iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.34974

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Shortest Path for Emergency Services: A Comparative Analysis

Jenil Jain[1], Bhavyansh Upadhyay[2], Kartikeya Pande[3], Harshit Singh[4], Omkaresh Kulkarni[5]
[1, 2, 3, 4]*U.G. Student,* [5]*Assistant Professor, School of Computer Science and Engineering, MIT World Peace University*

Abstract: ***In emergency situations, choosing a wrong route or farther end point may increase the travel time. Discovering the shortest path between the start and end point in a road network is a difficult task. This paper provides an analysis of a few path finding algorithms like Dijkstra's and A-star with various heuristic functions, used to find the shortest route. We also propose a system which can be used in case of emergency situations like accident, theft, etc to reach the emergency destinations as fast as possible, using Geographic Information Systems (GIS) based system and leaflet.js.***

Keywords: A-star Algorithm, Dijkstra's Algorithm, Heuristic function, Geographic Information Systems (GIS), Open Street Map, Leaflet.

I. INTRODUCTION

It is becoming difficult to find the best route to reach any emergency systems like hospitals, police stations, etc. especially in metropolitan cities which have huge road networks.

Geographic Information System (GIS) is a special information system that contains data related to geographic coordinates i.e. longitude and latitude, with road network connectivity. It also aims to assist decision-making and policies related to geographical information in a region.

One of the major implementations of GIS is to find the shortest route of travel from one point to another.

Shortest path finding is a very well-studied problem in the field of Computer Science and it has numerous real-world applications, such as detection of shortest path between source and destination on a map autonomous, navigation of a robot, determining the shortest network route, etc. If we convert the map into the graph environment, then, at its very core, all the pathfinding problems address the question of how to reach a destination node from a starting node in a graph. This can be done by implementing a graph search algorithm for the given problem, which searches the graph starting from the start node and exploring the adjacent vertices of the visited nodes until it reaches the destination node.

Dijkstra's algorithm is one of the most widely used algorithms by researchers to solve this problem. However, Dijkstra's algorithm does not handle memory usage, so this algorithm will use large memory space to store the points passed over each iteration when searching for the shortest route from a very large graph. Dijkstra cannot compete with more recent methods. A* tries to improve Dijkstra's performance by only changing the routing algorithm, which results in small speedups.

Therefore, this article aims to present a comparison between several path finding algorithms and thus provides the most efficient algorithm which can be used in case of emergency cases like accidents, theft, etc. to find nearest hospitals, police stations and other emergency services as per requirement.

II. PROBLEM DESCRIPTION

The aim is to design a system that finds the shortest path between a source location and a destination location within a city for emergency services. Shortest Path to reach various emergency services like hospitals, nursing homes, police station, fire station, vaccine centres etc can help in preventing lives, hazards, fuel. These paths can lead to the shortest path with least cost and least time duration.

To discover the shortest path between a source and a destination in a road network is a difficult task. In emergency services, choosing a wrong route or farther end point may increase the travel time. Delaying any emergency services for a few seconds can cause drastic effects on an individual or an authority.

In our project, we propose to examine factors that have an impact on routing algorithms consistency. Also, we will propose a comparative study between various routing algorithms. Our aim is to design a platform that finds the shortest path from a source to a destination of a particular city for emergency systems. Using the proposed method, even the longest routes can be calculated within efficient time, while shorter routes take only a few seconds.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

III. DATA DESCRIPTION

As mentioned before, a data source that can supply accurate data on the complete road network in Yavatmal is a requirement for the success of the project. Openstreetmap.org and Harvard Dataverse offer such a data source.
OpenStreetMap (OSM) is an open source map. Anyone can contribute GPS data, which is used to form a dataset describing all the roads in the world. The data is by far not complete, but fortunately Automotive Navigation Data (AND) has donated a street network of the entire Yavatmal . As a result the data is fairly complete for the Yavatmal and a suitable source for the route planner.
The Harvard Dataverse is an open source data repository, where you can share, archive, cite, access, and explore research data.
The entire street network for the Yavatmal (or any other cities) can be downloaded as two files consisting of node and edge files. Node file contains osmid, latitude and longitude and edge file contains the nodes it covers, length and several tags to indicate the type of road.

