ANT COLONY-BASED ARCHITECTURE FOR AIR POLLUTION AWARE VEHICLE ROUTING IN SMART CITIES USING MACHINE LEARNING

A Synopsis submitted to Gujarat Technological University

For the award of

Doctor of Philosophy

In

COMPUTER / IT ENGINEERING

By

VARIA DHAVALKUMAR JITENDRAKUMAR

Enrollment No. 169999913017

Under supervision of

DR. A.M.KOTHARI

ESTD - 2007

GUJARAT TECHNOLOGICAL UNIVERSITY

AHMEDABAD

AUGUST-2021

INDEX

Sr. No	TOPIC	Page No.
I	Title of the thesis and abstract	2
II	Brief description on the state of the art of the research topic	3
III	Problem Definition	4
IV	Objective and Scope of work	5
V	Original contribution by the thesis	6
VI	Methodology of Research, Results / Comparisons	24
VII	Achievements with respect to objectives	33
VIII	Conclusion	34
IX	A list of all publications arising from the thesis	35
X	Patents	35
XI	References	36

1. TITLE OF THE THESIS AND ABSTRACT

1.1 TITLE OF THE THESIS :

Ant Colony-Based Architecture for Air Pollution Aware Vehicle Routing in Smart Cities using Machine Learning

1.2 ABSTRACT:

The government of India has started the "100 Smart Cities Mission" project on 25th June 2015. The smart city is a live-capable, reasonable and flourishing economy offering numerous chances to their kin. The objective to establish a smart city is to distribute resources efficiently, to improve the communication between citizens and government, to help data-driven decision making, and to reduce environmental footprint, etc.

In smart cities, air pollution hazards are the central issue required to address. According to the WHO report of 2018, around 7 million premature deaths occurred due to air pollution, and more than 91% of the world population lives under severe pollution hazards [1]. To establish meaningful smart cities, it is essential to either reduce air pollution or moderate the impact of air pollution on the human body. But moving towards the zero pollution objective required lots of efforts and legislation.

Our research aims to reduce the effect of air pollution on the human body by selecting a better path during traveling, especially for the two-wheeler's rider, who suffers from diseases like aggravated cardiovascular, respiratory illness, accelerated aging lungs, asthma, bronchitis, emphysema, etc. Moreover, awareness about the current level of pollution will help them to take preventive actions.

We have proposed ant colony-based architecture for air pollution-aware vehicle routing in smart cities using machine learning to reduce the effect of air pollution on the human body during traveling.

The proposed architecture uses an ant colony algorithm to find the optimal path considering less distance and low air pollution level (in terms of air quality index) towards the destination. We have identified that the ant colony algorithm executes before the tour of the vehicle starts. Thus, it considers the current scenario (i.e., air pollution level, traffic volume) and suggests the best optimal path. However, the values of air pollution, traffic volume, etc., changes frequently with time. So, when the vehicle reaches the next junction, the scenario would get change. Thus the

vehicle ends up with a heavy volume of traffic or an increase in effective air pollution. We have proposed the air pollution prediction mechanism for all the intermediate locations before starting the journey to solve this issue. Based on this prediction, our algorithm selects the best optimal path. The prediction of air pollution is made using machine learning techniques. We have experimented with the two most popular machine learning techniques (1) Artificial Neural Network (ANN) and (2) Long Short-Term Memory (LSTM). Comparative analysis shows that LSTM for the time series problem is more promising. Thus we have considered LSTM for the prediction of air pollution. We have used a client-server approach to implement the architecture. The client module is developed on the Tkinter toolkit shipped with python. While server module is called using web service developed using python. The server module is designed using Traci and simulation of urban mobility simulator (SUMO) library.

In addition to this, We have proposed a modification in the ant colony system's edge selection process to detect and eliminate the cycle in the graph. The proposed algorithm reduces the ant to be stuck in the cycle of the diagram; this will let the ants contribute more to generate optimum solutions rather than stall out in the cycle.

