

# Complex Grid Path Optimization



**INDIAN INSTITUTE OF TECHNOLOGY JODHPUR**

**SEMESTER V OCTOBER 2020**

**Guided By Dr. YASHASWI VERMA**

Submitted By :

**Kartik Vyas**


**(B18CSE020)**

## Introduction

- ❖ The project deals with a problem based on finding out the best available path in terms of resources utilization (primarily the cost).
- ❖ Various paths would be given in the form of a grid, along with their costs of traversal and what kind of path they are and the utilities they offer.
- ❖ A car with a definite fuel capacity has to reach its destination keeping in mind the costs that would be incurred during the trip.
- ❖ The motivation for this project came from the idea of minimizing the traffic and petrol inputs on the roads. As in this way, different cars could work around what path would suit them after consideration of a good number of factors.
- ❖ The problem is based upon A \* Search algorithm. At each node of the path, the car makes the decision where to head and what path would be most suited based on the heuristic function.
- ❖ Overall, at the end the path with the minimum costs would be provided as the result along with the associated cost. While considering the heuristic for the next possible paths, the cost of reaching the current node would also be considered to give the best results.
- ❖ The reason why A\* search has been chosen after careful consideration and why it works to accuracy is that it considers the sum of expected cost and the cost already incurred and then with the help of data structures give priority to those which are the most promising, that is those which have the least costs, not only of the heuristic but the sum of heuristic along with current costs.
- ❖ The problem deals with various different utilities that are alongside the roads, such as petrol pumps, toll tax stations, potholes, or blockages (expressed in form of potholes). With these, the cost is considered accordingly, to overall yield the best path with various factors at play.

## Problem Description

A grid has been given with each element considered as a path. The vehicle is initially at one corner of the grid and has to reach the opposite corner. The elements of the grid are of five types : simple road, busy slow road, petrol pump station, potholes




and toll tax stations. Each of these have different features associated with them mentioned below :

- ❖ Simple Roads : The cost of traversing would be the only cost with no other complexity. Simple roads are represented by 'O' on the grid.
- ❖ Busy Roads : The cost of traversing would be higher as that from the case in the simple road. Busy roads are represented by 'S' on the grid.
- ❖ Petrol Pumps Stations : The vehicle must traverse through one of the stations before the vehicle runs out of the fuel. The vehicle is allowed to fill till full capacity at every station. Petrol pumps stations are represented by 'P' on the grid.
- ❖ Potholes : These are used to symbolize the potholes, construction works ongoing and blockages on the road. The vehicle cannot pass through this path. Potholes are represented by 'H' on the grid.
- ❖ Toll Tax Stations : If the vehicle must pass through one of these stations, then a certain amount of toll must be paid. The drivers thus would want to choose a path with a few of these, however keeping in mind the petrol cost and tank capacity as well. Toll tax stations are represented by 'T' on the grid.

The user can choose to generate a custom grid, alongwith the toll tax payment amount, fuel capacity of the vehicle, traversal cost of a simple road, traversal cost of a busy road and all the other details. The user can also choose from already generated default test cases, these cases are made to cover all the different possibilities. The users can also generate a random grid and all the costs associated would be generated randomly as well, this can be quite useful to check the correctness of the algorithm.

It is assumed that the vehicle would start from the left topmost corner of the grid with a fuel tank capacity, that is the left topmost node would always be a petrol pump. From here, the vehicle must traverse on a path minimizing the costs incurred. A very important aspect is that the vehicle needs to traverse through the petrol pump stations, otherwise the vehicle would halt and the user would not be able to reach the desired station which is the right bottom most node.

The potholes cannot be traversed over at any cost, they have been used to symbolise the blockages of the road. Another type of sub path is the one including toll tax utility, where some amount would be needed to be paid for the vehicle to



traverse. Now, the drivers would obviously want to avoid the path, but that might incur changing the path that might result in the need for traversal of more sub paths or those sub paths which are slow due to traffic and the need for fuel would also play a substantial role. Thus there would be factors at play and we will need to find the most optimal route.

