A* ALGORITHM TO FIND THE OPTIMAL PATH IN TRANSPORT SYSTEMS

CIS4049-N-BF1-2021

SCHOOL OF COMPUTING, ENGINEERING AND DIGITAL TECHNOLOGIES

JANUARY 14, 2021

Prepared by

B1326237 Gowthami Nagappan

Abstract

In this study, we are proposing a model to find the optimal path which also works efficiently within a loop structure. We are using 'A*' search algorithm to find the shortest distance between the start and end point. The algorithm is better when comparing with other algorithm which are not much efficient and cannot solve complex problems. Using 'Google maps' as a base for finding the active paths, the nodes and edges are plotted on map. The information of all the cities, nodes and paths are plotted on the algorithm. This project will serve as a boon for many sections of people who are daily commuters to work, courier drivers and food delivery drivers. Additionally, there are some taxi companies which operate locally, and they can rely on the efficiency of this project as this is based on a specific cluster. The project can be implemented on a larger scale with widening the base of the city points by giving accurate information. Further scope is that the algorithm can also be revised easily which keeps a track on the last point and from there the distance can be calculated with accuracy which will recalculate based on any major obstacles found. We can delve into the project by creating clusters for a specific set of distance and then linking them as nodes between different clusters which can provide real-time efficiency on controlling the travel distance which will have direct impact on the cost. The shortfall of the project is confined to certain limitations as there might have increased overheads when the size of the plot is increased. Our research paper can provide a base for future researchers who are working on the algorithm to overcome this obstacle.

Keywords - A Star Algorithm, Path planning, Transport System

Introduction

In the past days, there were many problems when travelling from one place to other. So, there is a need of finding a short distance from one point to another with the minimal cost as possible. Also, the project will discuss about further scope which can be implemented when working on a broader scale.

Except for the well-known person who knows the location very well were able to drive through various routes but for a new commuter it was almost impossible to do the same as they must rely on the sign board of the cities on the local roads and highways to get the required information. Then with the introduction of satellite navigation devices which were able to point out to the direction where the commuter needs to travel. Even with those instruments the accuracy and connectivity issues were a big problem. To cite an example, the food delivery drivers used to face many issues when they work for a take-away or a big restaurant. In initial days when a driver is recruited for a food delivery job, they were asked about the license they hold which is either provisional / full and the type of vehicle (motorbike/car) they hold. Additionally, they were also asked about the knowledge of the particular area where they are going to deliver the food which was a major obstacle for a driver if they have not lived in the area. The resident drivers in the area would be more comfortable in delivering in the area compared to the new drivers who need to keep the physical map on hand for navigation purpose which is a tiring process. This delays the delivery time and increase the cost for the restaurant as they need more drivers. After the introduction of google maps, the work is now easier for

the delivery drivers as they don't need to have prior experience or knowledge on the area. They can now easily drive with the help of a 'smart phone' and access to 'google maps' which are freely available on the play store. Many companies like Deliveroo, Uber eats and just eat have now integrated google maps with their online delivery platform which is helping the drivers to find the restaurants and easily navigate to the customer's place.

There is another example where I would like to talk about the taxi industry which is worth billions of pounds. In the initial days, the taxi industry is not much popular as the same problem what described under 'food delivery' were impacting this industry as well. Getting into as a taxi driver will need to have in the initial days been difficult as the area knowledge is very important for a faster commute. In bigger cities like London and Manchester where there are several small roads keeping everything on back of the mind is very difficult for a driver. The shorter the commuting time, a driver can earn a higher level of income. Therefore, the industry was open only to a few drivers who had very good knowledge on the area.

The introduction of 'Google maps' was a boon to major industries including the general public. The application of google maps has various applications which include Store locator, free space display, street view, specific parking spot, logistics companies, electric vehicle range planning, geo coding and fin-tech companies. It is just a few applications which have listed here. There are lot more applications than the one mentioned here.

As the project, describing here are localized to a specific cluster which will have applications in restaurants and hotels operating locally. Further it can be implemented with local logistics providers and taxi companies. Compared to 'google maps' which work on a large scale the projects where work will integrate as much information as possible due to the smaller cluster size. Due to the limited information available at this point of time, finding the shortest path or distance from the start to end point.

Further scope and research can be made on creating various local clusters. These clusters in various regions can be clubbed to form a single point so the area can be increased. This will specifically help the local companies when they want to expand their operational area. When specific clusters are formed, the real time data can be integrated easily when compared to the bigger clusters where feeding information will be difficult. This real time data can include any important functions, public rally, and obstruction in the road due to works carried out by the public works department.