2. BRIEF DESCRIPTION ON THE STATE OF THE ART OF THE RESEARCH TOPIC

From the literature study, we have identified that,

1. **Static shortest pathfinding algorithms**: Current study considered distance/traffic/Air Quality to find out the optimal route for the required destination using the Dijkstra algorithm[2] [3] [4] [5] [6], Genetic Algorithm [7], A^* Prune [8]. The Dijkstra algorithm is unsuitable for the problem where parameter changes are dynamic (like traffic and Air Pollution)[9]. The Dijkstra and A^* Family algorithms are the static shortest pathfinding algorithm that is not applicable where the road network changes occur [10] [11]. As per our knowledge, the only research article available on the topic uses the ant colony optimization technique to find the optimal path considering the distance, traffic, and Air Quality is [12].
2. **Selection of swarm optimization technique:** In computational intelligence, two optimization methods are prevalent (1) Particle Swarm Optimization (2) Ant Colony Optimization. Particle swarm optimization methods are simple and require less computation as compared to Ant colony optimization. However, it faces problems in combinatorial or discrete optimization problems like finding the shortest path depending on various parameters like Distance, Traffic,

and Minimal pollution. Thus we use ant colony Optimization to identify the optimal route from source to destination in our model [13].

3. **Solution of ant being stuck in the loop for the ant colony algorithm:** We have found that the ant colony system suffers from being stuck in the loop while solving the shortest path problem [14] [15]. The only work which we found in this direction is ACR Algorithm [15]. In this algorithm, If ants find the way where there are no nodes, other than choosing a node that makes a trap (here it can be identified using taboo list), ants start looking through another way. Perception is, this calculation permits all the ants to follow this way and afterward backtrack the ants. Which thus squander the computational assets and postpones convergence of the solution. So we find another way to build a novel system. So we have suggested the solution for eliminating the cycle in the graph during shortest path findings.
4. **Proposed dynamic ant colony algorithm:** Ant colony algorithm is applied before the beginning of the journey to find the shortest path [16] [9] [17] [18] [19]. The ant colony algorithm's dynamic application is proposed when ants reach each junction of the road network towards the destination to consider dynamic changes in the air pollutants' level.
5. **Proposed architecture for air pollution-aware vehicle routing in the smart city:** To eliminate recursive computation at each junction of the road network, we have proposed a machine learning approach. Using this approach, we predict air pollution before beginning the journey.

3. PROBLEM DEFINITION

According to the WHO report of 2018, around 7 million premature deaths occurred due to air pollution, and more than 91% of the world population lives under severe pollution hazards [1]. Usually, while selecting a route from one place to another, one chooses the shortest path or the path with lesser traffic density. However, in addition to the distance and traffic volume, it is more important to know the pollution level throughout the route for the two-wheeler's rider, who suffers from diseases like aggravated cardiovascular, respiratory illness, accelerated aging lungs, asthma, bronchitis, emphysema, etc. Moreover, awareness about the current level of pollution will help them to take preventive actions.

Our research suggests, all the $routes\ R$ head towards the destination and satisfy optimal parameters like distance and air quality. In our study, we consider the case of the urban area, where the traffic is dense, and thus the speed of the vehicles is very controlled.

In defining our problem, we consider þ as a set of routes that satisfies a set of constraints Ω. So our solution is to find a solution $S = (þ, Ω)$ where $Ω(t)$ are the constraints at time t between $þ(i, j)$. In which i and j indicate the section of route between $Junction_{\ i}\ and\ Junction_j$.

Vehicle V starts its journey from Source S towards Destination D. We will begin from the Solution S=Ø an empty set where Solution S is supposed to consist of all the route þ which satisfies constraints Ω. During the execution, Vehicle V will determine $r1, r2..rn \in R$, which is a set of routes available to destination D. Our proposed approach presents these routes to the user and helps them to select the optimal path. Each route in the set þ is represented with (I, E), Where I mean the Intersection/junction, and E is for the Edge/Road section that connects Intersection/junction.

4. OBJECTIVE AND SCOPE OF WORK

In our work, we have created the algorithm to solve the ant stuck in the cycle and proposed a methodology to implement a dynamic ant colony algorithm. Considering the dynamic ant colony algorithm results, we have presented the ant colony-based architecture for air pollution-aware vehicle routing in the smart city. This architecture uses our defined machine learning model to predict the air quality index for the road's respective junction for the Ahmedabad city, Gujarat.

The main objectives of the research are,

1. To suggest the procedure to identify and eliminate the cycle in the ant colony system's decision-making process. This study aims to show the number of ants stuck in the graph's cycle; if eliminated, results show improved computational efficiency.
2. To develop the dynamic approach for the ant colony algorithm. The selection probability P_{ij} is calculated dynamically before arriving at each junction of the route to consider the frequent changes in the parameter like air quality.
3. To compare artificial neural networks and long short-term memory techniques and develop an optimal model for predicting air quality for the Ahmedabad city.