Some technical aspects are mentioned below :

**States :** The vehicle at a particular node/subpath would be the state. The neighbouring states would be the four neighbouring states to which the vehicle can travel in up, down, left and right direction, whichever applicable for the concerned state.

**Cost Function :** The cost to travel to another state would depend on the state to which we want to transgress. If the state is a pothole, then the cost is very large (that is, not finite) and if the state is a toll tax station, then the cost would be the payment that would be needed to pay at the station. In case of empty and busy sub paths, the costs to transgress would be as mentioned / entered by the user.


**Start State :** The start state would consist of the vehicle at the left topmost path with full fuel tank capacity.

**Goal State :** The goal state is the car in the right bottommost subpath with the least possible cost incurred.

## Background Survey

This problem has wide implications and usage in real life. While online GPS tells us the path with minimum distance to reach to a particular destination. This approach can be used to customize the path according to the convenience of the users. All the different kinds of subpaths that have been used, that is potholes, empty roads, busy roads, toll tax stations, petrol pumps have been chosen to account for different types of needs of the users.

For example, if a user has a medical condition and is in dire need of washroom every 20 miles of commute then this problem can address this by the same method that is solved for the refuel issue at the petrol pumps. If a user does not want to go through a particular road for a reason, let's say that the road is too narrow, is accident prone, then this can be considered as a pothole to find out a route which



does not include this road. Similarly disinterest of different degrees can be represented by the busy roads or toll tax stations.

In this way, the aim is to solve the problem of finding a path which is suitable for the users, we are taking the data that is already provided to us by GPS and several datasets and then, this data can be used to achieve our required goal. Currently this problem is represented in the form of a grid, and this can be extended to work on graphs and then further on maps.


Various websites provide such maps and suggest the best path. Google scores the road segments based on factors like the shortest distance, the length of connecting road segments, and the traffic conditions at the time of the day and it returns us the highest scoring route, and some runner-up alternatives. We are adding different parameters to it, which would be of great importance and advantage for the masses.

## Discussion

The idea of finding a suitable route between two locations has been around in the last decade. The proposed problem suggests the consideration of a lot more factors as well, which also adds to the complexity of the problem but at the same time also has large potential as it can solve the problem of finding a convenient path according to a lot of factors and customized as well.

The problem offers novelty in the problem that it is solving and the factors that have been considered. Also, it must be noted in current times, if we want a path that does not include a particular road, or has washroom facilities consistently or we want a path that is time effective or cost efficient, then the users must search for them separately and specifically and then come with the suitable route. However, in this problem the user can give different weightage / importance to different utilities, that could give the suitable route.

The solution comprise of A\* search and is based on the easily computable heuristic function. Also the solution takes up a problem that is present in the real life world and is scalable and can be of huge advantage for the users. Thus, the proposed solution is an engineering based solution. It uses the known algorithm with modifications and optimizations and solves the problem in a driven approach.



The problem has been currently implemented with the help of a grid, this can be extended to graphs and later on actual maps, where user can put in the locations and the customized factors if they want to, or choose from the sets given. Then with the help of data that is already collected, such as location of medical shops, washroom availability, usual traffic, number of turns, the optimal path could be returned. This has the potential to be integrated with the platforms like Google Maps and address the masses to be a possible solution for finding out the most suitable path.

A presentation and demo video link has been attached, refer those for validation and demonstration.

## Algorithm

The approach that has been used to guarantee the correctness of the optimal solution is A\* search. A\* search algorithm is one of the best and popular technique used in path-finding and graph traversals. A\* search is usually preferred for problems including one source and one destination, and our problem has that only. A\* search is a very intelligent search algorithm as it makes decisions on the basis of the current cost function and the heuristic function. That is, actions are taken on the basis to minimize the total costs incurred and these costs are approximated with the help of two costs. In order to decide the next move, we check for all the sub paths and consider the sum of two values, the cost required to reach the subpath and the cost to reach the next subpath from there. Then these values are added to the priority queue and the route that has least cost associated with it is considered. This approach and computation is repeated until we find the destination node or the priority queue runs empty. In case we find the destination, the first path itself would be the optimal path as this would be the one with the least cost incurred and in case, the queue goes empty, it can be concluded that the destination cannot be reached under the mentioned constraints.


