Most Probable Path

using **A* Heuristic Search**

Dr. Nilotpal Chakraborty

Department of Computer Science & Engineering
The LNM Institute of Information Technology, Jaipur

November 2020

Most Probable Path

using **A* Heuristic Search**

Most Probable Path is a program that will show the optimal path visually on the map image(i.e LNMIIT Map) from the given source coordinates to destination coordinates having path in map (2-D Grid) and will also show the result comparison for the Dijkstra's Shortest Path Algorithm , A* Heuristic Search using Manhattan Distance , Euclidean Distance.

It will show the benefits of A* Heuristic Search in place traditional Approach to find shortest path i.e. Dijkstra's Shortest Path Algorithm.

We will further discuss the implementation of A* heuristic algorithm, comparing results for different Heuristics values like Manhattan distance or Euclidean Distance , what improvement is done using this algorithm , what problems we faced using this approach.

Contents

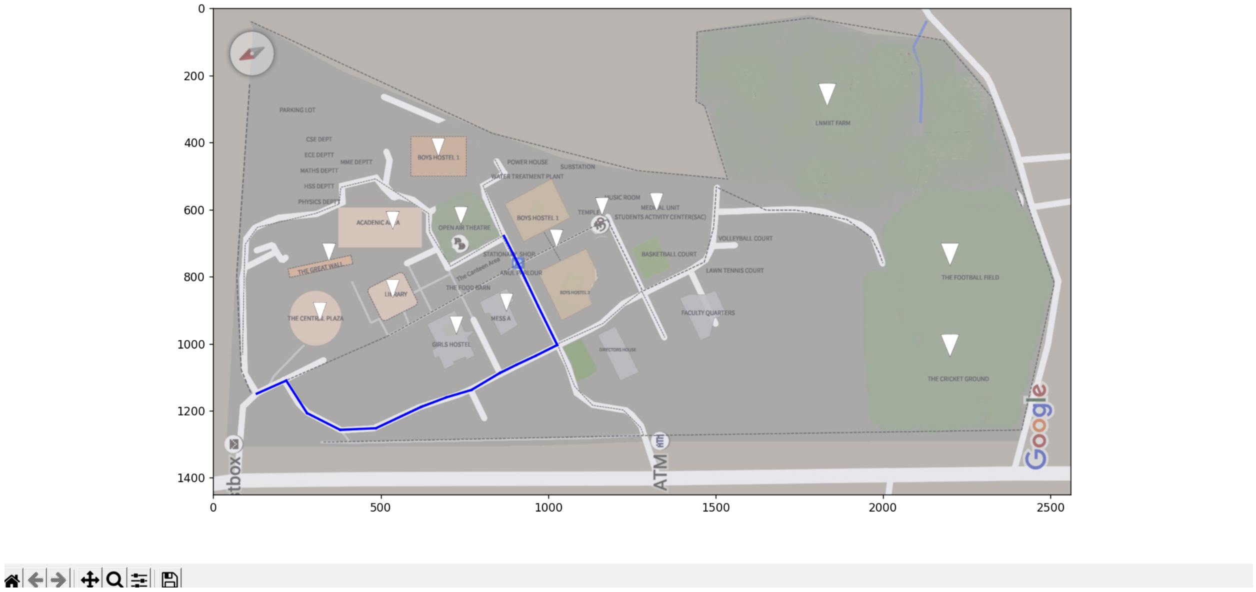
INTRODUCTION

Most Probable Path it is an program which will show us the optimal path in map of LNMIIT for given starting position to final destination using A* Heuristic Search.

This Project also includes the results of comparison of traditional approaches to solve this problem i.e. Dijkstra's Shortest Path Algorithm and A* heuristic algorithm.

Idea for the project was from Google Maps using 2-D grid for the graph and applying A* heuristic algorithm , Our aim was to use the topic from our course curriculum and apply it for real life applications.