4. To propose the ant colony-based architecture for air pollution-aware vehicle routing in the smart city using the machine learning model based on long short-term memory techniques.

Note : Probability p_{ij}^k to select the next node from all the available node is calculated using the equation:

$$p_{ij}^k = \begin{cases} \frac{\tau_{i,j}{}^{\propto} * \ \eta_{i,j}{}^{\beta} * \ \text{Pollution}_{i,j}{}^{\gamma}}{\sum_{j \epsilon N_i^k} \tau_{i,j}{}^{\propto} * \ \eta_{i,j}{}^{\beta} * \ \text{Pollution}_{i,j}{}^{\gamma}} \ If \ j \epsilon N_i^k \\ 0 \ if \ j \notin N_i^k \end{cases} \quad \text{---------- (equation 1)}$$

Where N_i^k is the practical neighborhood of ant_k when in $node_i$ and looking for available alternate $node_j$. η is called visibility and is calculated using :

$$\eta_{i,j} = \frac{1}{d_{ij}} \text{ ----------(equation 2)}$$

Where $\mathrm{d}_{i,j}$ *is lenght of path between* $node_i$ *and* $node_j$.

$\text{Pollution}_{i,j}{}^{\gamma}$ *is the Air Quality Index between* $node_i$ *and* $node_j$

$\propto, \beta, \gamma$ *are relative weight of the respective parameter*

5. ORIGINAL CONTRIBUTION BY THE THESIS

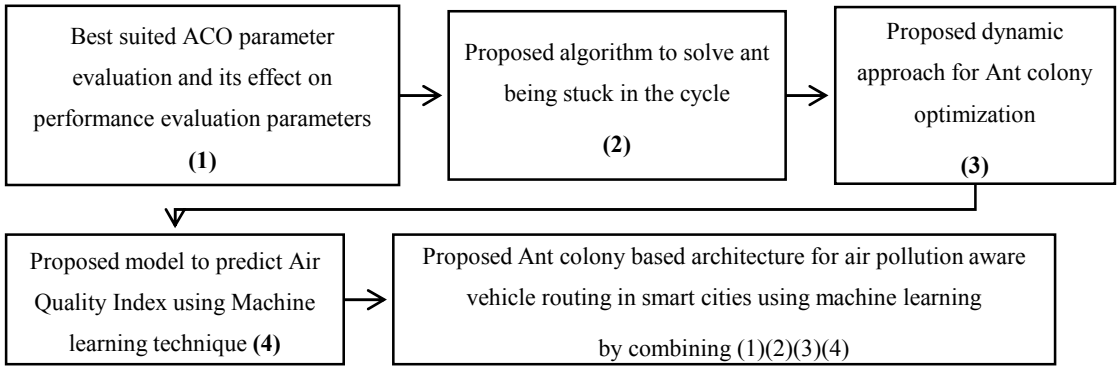


Figure : 1 Research Direction

Figure 1 shows the direction of our research. As per the problem identified and set objectives, we have devised the following:

1. Identify and eliminate the cycle in the decision-making process of the ant colony system [20].

2. The proposed methodology to implement an ant colony algorithm dynamically (DynamicACO) [21].
3. Proposed model for predicting the air quality index for the respective junction of the road for the Ahmedabad city [22].
4. Ant colony based architecture for air pollution aware vehicle routing in smart cities using machine learning.

(1) ANT COLONY OPTIMIZATION TECHNIQUE PARAMETER EVALUATION AND ITS EFFECT ON PERFORMANCE EVALUATION PARAMETERS:

Our proposed architecture is based on the ant colony optimization (ACO). The objective is to identify the optimal path for the constraints to minimize the distance and effect of pollution on the human body. Initially, we have identified the prerequisite parameter (transition rule parameter $\propto$ and the evaporation rate ρ) for the ACO to carry out further experiments.

Algorithm 5.1 Evaluate the transition rule parameter and the evaporation rate

```
1: procedure EVALUATEPARAMETERS
2:     transition rule parameter alpha ← [1,1.5,2,3,4,5]
3:     evaporation rate ← [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]
4:     for Each StartNode, EndNode ← SoiurceNodeList, DestinationNodeList do
5:         if StartNode != EndNode then
6:             for Each i ← lengthofarray : alpha do
7:                 for Each e ← lengthofarray : evaporationrate do
8:                     Execute ACO Algorithm and save trace to text file.
```

Algorithm 5.1 shows the procedure to evaluate the transition rule and evaporation rate.