IV. METHODOLOGY

A. Haversine formula

The haversine formula is used to find distance between two points on a sphere using their latitudes and longitudes measured along the surface. The project uses the haversine formula twice. Firstly, while finding the 5 hospitals, which are nearest to the mishap. And secondly, while calculating the value of heuristic in A* algorithm.

The haversine formula can be used to calculate shortest point to point distance (d) as shown:

$$d = 2r \sin^{-1}\left(\sqrt{sin^2\left(\frac{\Theta_2 - \Theta_1}{2}\right) + cos(\Theta_1)\, cos(\Theta_2)\, sin\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

where:

(φ_1, λ_1) and (φ_2, λ_2) are coordinates of the two points in radiance,

Θ is angle between these 2 points

r is radius of the sphere (radius of earth was taken as 6378100 meters)

Thus, the obtained value of shortest distance(d) was used as per requirement.

B. Dijkstra's Algorithm

Dijkstra's algorithm is an algorithm for locating the shortest paths between nodes within a graph, which shall represent road networks. The algorithm can work in two ways; finding the shortest path between two nodes (single source- single destination problem), and a more common one which fixes a node as source node and finds shortest paths from the source to all nodes in the graph, producing a shortest path tree.

C. A algorithm*

A* algorithm is a graph search algorithm that finds the shortest path from a given initial node to a given goal node in that road network-based graph. It employs a "heuristic estimate" h(x) that provides an estimate of the best route that goes through that node. It follows the approach of best first search.
It does that by combining the pieces of data that Dijkstra's algorithm uses (favouring vertices that are closer to the starting point) and knowledge that Best-First Search uses (favouring vertices that are close to the goal).
In the standard terminology used for A*, g(n) represents the precise cost of the path from the start point to any vertex n, and h(n) represents the heuristic estimated cost from vertex n to the goal. The algorithm evaluates the best next step in searching for a path. This is done by evaluating each of the possible next steps against a heuristic to give a value that can be used to sort the list by evaluating each of the possible next steps against a heuristic to give a value that can sort the list and hence determine the most likely step. As one can imagine this makes choosing the best heuristic for your map really important to getting good pathfinding performance.
f(n)=g(n)+h(n)

D. Pseudocode

The goal node is denoted by node_goal and the source node is denoted by node_start

We maintain two lists: OPEN and CLOSE:

OPEN consists on nodes that have been visited but not expanded (meaning that sucessors have not been explored yet). This is the list of pending tasks.

CLOSE consists on nodes that have been visited and expanded (sucessors have already been explored and included in the open list, if this was the case).

H represents the heuristic function which will return 1 for best results.

1) Put node_start in the OPEN list with f (node_start) = h(node_start) (initialization)
2) while the OPEN list is not empty{
3) Take from the open list the node node_current with the lowest
4) f (node_current) = g(node_current) + h(node_current)
5) if node_current is node_goal we have found the solution; break
6) Generate each state node_successor that come after node_current
7) for each node_successor of node_current{
8) Set successor_current_cost = g(node_current) + w(node_current, node_successor)
9) if node_successor is in the OPEN list {
10) if g(node_successor) <= successor_current_cost continue (to line 20)
11) } else if node_successor is in the CLOSED list{
12) if g(node_successor) <= successor_current_cost continue (to line 20)
13) Move node_successor from the CLOSED list to the OPEN list
14) } else {
15) Add node_successor to the OPEN list
16) Set h(node_successor) to be the heuristic distance to node_goal
17) }
18) Set g(node_successor) = successor_current_cost
19) Set the parent of node_successor to node_current
20) }
21) Add node_current to the CLOSED list
22) }
23) if(node_current != node_goal) exit with error (the OPEN list is empty)