Now coming to the part of calculation of the cost incurred, we can compute the cost required to reach that subpath by just keeping the variable for this specifically inside the priority queue, this would be an exact value. It is to be noted that the other part, that is the heuristic functions are approximate measures which give the correct part in most of

the cases. The heuristic that has been used for the purpose of this has been derived by applying and validating it over many test cases and makes sense logically.

```
5  int heuristicValueCalculator (vector <vector <char> > &grid,int fuelLeft,int fuelCapacity,
6  |   int payToll, int payTravel, int paySlowTravel, int x, int y,int rows, int columns)
7  {
8  |   int heuristicValue;
9  |   heuristicValue = rows+columns;
10 |   heuristicValue= heuristicValue - (x+y)-2;
11
12 |   if(grid[x][y]=='T')
13 |   {
14 |       heuristicValue=heuristicValue+(payToll/3);
15 |   }
16 |   if(grid[x][y]=='S')
17 |   {
18 |       heuristicValue=heuristicValue+(paySlowTravel-payTravel);
19 |   }
20 |   heuristicValue=heuristicValue - fuelLeft;
21
22 |   if(fuelLeft<=0)
23 |   {
24 |       heuristicValue=INT_MAX;
25 |   }
26 |   if(grid[x][y]=='H')
27 |   {
28 |       heuristicValue=INT_MAX;
29 |   }
30 |   return heuristicValue;
31 | }
32
```

The above is the heuristic function that is used, it takes input of the path grid given, the fuel left, the fuel capacity, the tax to be paid at the toll station, the traversing cost over simple and busy roads, and the current position. The heuristic function is a representation of how far the current state is from the goal state. Thus, the less the heuristic function value the less it is away from the goal state. This is the reason why we use a minimum priority queue for the purpose of A\* algorithm.

Coming to the calculation, till line 10 we are calculating how many boxes is the current state away from the goal state, that is this is the Manhattan distance calculation. Then in the next two if conditions, we are checking if the state is a busy (slow due to traffic) subpath or is a toll tax station, according we increase the



heuristic value, as we do not want these two kinds of states. The degree of increment depends on the factors as to how much expected cost would be needed to traverse these subpaths.

Now, if the fuel left is more, it is desirable, thus in line 20, we are subtracting it from the heuristic to decrease the value in case of more fuel left. In the next two if conditions, we obviously cannot traverse a subpath with zero fuel or if the state is a pothole, thus we assign them the maximum heuristic value to make sure they are not considered for the consideration of the possible optimal path.

An algorithm that need not give the optimal solution but is time efficient would be to use the greedy algorithm and just switch on to the nearest petrol pump station. This would not consider the cost of toll tax stations or busy road traversal, but would have less time complexity and would give the path in case of large constraints, though the path not necessarily be the optimal solution.

Thus A\* would be a very efficient solution for our problem and with the help of a minimum priority queue and a good heuristic function, we are getting the optimal route effectively.

## Algorithm Complexity

Space Complexity : The space complexity is  $O(N*M)$  since it stores all generated nodes in the priority queue in the worst case. Here, N is the number of rows and M is the number of columns.

Time Complexity : In A\* search, the time complexity is highly dependent upon the computation of heuristic function. In the worst case of an unbounded search space, the time complexity is  $O(N*M)$ . It can be stated that the worst case time complexity is  $O(\text{Number of States} * \text{Complexity of heuristic function})$ . In our case, the complexity of heuristic function is  $O(1)$ , thus, the time complexity is  $O(N*M)$ .