A* algorithm is one of the best path finding algorithm which can be used to find the easiest way to move from one node to the other. Even though it works on the principles of 'Dijkstra's algorithm, the use of 'Heuristics functions' will provide the solution in an easier way while travelling from one point to another in terms of distance covered and the cost reduction. It is a kind of interconnected nodes where the cost is the actual cost of the start node and the estimated cost of the target node. It uses the result to select the next node to evaluate. Also 'Dijkstra's algorithm evaluates various routes which in unnecessary and can be ignored through this method. Even 'Google maps' work on this algorithm and it has the capacity to accept large amount of data which is then used to calculate the short distance between starting and end point.

A* algorithm also has certain limitations when too much of grid size are included and, in that case, need to make some modifications by using 'Heuristics' and 'secondary components' which can increase the efficiency of the algorithm. There are many other path finding algorithms which were developed based on 'A*' are more advanced and can provide better solution in terms of cost reduction and efficiency. The following papers in method section, reference which had provided the required inputs for this project and were used as a base for developing this project.

Methods

3 different projects which have used 'A*' algorithm are taken as reference

"A systematic literature review of A* path finding" by Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah , Eric Gunawan published in 'International conference on computer science and computational intelligence 2020'

"Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment" by Gang Tang, Congqiang Tang, Christophe Claramunt, Xiong Hu and Peipei Zhou

"Improved A-Star Algorithm for Long-Distance Off-Road Path Planning Using Terrain Data Map" by Zhonghua Hong and others

"Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem" by Dian Rachmawati and Lysander Gustin

There are 3 important parameters when using the algorithm. These are of utmost parameters which will help us in identifying the shortest route possible with high level of accuracy. These parameters are G(N), H(N) and F(N). In respect to G(N) it calculates the cost of travel from starting point to another point and in relation to H (N) it calculates the distance taken to travel from starting to destination point. F(N) is the combination of both the parameters G & H. The decision made using algorithm is based on the 'F' value cell. The smallest 'F' value cell is chosen by the algorithm and then it proceeds to multiple cells until it finds the target or goal cell. The formula is computed as per the below:

$F(n) = G(n) + h(n)$

In case F(n) is considered as the overall cost of the path, G(n) is considered as the cost which incurred until the past step and h(n) is considered as the goal point where the end cost is calculated based on the distance covered until this point.

In simple terms, F(n) is called as an intrinsic function and H(n) is completely based on prediction which is the end node point.

Operational process of Algorithm

The structure of the algorithm is very simple where it uses the above equation for calculation. The priority queue is used in the algorithm as the data structure where the lowest value of f(n) is also considered the highest priority node.

The algorithm will pick the node which has the lowest value and then it kicks it out from the queue which has other nodes as well. Then the other nodes in the same will get updated and the cost of the node from one to our priority is calculated which takes on F (n).

The same steps will get repeated and again and it's the normal process of algorithm. Until the target node is identified, the algorithm will perform the process. Finally, the process will go on until the node is in the priority queue or the nodes are not present in the queue.

The algorithm was used for finding the process on the graph from starting node point to end point. The aim of the algorithm is to fine the shorter path between the start point and point. The user will provide the value from where he needs to start and gives the end point where he wants to reach so the A* algorithm is used to find the shortest path using the graph.

The main step of the algorithm as follows:

1. Algorithm will generate open list, closed list and path list. It will start with the open list which is also called as start node.
2. Algorithm will go to the open list at the initial stage and it will check for any values present. If in case the list is non-empty it will find the lowest value and then it will take it to the close list where it will be considered as the current or stating point.
3. The neighbouring node will be checked by the algorithm to find whether its on the open list. If its not on the open list, then the current point should be added to the path list where the Heuristic distance is calcuated. If it is on the open list, then the calculation of 'G' value is performed. It will travel to other feasible nodes and if not then the current one needs to be added to the path list and the the distance is calculated.
4. Iteration will keep on running until it reaches the end point where there are 2 scenarios. One scenario is if the open list is empty or the current point is on the closed list which will mean that there is no path found. The other scenario is where the current point or node which becomes the end point. It means that the final shortest path has been identified.

Algorithm pseudo code (Step by step process)

The input will be the source location whereas the end node will be the destination and the output will be the number of locations visited by our algorithm and provide us with the shortest optimal route. Please find the steps which were performed.

Starting with necessary packages like networkx, plotly graph objects and matplotlib using pip install method in Anaconda

Initialising Graph with the 12 Locations with the distance as heuristic value and cost to travel from each location to neighbourhood locations.