The results of algorithm 5.1 are discussed in section 6.1.

We have carried out the experiment defined in (procedure 1) and (procedure 2) to analyze the effect of the transition rule parameter $\propto$ and the evaporation rate ρ on the performance evaluation parameter. For the assessment of $\propto$ and ρ, we have considered the following performance evaluation parameters:

1. Length of the path, i.e., distance
2. The number of Diffusions: This measure is defined as the total number of decision-making done by the ants based on the transition rule. As the number of Diffusions is low, i.e., respective parameter ($\propto$ and ρ) is best suited for our implementation.

Procedure 1: To obtain the min, max, and average of distance & diffusion by varying values of transition rule parameter ∝ with [1,1.5,2,3,4,5] and set evaporation rate ρ to 0.1 [Algorithm 5.2]

Algorithm 5.2 To obtained the min,max and average of distance and diffusion by varying values of transistion rule parameter with fixed evaporation rate

```
1: procedure EVALUATEDISTANCEDIFFUSION
2:     transistion rule parameter ← [1,1.5,2,3,4,5]
3:     evaporation rate ← 0.1
4:     for Each StartNode, EndNode ← SoiurceNodeList, DestinationNodeList do
5:         if StartNode != EndNode then
6:             for Each i ← lengthofarray : alpha do
7:                 Execute ACO Algorithm and save trace to text file.
```

Procedure 2: To obtain the min,max and average of distance & diffusion by varying values of evaporation rate ρ to [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8] and set transistion rule parameter ∝ 1.5 [Algorithm 5.3].

Algorithm 5.3 To obtained the min,max and average of distance and diffusion by varying values of evaporation rate with fixed transition rule parameter

```
1: procedure EVALUATEDISTANCEDIFFUSION
2:     transistion rule parameter ← 1.5
3:     evaporation rate ← [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]
4:     for Each StartNode, EndNode ← SoiurceNodeList, DestinationNodeList do
5:         if StartNode != EndNode then
6:             for Each i ← lengthofarray : evaporationrate do
7:                 Execute ACO Algorithm and save trace to text file.
```

The results of these experiments are discussed in section 6.2

(2) PROPOSED ALGORITHM TO SOLVE ANT BEING STUCK IN THE CYCLE [20]**:**

There is a requirement to suggest a novel method to identify and eliminate the cycle in the decision-making process of the proposed ant colony algorithm [15] due to the following reasons**:**

1. As shown in figure 2, suppose there are x no. of ants out of the total ants, follow the way 1->2->3->4->5 and stuck each time due to the path contains cycle like 1->2->3->4->5->1, then this will lead to misuse of computational resources. Instead of this, if we identify and isolate the graph's cycle during the journey of the initial ant, the succeeding ants follow another way, which results in faster solution convergence and saves the computational resources.

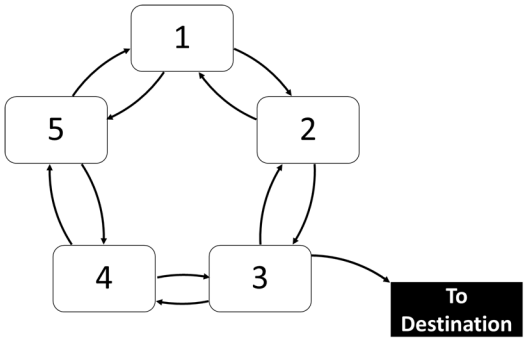


Figure 2 The path contains the cycle

2. It is not feasible to adjust the road network/graph each time while finding the shortest path to expel the path's cycle. So it is required to build up another system to identify the cycle in the graph and segregate it.

We have proposed a modification in the ant colony system's edge selection process to detect and eliminate the cycle in the graph as shown in algorithms 5.4 and 5.5.

The idea behind the algorithms is to detect the cycle during the edge selection process is as follows:

As per the graph theory, the cycle is a path $n_1, n_2 n_3 n_4 \dots \dots n_k$ with $K \geq 3$ such that $n_1 = n_k$ i.e., the cycle in any graph consists of a minimum of three nodes. So,

1. If the ant's trace contains equal or more than three nodes and
2. The next available node is already in the ants' trace

   Then there is a cycle from that node to the rest of the path.