## Worked Examples

### Example 1

```

*****
!! Welcome !!
*****

Would you like to create a new grid [Enter N] or use the default one [Enter D] or create a randomly generated grid [Enter R]?
D
Choose the default grid :
1 to 7 :
3

The grid is :

  0 1 2 3 4 5
0 P 0 0 0 0 0
1 P S 0 S 0 P
2 P 0 P T P 0
3 P 0 S S 0 0
4 P 0 0 H 0 S
5 P 0 0 T 0 0

The fuel capacity of the vehicle is : 4
The cost of one traversal is : 1
The cost of one traversal for slow path due to traffic is : 3
The units of amount needed to be paid on one toll tax station is : 10

The most optimized path is :
0 0
1 0
2 0
2 1
2 2
1 2
1 3
1 4
2 4
3 4
4 4
5 4
5 5
The total cost [toll tax stations + petrol cost for traversing of both types of road] of the most optimized path is : 14

```

This example uses a default grid as the input with the traversal on a simple road as 1 unit and on a busy road as 3 units, the amount to be paid at toll tax stations is 10 units and the fuel capacity is 4 units.

Starting from (0,0), the goal should be to reach (2,2) to plan out the further journey otherwise we cannot traverse the third column. To reach (2,2), we choose a path which is without any slow road and we have fuel at every subpath. From (2,2), we need to reach (2,4) now we can choose two paths, upper and lower; we choose the upper path to choose traversal of one busy road instead of two. Then, we reach the final goal by the optimal path without any busy roads or toll tax, while also keeping a check on the fuel. Thus, the path that we have chosen in the solution is the optimal one with a cost of 14 units as mentioned.

## Example 2

```

*****
!! Welcome !!
*****

Would you like to create a new grid [Enter N] or use the default one [Enter D] or create a randomly generated grid [Enter R]?
D
Choose the default grid :
1 to 7 :
6


The grid is :

 0 1 2 3 4 5 6
0 P T 0 0 H 0 0
1 T T 0 0 0 0 0
2 S 0 P T P 0 0
3 P T 0 0 0 0 0
4 0 S P H 0 0 0
5 0 0 S T P 0 0
6 0 0 P 0 H P S

The fuel capacity of the vehicle is : 3
The cost of one traversal is : 1
The cost of one traversal for slow path due to traffic is : 3
The units of amount needed to be paid on one toll tax station is : 10

The most optimized path is :
0 0
1 0
2 0
3 0
4 0
4 1
4 2
5 2
5 3
5 4
5 5
6 5
6 6
The total cost [toll tax stations + petrol cost for traversing of both types of road] of the most optimized path is : 40

```



This example uses a default grid as the input with the traversal on a simple road as 1 unit and on a busy road as 3 units, the amount to be paid at toll tax stations is 10 units and the fuel capacity is 3 units.

Starting from (0,0), the goal should be to reach (3,0) as it is the only petrol pump within the limits to reach under a 3 units fuel capacity. To plan out the further journey, we must choose either from (4,2) or (2,2); we should choose (4,2) as it is closer to the goal state and another petrol pump station as well and we can reach the next petrol pump station, which is not the case in (4,2). To reach (5,4), we need to traverse a toll tax station, which leads us to our final goal by the optimal path without any busy roads or toll tax, while also keeping a check on the fuel. Thus, the path that we have chosen in the solution is the optimal one with a cost of 40 units as correctly mentioned.

## Codebase Link


<https://github.com/vyaskartik20/PathOptimization>

## Summary and Conclusions

A \* search algorithm is a very efficient algorithm that helps us to find the optimal path with a lot of constraints and factors that have been given. The problem can be expanded upon and has a lot of scope for implementation in real life as then users can get customized optimal routes according to their convenience. If proper data and resources are available, this can be integrated with the GPS services, the idea is scalable.

Upon learning the A\* search algorithm, I was fascinated by how optimized and effective code is, and uses heuristic function to find out the most optimal solution. It also considers the cost to reach a particular point and then consider all the neighbouring states to find out the one with minimal cost.

The algorithm is efficient and thus can be used for large test cases as well. It was challenging to come up with an accurate heuristic problem since the problem deals with a lot of different kinds of states, which can be further extended as well



according to real life use. Various examples were used to build the heuristic and the whole program was tested upon manual and randomly generated inputs as well.

## References

The project idea was thought of by me in the pursuit of building something novel and I have not referred to any sources. By attending the classes of CS323 IIT Jodhpur, I developed the sense for A\* search and have implemented it from scratch and by using a proper heuristic function by testing on many examples.