Intelligent Route Finding System using M Star Algorithm

Madhu Amarnath[1] Sankar Palanisamy[2]
[1]Assistant Professor [2]Assistant Consultant
[1]R.M.K. College of Engineering and Technology, Chennai [2]Tata Consultancy Services, Bengaluru

Abstract— The increased traffic and complex modern road network have made finding a good route from one location to another a difficult task. There are many search algorithms that have been proposed to solve this problem. The most well-known among them are Dijkstra's algorithm, and A*. While these algorithms are effective for route finding, they are wasteful in terms of computational overhead. In this paper a novel algorithm M*(pronounced as M Star) is suggested for intelligent route finding which uses better evaluation criteria and the notion of cumulative proximity score. This proposed technique will reduce the time and space requirements in computation, and also produces better result in terms of accuracy and shortness of path found.

Key words: Dijkstra's Algorithm, Route finding

I. INTRODUCTION

Route finding (or better known as path-finding in the field of Artificial Intelligence), is a process of finding the least-cost route given a source-destination pair. It has many uses, ranging from transportation, networking and gaming systems to helping robots and travellers navigate through an environment.

There are many algorithms that have been devised to solve route finding. Some of them are Breadth-First search, Depth-First search, Random Walk, Hill Climbing, Dijkstra's Algorithm, Bellman-Ford, Johnson Algorithm, and A*.

The route finding algorithms are used for cell layout and channel routing in VLSI layout. It is extensively used for routing in computer networks. Robot navigation (which is generalization of route-finding problem) is useful in automatic assembly sequencing. The more recent application of robot navigation is Internet searching. Transportation application includes airline travel planning system.

II. METHODOLOGY

To find the shortest path between two places one needs the distance information. The road network is represented as a search tree with reference to an adjacency matrix (i.e. two dimensional array) showing the connectivity between cities and towns. Each city/town in the network holds its respective geographical coordinates. Distance between two cities/towns is obtained mathematically with respect to these coordinates based on "haversine" formula as follows:

```
Distance = r * acos[sin(lat1) * sin(lat2) + cos(lat1) *
cos(lat2) * cos(lon2 - lon1)]
```

where longitude *lon* and latitude *lat* are in radians, and the earth's radius *r* is 6378.7 km.

III. SEARCHING TECHNIQUES

A. *Dijkstra's Algorithm:*

In 1959, Dijkstra introduced a single source shortest path algorithm which was named after himself [4, 5]. This technique is a combinatorial algorithm of best first search and breadth first search [4] due to expansion based on adjacency matrix of all nodes forming a network.

Dijkstra's algorithm starts by assigning infinity as default score to all nodes except the source. Candidate nodes for subsequent computation will be stored into a priority queue according to adjacency matrix that shows directional connectivity between nodes. Priority queue integrates best first search notion for computational purposes. Each connection edge has values which imply the respective distance from one node to another.

As the computation domain expands to the entire network, node score of adjacent nodes will be updated to hold the shortest distance from source to the current node. This process will continue until the priority queue is empty, i.e. the goal is reached.

The major shortcoming of Dijkstra's algorithm is the time and space requirement. Especially for huge networks, the computational cost for it to work is far too excessive. This technique integrates greedy best first paradigm in locating the local optima as a heuristic to obtain global optimum. Nevertheless, a lack of direction differentiation features greatly reduces its efficiency in route finding process. A similar characteristic is found in A*. Therefore, improvement is needed.

B. *A* Search:*

A* search is a heuristic search algorithm and it is an extension of best first search. It is complete i.e. it will always reach the destination. A* is the optimally efficient of all algorithms used in route finding that works by expanding root node.

C. *Evaluation Function for A* Search:*

It uses the following evaluation function in choosing the next node in the shortest path:

$$f(node) = c(node) + t(node)$$

Where,

- f(node) = node's score
- c(node) = cost to travel from source to current node
- t(node) = estimated cost to travel form current node to destination

The node with lowest score is the better one and it is chosen as next node.

D. *Drawbacks:*

1) Time complexity and Space complexity of the algorithms increases exponentially with the complexity of the problem domain.

2) It does not specify how close the current node is to destination node.
3) In some cases, A* approach produces the same score for two nodes at different locations.