Algorithm 5.4 Modification in the edge selection process of ant colony algorithm

Input ants, graph
Output Detection of the cycle and set pheromone to infinite for all/some of the edges available in the trace

```
1: procedure ModifyEdgeSelectionProcess
2:     while any of the ant complete its path i.e. endnode is in the trace do
3:         for Each Ant ← AntsList do
4:             for  Each AvailableNextNode ← AvailableNextNodesList do
5:                 if PheromoneValue (CurrentNode,AvailableNextNode) > 0 then
6:                     if TraceOfAnt(i) ≥ 3 then
7:                         if AvailableNextNode ∈ trace then
8:                             print("Cycle Detected")
9:                             call
10:                            procedure FunctionSetPhermononeToInifite(trace,graph) //for all or some of the edges available
   in the trace
```

The Ant colony algorithm executes for ten iterations, and the best path (in terms of minimum distance) is considered. The algorithm 5.4 (***ModifyEdgeSelectionProcess***) detects the cycle in

the ant's path/trace. And this trace is passed to our proposed procedure *FunctionSetPheromoneToInfinite depicted in algorithm 5.5..*

Algorithm 5.5 Proposed process to set pheromone for the cycles

Input trace (Where Cycle Detected), graph
Output set the pheromone matrix to infinite for the edges part of the cycle

```
1:  procedure FUNCTIONSETPHERMONONETOINIFITE
2:  getNodeid ← 0
3:  defaultPheromone ← 1
4:      for Each NodeId ← trace do
5:          if OutgoingEdges(NodeId) > 2 then
6:              getNodeid ← NodeId
7:      if getNodeid == 0 then
8:          for each x ← trace do
9:              pheromone[x][x + 1] ← -1
10:     else
11:         pheromone ← defaultPheromone
12:         for each SelectedNode ← trace do
13:             if SelectedNode == getNodeid then
14:                 pheromone ← −1
15:             pheromone[x][x + 1] ← pheromone
```

The procedure mentioned in algorithm 5.5 will set the pheromone value to infinite (i.e., the edges are isolated) for all or some of the edges based on the following rule:

If none of the nodes in the trace of ant having more than 2 *outgoing edges,*

then set the pheromone of all the edges between respective nodes to infinite.

Else

Set pheromone to infinite, from the node having more than two outgoing edges.

The results of this proposed work are discussed in section 6.3

(3) PROPOSED DYNAMIC APPROACH FOR ANT COLONY OPTIMIZATION [21]

During the actual implementation of ACO, the algorithm will execute at the beginning of the vehicle tour. Thus, it will take the current scenario (i.e., air pollution level, traffic volume) and suggest the best optimal path in decision making. However, the values of air pollution, traffic volume, etc., changes frequently with time. So, when the vehicle reaches the next junction, the scenario would get change. Thus the vehicle ends up with a heavy volume of traffic or an increase in effective air pollution. Therefore, we have proposed executing the ant colony algorithm before reaching the next junction and fetching the following optimal path to solve this issue.

Algorithm 5.6 Proposed DynamicACO

Input SourceNode,DestinationNode
Output Optimal Path

procedure DYNAMICACO(SOURCENODE,DESTINATIONNODE)
FinalTraceList[] ← SourceNode
    Get Ahmedabad.net.xml using SUMO script osmWebWizard.py
    graph ← *NetwrokToGraph*(*Ahmedabad.net.xml*)
    AdjacencyMatrix ← *CreateAdjacencyMatrix*(*graph*)
    do
        Subscribe if the vehicle reaches near to the SourceNode
        if Intrupt Received Due To The Subscription **then**
            traceList ← ExecuteACO(SourceNode,DestinationNode)
            // TraceList consists of SourceNode,DestinationNode, and series of the intermediate nodes that leads to the destination
            FinalTraceList← traceList[1]
            SourceNode ← traceList[1]
    while *SourceNode* ≠DestinationNode
    Return FinalTraceList // Collection of nodes which is part of optimal path

procedure CREATEADJACENCYMATRIX(GRAPH)
AdjacencyMatrix[]
    for each *edge* ← *GRAPH'sEdgeList* **do**
        AdjacencyMatrix.addVertex(edge.getFromNode().getID())
        AdjacencyMatrix.addVertex(edge.getToNode().getID())
        AdjacencyMatrix.addEdge(edge.getFromNode().getID(),graph.edge.getToNode().getID(),edge.getLength())
    Return AdjacencyMatrix