Then display the graph model of the locations inputted with distance and cost

1. Initialise: Graph route List
2. Pathfunction(Start, end)
3. Initialzing the global keywords
4. Initializing the Closed and Open List
5. CLOSED = EMPTY LIST
6. OPEN (START , 11)
7. TOTAL COST OF NODE VISITED 0
8. While (True)
9. Fn = i in opened
10. Choosing the min fn
11. Current node will go to closed
12. Break the loop if already destination found
13. for if the node is visited will add into closed list contiously untill destination

Getting Input from user

- Getting the input from user about Source and Destination Location
- If the input not found print(" Source or Destination Location does not exists")

Displaying Plot in Graph

- importing networkx
- importing matplotlib.pyplot
- To create an empty graph
- Graph nodes and weighted edges assigned to E
- Adding edges from E to the Graph G
- Taken a Spring layout to display Graph
- mentioned the features to draw graph
- Displaying the Final Plot

The goal node is denoted by end and the source node is denoted by start

maintained two lists: OPEN and CLOSED:

OPEN consists on nodes that have been visited but not. This is the list of task to be performed.

CLOSED consists on nodes that have been visited and expanded

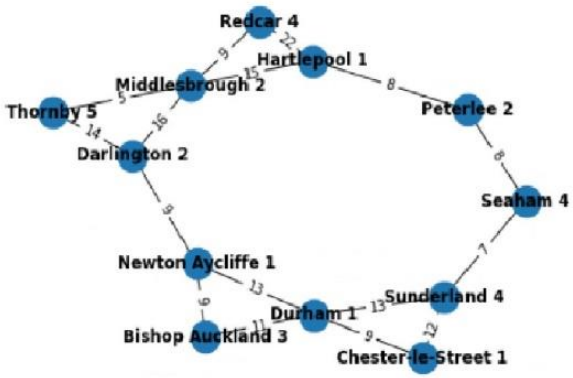
1. Put list of Parent and Child nodes in ‘Tree’ Dictionary
2. Put list of Heuristic vales in ‘heuristic’ Dictionary
3. Def function AStarSearch (Source, Destination)
4. Initially CLOSED = empty list
5. Opened = Source, 11
6. While True
7. F(n) calculated for each node
8. Select the Minimum f(n) value
9. Current Node will be Opened List
10. Visited Node will be updated in CLOSED
11. break the loop if node Destination has been found
12. or Continue with the calculation of f(n)
13. Display the Best Optimal Route with Visited locations
14. check whether h(n) + g(n) = f(n). If so, append current node to optimal sequence
15. change the correct optimal tracing node to current node
16. Reverse the optimal sequence
17. Print the visited Nodes with Optimal Path found

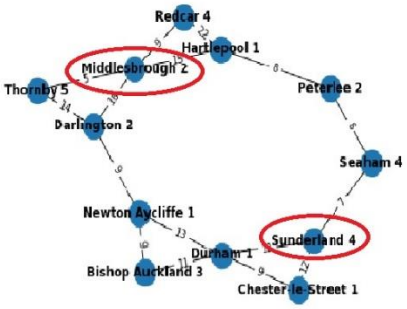
Result

The final route which the Algorithm has identified as the shortest route is:

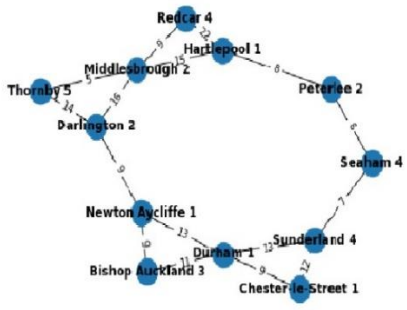
Sunderland –-> Seaham –-> Peterlee –-> Hartlepool –-> Middlesbrough

The F(n) value for the shortest distance covered from Sunderland to Middlesbrough is 40.



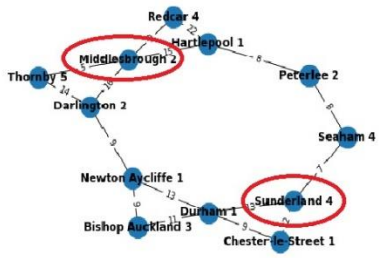



```
Source Location: Sunderland
Destination: Middlesbrough
Visited nodes: [['Sunderland', 11], ['Seaham', 11], ['Chester-le-Street', 13], ['Durham', 14], ['Peterlee', 17], ['Durham', 2
2], ['Hartlepool', 24], ['Newton Aycliffe', 35], ['Newton Aycliffe', 35], ['Bishop Auckland', 37], ['Bishop Auckland', 37], ['M
iddlesbrough', 40]]
Best Shortest Path: ['Sunderland', 'Seaham', 'Peterlee', 'Hartlepool', 'Middlesbrough']
```




```
Source Location: Sunderland
Destination: Middlesbrough
visited nodes: [['Sunderland', 11], ['Seaham', 11], ['Chester-le-Street', 13], ['Durham', 14], ['Peterlee', 17], ['Durham', 2
2], ['Hartlepool', 24], ['Newton Aycliffe', 35], ['Newton Aycliffe', 35], ['Bishop Auckland', 37], ['Bishop Auckland', 37], ['M
iddlesbrough', 40]]
Best Shortest Path: ['Sunderland', 'Seaham', 'Peterlee', 'Hartlepool', 'Middlesbrough']
```