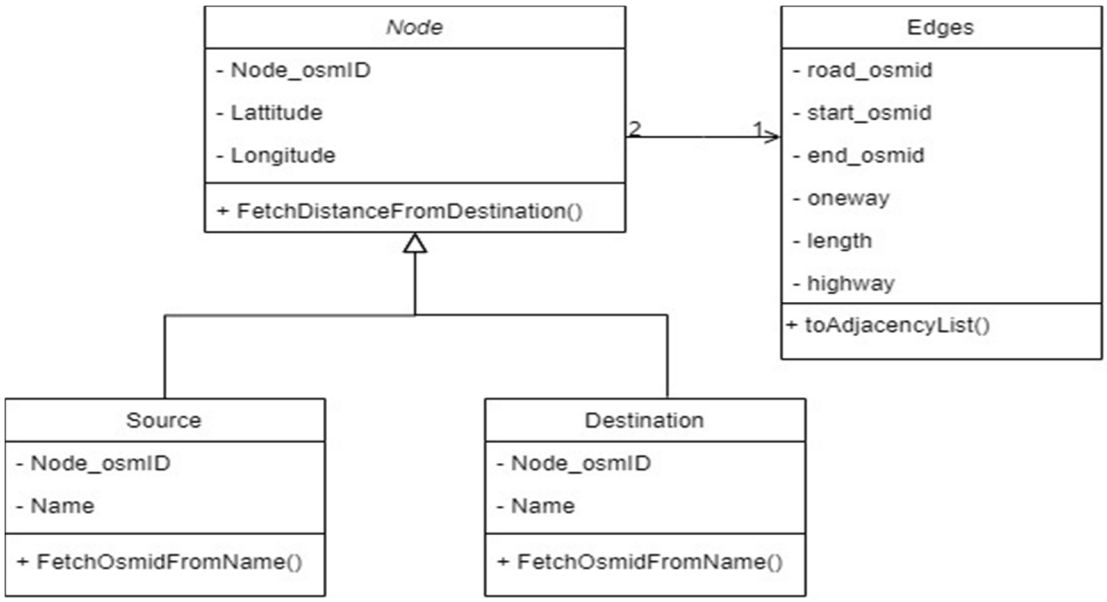


Fig. 1: Class Diagram

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

E. Leaflet.js

Leaflet is an open-source JavaScript library for building web mapping applications. It supports most of the mobile and desktop applications via a browser. It allows developers to display tiled web maps, even without having to use GIS. It also has features that can be added to the map such as overlays and markers. It also supports GeoJSON layers. We have used leaflet.js for displaying the path obtained by the algorithm on the front end. It takes the array of latitude and longitude from the backend and uses it to plot the route on a map as shown in Fig. 2. The project also provides the facility of getting the current location of the device.

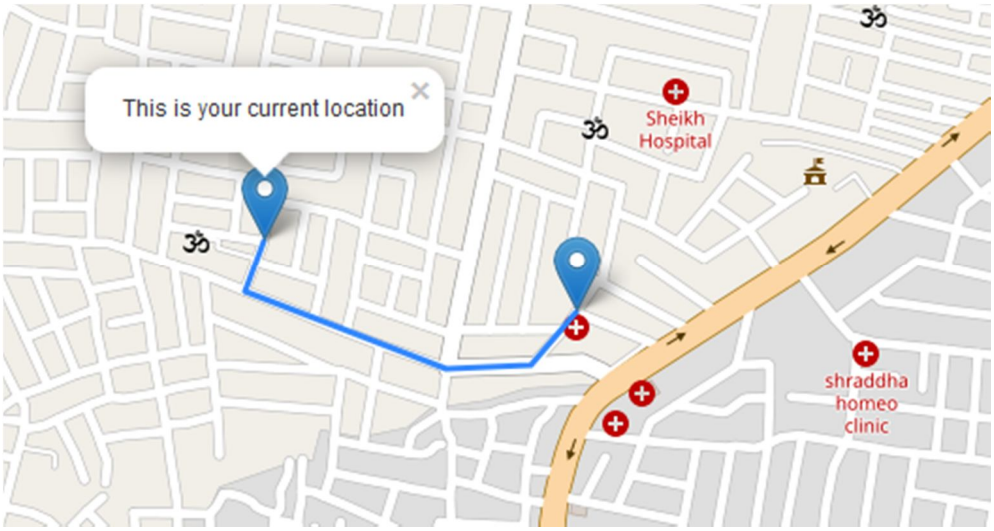


Fig. 2: Visualizing shortest path using leaflet.js

V. ANALYSIS

The comparative study was done to analysis time taken to generate shortest path for each of the three algorithms namely dijkstra, A* with heuristic as haversine and A* with heuristic as 1. Fig. 3 shows the result of the case study.