Sample Result

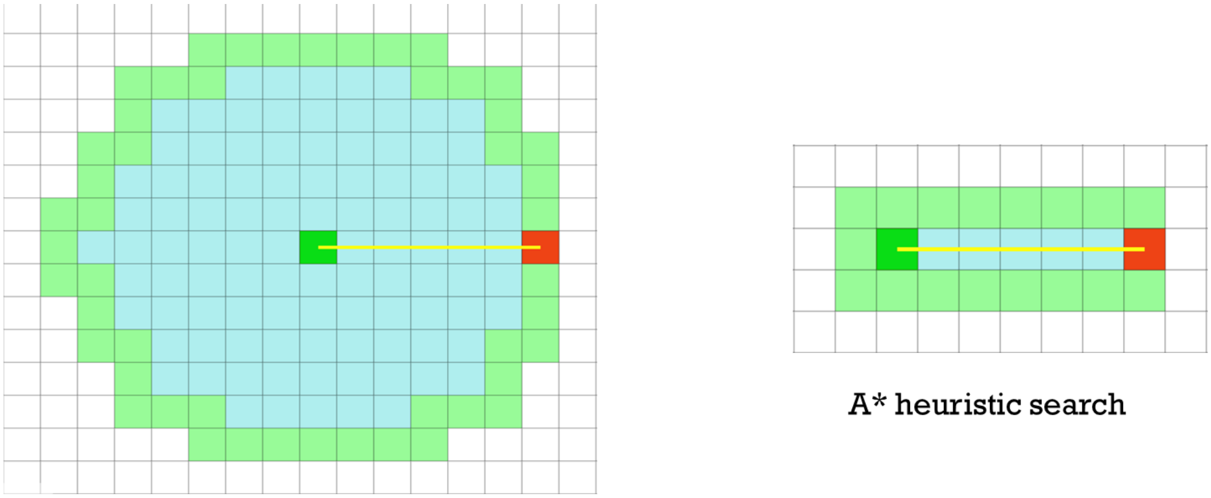


*This is the result for the given source and destination coordinates.

CONTRIBUTION AND NOVELTY

Project is based on finding a path from source to destination which can be done using Dijkstra’s Algorithm which is an uninformed algorithm which means it doesn’t need to know about the target node beforehand. But we can use these information to get to the better complexity which takes us to our approach using A* heuristic search.

Our approach for Project was to consider the graph to be big like google maps (2-D Grid) but Dijkstra’s algorithm moves all nodes till destination means to cover a large area of the graph which can be costly in terms of memory and time on contrary A* Heuristic search uses the pre-known knowledge to reach the destination and we have used Manhattan or Euclidean distance as heuristic values for the better result.



A* heuristic search

Dijkstra's shortest path **algorithm**

Number of visited nodes in both searches

PROPOSED SYSTEM AND ALGORITHM

Project is based on finding an optimal path from the given coordinates to destination in a 2-D Grid having defined edges of the graph and outputting the path to cover in the graph visually.

So to solve this problem we need to visit connected nodes to initial node and move further to all connected nodes connected to the original node and so on like the traditional approach we use in Dijkstra's Shortest Path Algorithm but it is an uninformed algorithm and we have to design it for a big graph like Google Maps (2-D Grid) and this approach would cover a large area of the graph to reach to final result or destination which can be costly in terms of memory and time.

To tackle the problem from the above approach we can use the information about the graph we have , we can use A* Heuristic Search with the heuristic values best for our problem.

A* Heuristic Search

The A* algorithm is based on heuristics for navigating the search, but unlike many similar algorithms with this base (for example Best Search Algorithm), it is both complete and (under certain conditions – if graph is pre-known) optimal:

- A complete algorithm is an algorithm that guarantees a correct answer for any correct input, if that answer exists
- An optimal algorithm is an algorithm that returns an answer for the shortest possible amount of time.

For the A* algorithm, the evaluation function has a specific form:

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

$g(n)$ is distance of current node from source node

$h(n)$ is approximate distance of goal node from current node

The optimality of A* algorithm is heavily dependent on the quality of its function of evaluation.

For our approach we found the best result using heuristic value to be Manhattan Distance/Euclidean Distance.

Manhattan Distance = $| x1 - x2 | + |y1 - y2|$

where $(x1,y1)$,$(x2,y2)$ are coordinates in a 2-D grid.