Current Location		Visited Location			Closed Location
		G(n)	H(n)	F(n)	Location
Sunderland	Seaham	7	4	11	Seaham (Expanding)
	Chester-le-street	12	1	13	Hold
	Durham	13	1	14	Hold
Seaham	Peterlee	15	2	17	Peterlee (Expanding)
Peterlee	Hartlepool	23	1	24	Hartlepool (Expanding)
Hartlepool	Red Car	45	4	49	Hold
	Middlesbrough	38	2	40	Middlesbrough (Goal)




```
Source Location: Sunderland
Destination: Middlesbrough
visited nodes: [['Sunderland', 11], ['Seaham', 11], ['Chester-le-Street', 13], ['Durham', 14], ['Peterlee', 17], ['Durham', 2
2], ['Hartlepool', 24], ['Newton Aycliffe', 35], ['Newton Aycliffe', 35], ['Bishop Auckland', 37], ['Bishop Auckland', 37], ['M
iddlesbrough', 40]]
Best Shortest Path: ['Sunderland', 'Seaham', 'Peterlee', 'Hartlepool', 'Middlesbrough']
```

Discussion

The performance of A* algorithm is evaluated by using 12 different cities where the cost and miles are the important coordinates. It will provide the users with the shortest path available from the starting point to end point. In terms of the cost, it provides the user with a cheapest cost for travel from start point to end point. In relation to miles, it provides the shortest distance available where the user can travel to the destination point. With different values of f(n), the shortest path is calculated with the proposed algorithm.

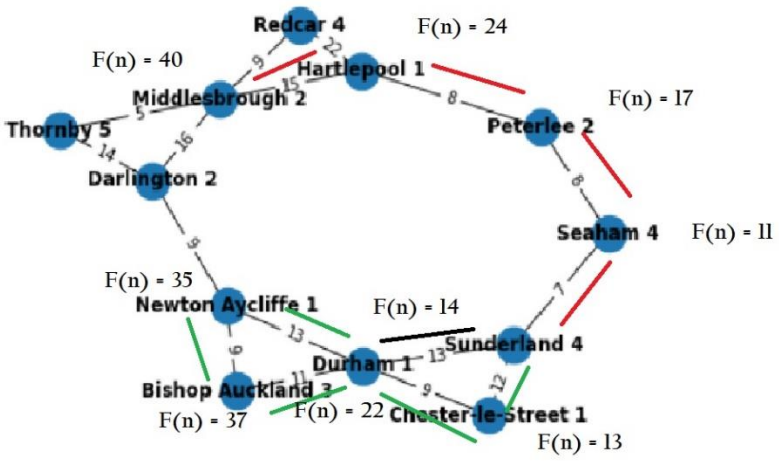
Input data

When the alogrithm starts running, the priority queue is generated which will identify all the possible points and it will take the point closer to the destination from the starting point into account. In the inital phase the cost is not taken as a priority. The queue will have all the possible points where it considers the shortest way or path to the end point. It takes into account all the possible paths which are viable where the time taken to travel from start to end point and the lowest miles travelled is taken into account.

In the normal process of A* search alogrithm, in each step of the process the node point is picked which is according to the value of f (n). This parameter is equivalent to the sum of both g(n) and h(n). In each and every step, the cell point with the lowest value of f (n) is taken into consideration.

In simple terms, I would like to explain about the process which will provide the clarity.
get the source and destination location from the user and need to make sure that the user provides with the correct input and avoid inputting invalid data. The alogrithm will consider the ‘source’ as ‘start’ point and the user destination as ‘goal point’.

For example the user wants to travel from Sunderland to Middlesbrough. There are three routes which connects from start city to end point. The intermediate cities which can connect to Middlesbrough are Chesterly or Durham or through. Seaham. As already discussed earlier that, A* algorithm always calculate f(n) value to select the path. In this case, F(n) of chesterly street is 13 ; F(n) of Durham is 14 and F(n) of Seaham is 11. When comparing all these values, the F(n) value of Seaham is 11 which is the lowest value and correcponds to the shorterst route. In the next process, the same steps discussed above will get repeat from the cities (Chesterly / Durham / Seaham) to Middlesbrough and correspondingly F(n) calculation will get done. Until the destination point is reached F(n) calculation will take possible through all routes.