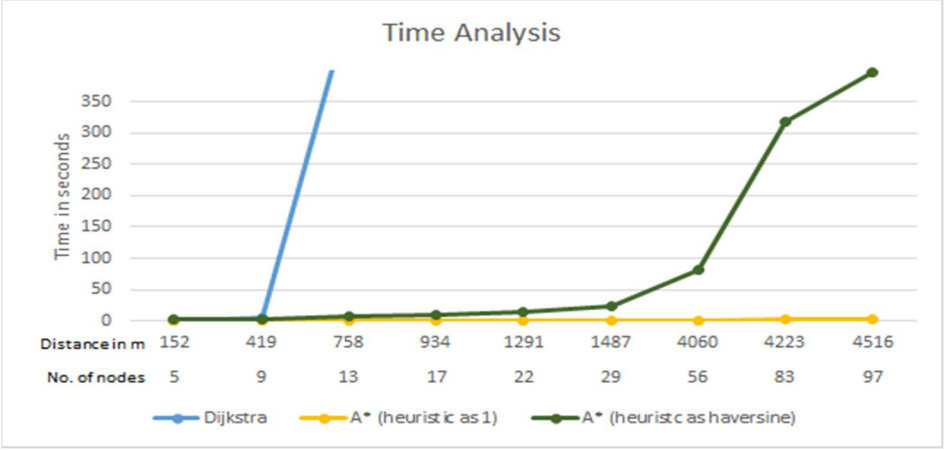


Fig. 3: Comparison of algorithms

The above graph has time taken in seconds on y axis and distance between the source and destination in meters on x axis. Also the x axis has the number of nodes the path traversed while going from source to destination.From the graph we can observe that upto 419 meters or 9 nodes, all the three algorithms give results in acceptable time. After which, dijkstra shows RAM exceeded and thus takes infinite time. It states that for shorter distances like this, dijkstra is preferred as it has less space complexity as compared to other two algorithms. Later till 1000 meters or number of nodes equal to 15, both the A* algorithms give results in a few seconds. But after 1000 meters, A* with heuristic as haverine breaks the threshold and takes more time for computation.

The best suited algorithm according to the case study, turns out to be A* with heuristic as 1. It takes less than 1.5 seconds to find the shortest path separated by a distance of around 5000m i.e. 5km. For any city in India, it was observed that at least one emergency system like hospital, police station, etc exists in the radius of 5km thus, use of this algorithm for path finding is most preferred

VI. CONCLUSION

The most well-known algorithm for solving a shortest path problem is Dijkstra's algorithm. However, when implementing an emergency service, where delays cannot be more than a few seconds, Dijkstra does not always suffice in terms of speed. A* with heuristic 1 and haversine are two methods that only adjust the route calculation algorithm and lead to very small time.

When using these methods, routes can be found thousands of times faster than when using Dijkstra. For online route planners, this means that the calculation time can be decreased to a few milliseconds. With such a time efficient algorithm, there is no longer a bottleneck, but overhead like transmitting the data over a network or displaying the route will.

Using this knowledge, an emergency service route planner for the Yavatmal was implemented. The route planner was written in Python and uses A* to calculate the routes, based on data from openstreetmap.org and Harvard dataverse, which offers a complete road network for the Yavatmal.

Performance statistics show that even the longest routes can be calculated within a second. The performance of A* over Dijkstra is impressive for longer routes, but the difference is almost negligible for very shorter routes. This makes A* stand out from the other route planners' algorithms when it was compared.