Figure 1 shows an example in which the limitation of A* approach produces the same score for two nodes at different locations. Considering node X and node Y which are located on the straight line between the source and destination, these nodes will have similar score due to the location differential incompatibility of the evaluation function of A*. [6]

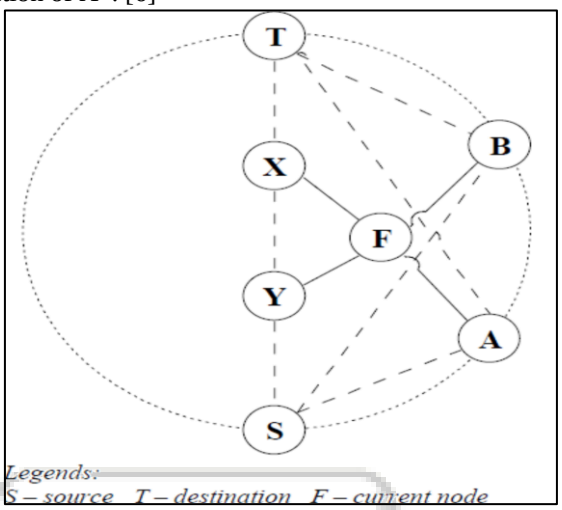

Fig. 1: Limitation of Conventional A*

This is because it does not determine the proximity of current node to the destination due to the summation operator. Similar situation is encountered when two nodes are located at a hemispheric curve, where diameter is the straight line between the source and destination, i.e. node A and B.

E. Variants of A:*

There are some variants of A* developed to overcome space complexity. Some of them are Iterative-deepening A* (IDA*), Memory-bounded A*(MA*) and Simplified MA* (SMA*). Though they did ensured optimality and completeness, they compromised on execution time [3].

F. Modified A:*

Further analysis is performed upon the A* technique based on physical displacement efficiency. A node with good score is a node that is:

1) far from source – high c(Node)
2) close to destination – low t(Node)

Hence, the following evaluation function is instead used for our modified A*:

$$f(node) = c(node) - t(node)$$

Assignment of negative sign to the term *t(Node)* allows displacement differential compatibility. In other words, node A and B and node X and Y will have respective scores which are different from each other. As opposed to the conventional A* evaluation, higher score from the modified evaluation function is the better score. Both positive and negative scores are possible despite no negative weight existed between cities/towns. Positive score suggests that the node is closer to destination than to source and vice versa. Higher score implies that a node is travelling further away from source and closer to end node. [6]

IV. M* ALGORITHM

The approach outlined in [6] does not always produce optimal solutions if the graph tends to be sparse. So, in this paper a better alternative is suggested. The objective of optimality, completeness, less time and space complexity is ensured by this proposed algorithm. It borrows some of its ideas from [6].

The practical evaluation function for the M* search considers the cumulative score for two subsequent states. A node with high score is desired in the proposed search because a good node is:

1) far from source – high c(Node)
2) close to destination – low t(Node)

The cumulative score is:

$$f(node) = f1(node) + f2(node)$$

where,
f1(node) = c1(node) - t1(node)
f2(node) = c2(node) - t2(node)
f1(node) = score of the 1st node
c1(node) = cost to travel form source to 1st node
t1(node) = cost to travel from 1st node to destination
f2(node) = score of the 2nd node
c2(node) = cost to travel form source to 2nd node
t2(node) = cost to travel from 2nd node to destination

V. CONCLUSION

In this paper, the shortcomings of some conventional search algorithms like Dijkstra's Algorithm and A* is discussed.

The idea of location differential compatibility and cumulative proximity score borrowed from [6] is used in this paper for efficient and effective in intelligent route finding.

An application can be developed for Intelligent Route Finding System. A sample application for Chennai city which demonstrates M* search algorithm can be implemented.

Future work will investigate problems with higher level of complexity. The breadth of the proposed cumulative evaluation can also be further expanded.

REFERENCES

[1] Hart, P., Nilsson, N., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Syst. Science and Cybernetics, SSC-4(2):100-107, 1968.
[2] Hart, P., Nilsson, N., and Raphael, B., "Correction to 'A Formal Basis for the Heuristic Determination of Minimum-Cost Paths'," SIGART Newsletter, no. 37, pp. 28-29, December, 1972.
[3] Russell, S. and Norvig, P. (2007) "Solving Problems by Searching & Informed Search and Exploration", Artificial Intelligence - A Modern Approach, pp. 87-164.
[4] M. Puthuparampil, *Report: Dijkstra's Algorithm*, tech. report, Computer Science Department, New York University, 2007.
[5] J. Siek, *Bellman-Ford Shortest Path*, Boost C++ Libraries, Indiana University, 2000;

http://www.boost.org/libs/graph/doc/bellman_ford_shortest.html

[6] Yang Yaw Chang, Stephen Yung and Raymond Chiong "A Novel Approach for Intelligent Route Finding through Cumulative Proximity Evaluation", Second Asia International Conference on Modelling & Simulation, pp. 672-677.