Euclidean Distance = $(x1 - x2)^2 + (y1 - y2)^2$

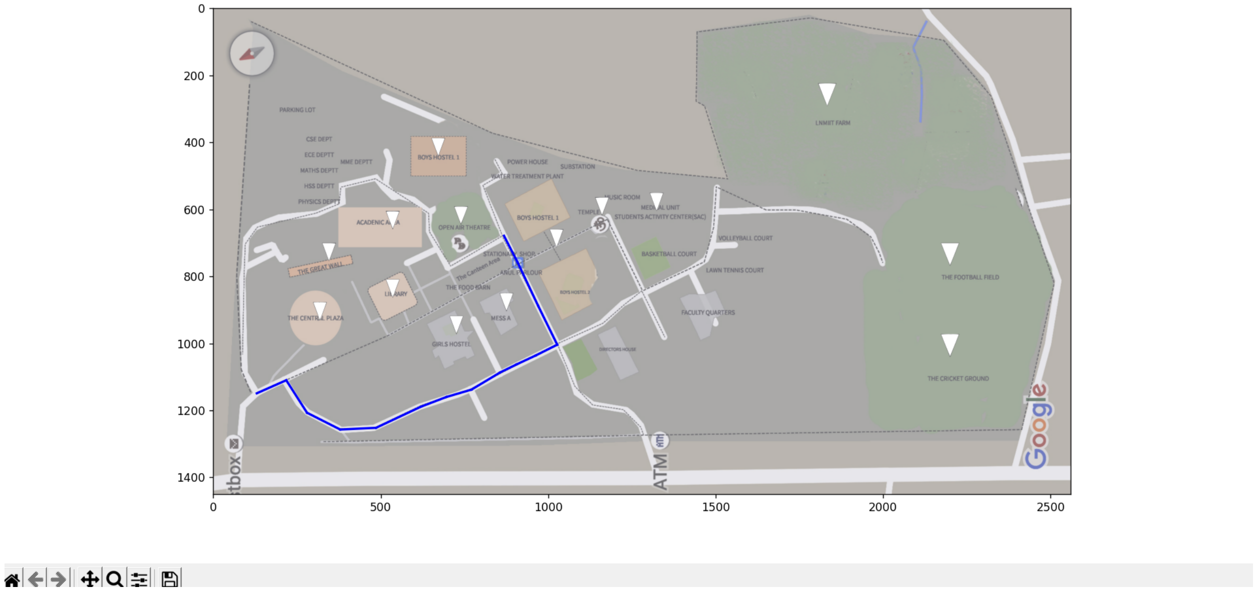
where $(x1,y1)$,$(x2,y2)$ are coordinates in a 2-D grid.

A* Algorithm

```
1. Initialize the open list
2. Initialize the closed list put the starting node on the open list (you can leave its f at zero)
3. While the open list is not empty
        a) find the node with the least f on the open list, call it "q" (Here we used min heap to minimize the complexity of finding least from the list)
        b) pop q off the open list
        c) generate q's successors and set their parents to q
        d) for each successor
                i)  if successor is the goal, stop search successor.g = q.g + distance between successor and q  successor.h = distance from goal to successor (This can be done using many ways, we will discuss three heuristics- Manhattan, Diagonal and Euclidean Heuristics) successor.f = successor.g + successor.h
                ii)  if a node with the same position as successor is in the OPEN list which has a lower f than  successor, skip this successor
                iii)  if a node with the same position as successor is in the CLOSED list which has a lower f than      successor, skip this successor otherwise, add the node to the open list end (for loop)

        push q on the closed list
        end (while loop)
```

RESULTS



*Output is the path for the given input coordinates

```
Starting Coordinates = (130, 1149)
Destination Coordinates = (868, 679)
```

Comparison of Algorithms

Using Dijkstra’s Algorithm

```
Total nodes visited - 11992000
Total time Dijkstra's --- 94.24340987205505 seconds ---
```

Using A* Heuristic Algorithm

a. Manhattan Distance as Heuristic Value

```
Total nodes visited - 5285700
Total time A* (Manhatten) --- 43.287054777145386 seconds ---
```

b. Euclidean Distance as Heuristic Value

```
Total nodes visited - 5931900
Total time A* (Euclieedean) --- 52.367024183273315 seconds ---
```

CONCLUSION

Problem that we faced in this project is firstly that using the Dijkstra’s Approach it could take so much time and space to reach out the result So we solved it using the informed search approach.
But using only the informed search will not make sure that we will reach out to our result which means that , the optimality of A* algorithm is heavily dependent on the quality of its function of evaluation.

Admissible Heuristic
Write h*(n) = the true minimal cost to goal from n.
A heuristic h is admissible if
$h(n) <= h^*(n)$ for all states n.
An admissible heuristic is guaranteed never to overestimate cost to goal.
A* with Admissible Heuristic Guarantees Optimal Path.

So we have to use the right heuristic values to reach the correct result quickly.
So we got the best results for these heuristics

1. Manhattan Distance(Used this in Project)
2. Euclidean Distance

From the project we can say the more improved heuristics value will be faster the run time and lower the time complexity we can get.

Future Improvements: For different Situations we can learn to use different heuristic values for example a path containing traffic we can increase the heuristic value as it will become less favorable to use then the original situation and using the heuristic updates we could possibly reach the complexity of O(n).