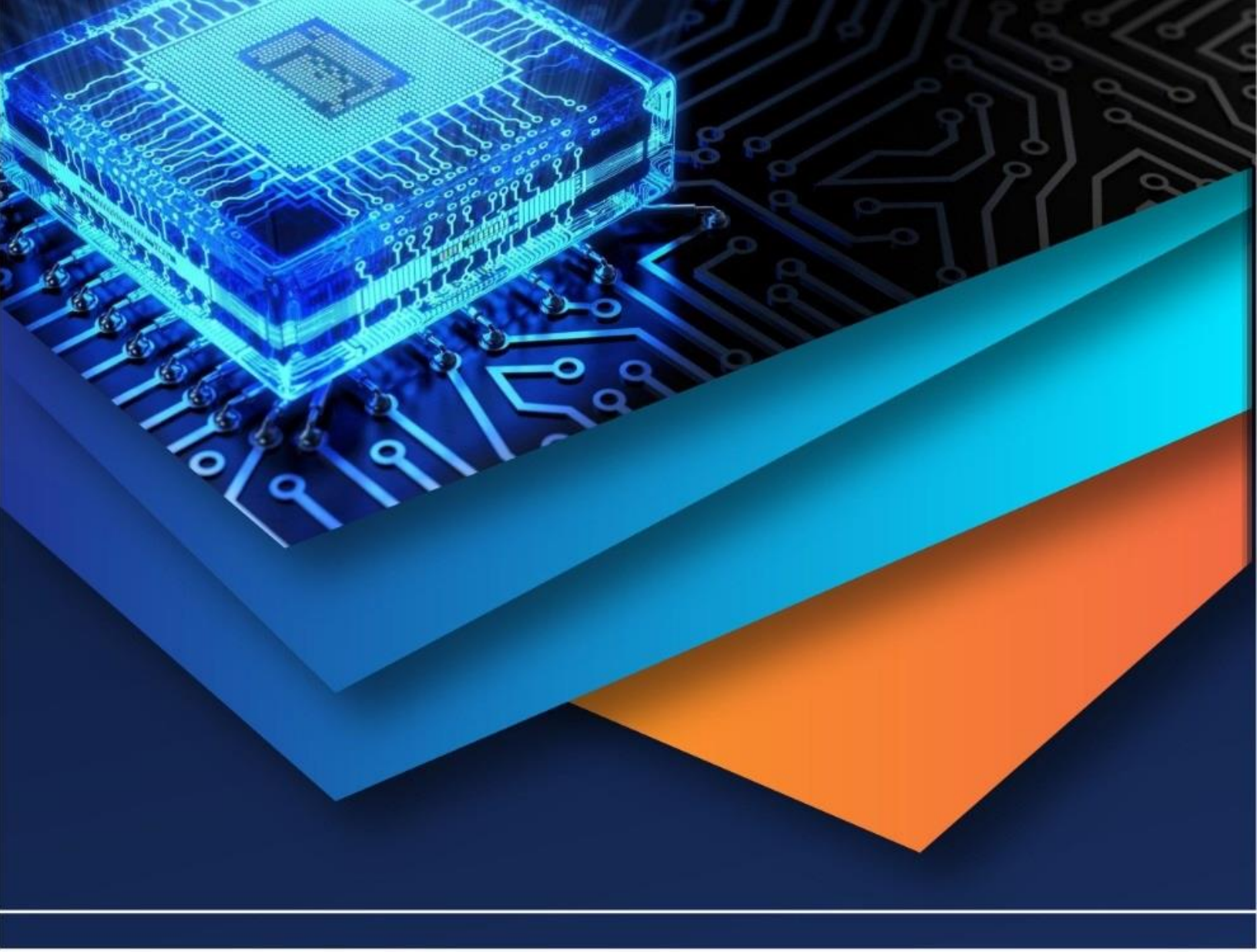
VII. ACKNOWLEDGEMENT

Firstly, we are grateful to Dr. Vishwanath Karad MIT World Peace University for allowing us to work on this project. We are fortunate to have worked under the supervision of our guide Prof. Omkaresh Kulkarni. His guidance and ideas have made this project work.

We express our sincere appreciation for the cooperation given by HOS of School of Computer Engineering and Technology, Prof. Vrushali Kulkarni for giving us access to all the resources that went into building this project.

REFERENCES

[1] Stuart E. Dreyfus, (1969) An Appraisal of Some Shortest-Path Algorithms. Operations Research 17(3):395-412.
[2] Rhyd Lewis "Algorithms for Finding Shortest Paths in Networks with Vertex Transfer Penalties", School of Mathematics, Cardiff University, Cardiff CF10 3AT, Wales, UK (2020).
[3] Achmad Fitro, Otong Saeful Bachri, Arif Ilham Sulistio Purnomo and Indra Frendianata, Shortest Path Finding in Geographical Information Systems using Node Combination and Dijkstra Algorithm, International Journal of Mechanical Engineering and Technology 9(2), 2018. pp. 755–760.
[4] C. Liu, M. Zhou, J. Wu, C. Long, and Y. Wang, "Electric vehicles en-route charging navigation systems: Joint charging and routing opti- mization," IEEE Transactions on Control Systems Technology, vol. 27,no. 2, pp. 906–914, Mar. 2019.
[5] Cui, G., Luo, J., Wang, X., 2018. Personalized travel route recommendation using collaborative filtering based on GPS trajectories. Int. J. Digital Earth 11, 284–307.
[6] Boeing, G. 2017. "OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks." Computers, Environment and Urban Systems. 65, 126-139.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY