy, medium and hard map informed and uninformed search.

TABLE XVI. COMPARISON BETWEEN MODIFIED INFORMED AND UNINFORMED ALGORITHMS

Algorithm	Visited nodes			Search time in second		
	Easy	Medium	Hard	Easy	Medium	Hard
Mod Dijkstra	1010	976	814	5.3	4.32	3.96
Mod A star	726	768	698	3.15	3.21	2.97
Mod Bidirectional A star	462	659	756	1.57	2.73	3.19
BFS	462	659	756	1.54	2.75	3.17
BRFS	678	673	805	2.95	2.97	3.81
DFS	625	526	526	2.68	2.61	2.35

IV. DISCUSSION

To demonstrate the Benefits of the modified algorithms in terms of search speed, number of rotation, path selection and path length, three separate experimental locations have been created. The experiments establish a starting point and target point environment. Dijkstra algorithm improved by more than 5% in Easy Map, more than 32% in medium Map, more than 19% in Hard Map.

A Star Algorithm improved by more than 3% in Easy Map, more than 10% in medium Map, more than 1% in Hard Map.

Bidirectional A Star Algorithm is improved by more than 1% in Easy Map, more than 2% in medium Map.

Modified Dijkstra has better results over the other two algorithms.

V. CONCLUSION

The purpose of this paper is to modify and adjust informed search algorithms and compare pure informed algorithms and uninformed algorithms to find the shortest path between the start and the goal on designed maps of various obstacles' quality. Additionally, to enhance the identification of the shortest path for each algorithmic search with less rotation within the path and to simulate its performance in the designed maps, it can be noticed from the results that algorithms that have high visited nodes have more optimums than algorithms with low visited nodes. Search time is directly proportional to the number of visited nodes. Finally, heuristic search algorithms are more flexible than blind search algorithms.

Future work will involve the proposed algorithms that are compatible with Artificial Intelligence to make high-response and more features to keep up with the modern technology requirements.

REFERENCES

[1] Mester, “Applications of mobile robots,” 2006.

[2] P. E. Teleweck and B. Chandrasekaran, “Path planning algorithms and their use in robotic navigation systems,” in Journal of Physics: Conference Series, 2019, vol. 1207, no. 1, p. 12018.

[3] C. M. Pop, G. L. Mogan, and M. Neagu, “Localization and Path Planning for an Autonomous Mobile Robot Equipped with Sonar Sensor,” in Applied Mechanics and Materials, 2015, vol. 772, pp. 494–499.

[4] T. T. Hoang, D. T. Hiep, P. M. Duong, N. T. T. Van, B. G. Duong, and T. Q. Vinh, “Proposal of algorithms for navigation and obstacles avoidance of autonomous mobile robot,” in 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), 2013, pp. 1308–1313.

[5] Stentz, “Optimal and efficient path planning for partially known environments,” in Intelligent unmanned ground vehicles, Springer, 1997, pp. 203–220.

[6] I-H. Zhou and H.-Y. Lin, “A self-localization and path planning technique for mobile robot navigation,” in 2011 9th World Congress on Intelligent Control and Automation, 2011, pp. 694–699.

[7] G. E. Jan, K.-Y. Chang, and I. Parberry, “A new maze routing approach for path planning of a mobile robot,” in Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), 2003, vol. 1, pp. 552–557.

[8] R. Katsuki, T. Tasaki, and T. Watanabe, “Graph Search Based Local Path Planning with Adaptive Node Sampling,” in 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 2084–2089.

[9] C. R. Reeves, “Heuristic search methods: A review,” Oper. Res. Pap., pp. 122–149, 1996.

[10] C. A. Floudas and P. M. Pardalos, Encyclopedia of optimization. Springer Science & Business Media, 2008.

[11] C. Grosan and A. Abraham, "Informed (Heuristic) Search," in Intelligent Systems, Springer, 2011, pp. 53–81.

[12] S. Pooja, S. Chethan, and C. V Arjun, "Analyzing uninformed search strategy algorithms in state space search," in 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 2016, pp. 97–102.

[13] Hassani, I. Maalej, and C. Rekik, "Robot path planning with avoiding obstacles in known environment using free segments and turning points algorithm," Math. Probl. Eng., vol. 2018, 2018.

[14] Y. Wang, X. Yu, and X. Liang, "Design and implementation of global path planning system for unmanned surface vehicle among multiple task points," Int. J. Veh. Auton. Syst., vol. 14, no. 1, pp. 82–105, 2018.

[15] N. Buniyamin, W. W. Ngah, N. Sariff, and Z. Mohamad, "A simple local path planning algorithm for autonomous mobile robots," Int. J. Syst. Appl. Eng. Dev., vol. 5, no. 2, pp. 151–159, 2011.

[16] Selamat, M. Zolfpour-Arokhlo, S. Z. Hashim, and M. H. Selamat, "A fast path planning algorithm for route guidance system," in 2011 IEEE International Conference on Systems, Man, and Cybernetics, 2011, pp. 2773–2778.

[17] Jiang, L. D. Seneviratne, and S. W. E. Earles, "A shortest path based path planning algorithm for nonholonomic mobile robots," J. Intell. Robot. Syst., vol. 24, no. 4, pp. 347–366, 1999.

[18] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," J. ACM, vol. 20, no. 3, pp. 385–388, 1973.

[19] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A," J. ACM, vol. 32, no. 3, pp. 505–536, 1985.

[20] X. Liu and D. Gong, "A comparative study of A-star algorithms for search and rescue in perfect maze," in 2011 International Conference on Electric Information and Control Engineering, 2011, pp. 24–27.

[21] F. Zhang et al., "An adaptive breadth-first search algorithm on integrated architectures," J. Supercomput., vol. 74, no. 11, pp. 6135–6155, 2018.

[22] P. Muntean, "Mobile robot navigation on partially known maps using a fast a star algorithm version," arXiv Prepr. arXiv1604.08708, 2016.

[23] Kaur, P. Sharma, and A. Verma, "A appraisal paper on Breadth-first search, Depth-first search and Red black tree," Int. J. Sci. Res. Publ., vol. 4, no. 3, pp. 2–4, 2014.

[24] H. Wang, Y. Yu, and Q. Yuan, "Application of Dijkstra algorithm in robot path-planning," in 2011 second international conference on mechanic automation and control engineering, 2011, pp. 1067–1069.

[25] G. Qing, Z. Zheng, and X. Yue, "Path-planning of automated guided vehicle based on improved Dijkstra algorithm," in 2017 29th Chinese control and decision conference (CCDC), 2017, pp. 7138–7143.

[26] P. Sari, M. F. Fahroza, M. I. Mufit, and I. F. Qathrunad, "Implementation of Dijkstra's Algorithm to Determine the Shortest Route in a City," J. Comput. Sci. Inf. Technol. Telecommun. Eng., vol. 2, no. 1, pp. 134–138, 2021.

[27] B. Wang, "Path Planning of Mobile Robot Based on A* Algorithm," in 2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI), 2021, pp. 524–528.

[28] F. Duchoň et al., "Path planning with modified a star algorithm for a mobile robot," Procedia Eng., vol. 96, pp. 59–69, 2014.

[29] T. XiangRong, Z. Yukun, and J. XinXin, "Improved A-star algorithm for robot path planning in static environment," in Journal of Physics: Conference Series, 2021, vol. 1792, no. 1, p. 12067.

[30] H. K. Tripathy, S. Mishra, H. K. Thakkar, and D. Rai, "CARE: A Collision-Aware Mobile Robot Navigation in Grid Environment using Improved Breadth First Search," Comput. Electr. Eng., vol. 94, p. 107327,2021.