Comparison of A* alogrithm vs Greed first search algorithm

The main difference between A* and Greed first search algorithm is the evaluation function. In respect to Greedy, F(n) is equivalent to H(n) and in the case of A*, F(n) is equivalent to G(n) and H(n). A* uses more memory than Greedy as it takes multiple ways to find an optimal path whereas in Greedy, the memory used is lower and it does not find the optimal path.

In relation to A*, it looks at all the paths from start to destination point whereas in Greedy it traverses to the location which is nearer to the goal. This way of model is very efficient when compared to DFS. also, there is a higher chance of getting stuck in a loop is higher in this case.

A* does not cross beyond the estimate whereas in case of Greedy the estimate value might get surpassed.

For example, user planning for travel from one city to another with a shortest path or distance and also wants to limit the number of cities which travelling through with the least number of steps.

Whenever user arrive at any point before reaching the destination city, user would be able to know the distance from that point to the destination city. The distance approximates how close the goal from the given node which is denoted by the heuristic function H(n). The greedy algorithm can switch between BFS and DFS where the BFS stands for Breadth first search and DFS stands for Depth first search. It has the advantages of both of these. The heuristic value is mentioned within each node. This is not always equal to the actual road distance as the path will have many curves while moving up a hill.

H(n) value in the graphical representation for each location and also the number of miles to reach the point is given. Our plot clearly shows, what are the possible ways from the starting point to the destination as given by the user. Our algorithm will start at the source location and then goes to all the possible nodes which are closer to the destination point by using both the BFS and DFS. The same gets performed at each and every stage until it finds the closest point to our destination point.

In the graph map, for example if the source location as Durham and the destination as Darlington. The cities on the graph as the location points and the path between cities as edges, finally need to reach Darlington which is the destination point.

The final output which will get generated as per the algorithm is:

Durham →Newton Aycliffe→ Darlington

Looking at a number of advantages, A* algorithm has significant advantages compared to other types where it has the ability to resolve many complex problems.

Conclusions

From the above we can conclude about the various advantages offered by ‘’A* in comparison with the ‘Greedy best search’ in terms of efficiency, travelling around the loop, path optimisation and the speed of the search.

Matplotlab is used for displaying 12 cities on the graph which includes plots of the cities, cost to travel and Heuristic functions. The road depicted on the graph is using Google maps as a reference. For better understanding, instead of using maps in this program, plots on the graphs are used so the different nodes of travel can be easily identified. After getting the user input of both the source and destination place, F(n) for each travel path gets calculated and all the values gets stored in the memory. using this algorithm on a smaller scale of approx. 25-kilometre distance whereas the same can be implemented on a larger scale with many cities. Multiple travel propositions can be made such as traveling on a car, bike, motorbike and HGV vehicles.

There are some limitations in using the algorithm which are already discussed in this project. It also sets a scope for future work to be carried out which can make the process more efficient and applicable in various circumstances.

Recommendation

As Google maps has certain limitations in regard to the regional areas in getting the correct updates on time which will result in problems for the user when calculating the optimal path with an efficient cost. Instead, the big maps can be broke into small clusters where the data feed will be much easier when operating on a smaller scale. When the feeds are updated from these clusters, then the process would become much easier and effective when they are integrated into a larger scale. Also, the clusters would be beneficial for the companies which are operating on the smaller scale regional areas. Local Logistics companies, courier drivers, food delivery drivers and local commuters would be the ones which will get benefitted from this model. The same can be incorporated in highly congested cities located in India and China where unexpected things happen on the road where larger scale data feeds are difficult.

References

"A systematic literature review of A*pathfinding" by Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, Eric Gunawan published in 'International conference on computer science and computational intelligence 2020'

https://www.sciencedirect.com/science/article/pii/S1877050921000399#!

"Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment" by Gang Tang, Congqiang Tang, Christophe Claramunt, Xiong Hu and Peipei Zhou

https://ieeexplore.ieee.org/document/9391698

"Improved A-Star Algorithm for Long-Distance Off-Road Path Planning Using Terrain Data Map" by Zhonghua Hong and others

https://www.mdpi.com/2220-9964/10/11/785

"Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem" by Dian Rachmawati and Lysander Gustin

https://iopscience.iop.org/article/10.1088/1742-6596/1566/1/012061/pdf