Algorithm 5.7 ExecuteACO

Input SourceNode,DestinationNode,AdjacencyMatrix,graph
Output Optimal Path

procedure EXECUTEACO(SOURCENODE,DESTINATIONNODE,ADJACENCYMATRIX,GRAPH)
EvaporationRate ← 0.1
threshhold ← 0.5
alpha ← 1.5
beta ← 2.0
size ← length(Graph's EdgeList)
NumberOfAnt ← size/2
    if Path exsists between graph.nodes **then**
        //size of following matrix is size X size
        distancematrix ← NormalizedLength(graph.edges)
        pheromonematrix ← 1.0;
        pollutionmatrix ← Normalized(Random AQI values)
    else
        distancematrix, *pheromonematrix*, *pollutionmatrix* ← -1
    Initialize all ants at SourceNode with distancecost = 0, pollutioncost=0, selectioncost=0 , trace= [SourceNode]
    while any of the ant complete its path i.e. endnode is in the trace **do**
        for *ant* ← 0, *NumberOfAnt* **do**
            for *edge* ← *graph.edges* **do**
                Find the most probable edge to select
                out of the available alternate edges to reach at the destination.
                trace[] ← EndNode(edge)
    AntCompleted ← TotalNumberOfAntsReachAtDestination()
    Calculate *AverageCost*/ = *AntCompleted* where AverageCost is the average cost considering distancecost and pollutioncost
    Update pheromone *pheromonematrix*[*i*][*j*]∗ = (1 − *EvaporationRate*)
    if *pheromonematrix*[*i*][*j*] < *threshhold* **then** pheromonematrix[i][j] ← -1

Based on the Ant Colony algorithm's background, we have suggested a dynamic ant colony algorithm where the decision to select the route from the available set of routes executes at each junction. Algorithm 5.6 and Algorithm 5.7 describe the proposed algorithm.

The results of this algorithm are discussed in section 6.4

(4) PROPOSED MODEL TO PREDICT AIR QUALITY INDEX USING MACHINE LEARNING TECHNIQUE [22]

The novel part of this study is based on the fact that a different part of the city has a distinct air quality index level due to the variation of the meteorological data (temperature, wind speed, direction, air humidity etc.). In this scenario, it becomes essential to know the AQI of the place to visit and the intermediate locations instead of learning the AQI of the whole city. The significant issue with this investigation is the accessibility of the dataset for every crossway of the town. We have created the estimation information for the air quality index (AQI) for the Ahmedabad city from a similar approach as the dataset produced for Aarhus city, Denmark [23] [24] to predict the air quality index for the intersections, utilizing Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM).

Finally, we examine & compare the correctness of different configurations using root mean square error in predicting the actual vs. predicted AQI using the proposed configuration. As shown in section 6.5, our experiment results show that the LSTM predicts AQI accurately than the ANN for the given dataset. This study is a critical inspiration for inquiring into urban air quality and helping the administration design.

DATASET AND DATA PREPROCESSING:

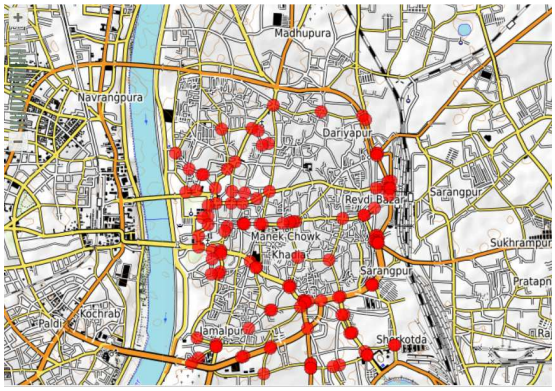


Figure 3 Ahmedabad city map: Case study Area and relevant road intersections [Downloaded from https://www.gpsvisualizer.com/

For the experiment, we have considered a map that is part of the Ahmedabad city, as shown in figure 3. The map under consideration is converted to Graph/Network using SumoLib. And the derived Network file contains 145 nodes, 300 edges. Dataset contains [JunctionId, ObservationDate Time, AQI] where the data type of each values are [int64, datetime64, int64] respectively. The total number of rows in the dataset is 2547302.

The pollutant concentration is observed differently at different junctions at different times of the day. As studied in the article [25] and [26], there is a pattern in the changing of AQI for the particular junction at a given duration, i.e. {JunctionId, Day, Time, Month}. The pattern occurs due to many parameters, like office hour rush, evening picnic spot, a dense area like the railway station, or bus stand at the junction. This argument drives us to conclude that the AQI depends on Day, Month, Time Of day, and the city's location. Thus ANN has an input layer { Day, Month, Time Of the day, JunctionID} to predict AQI. While in the case of LSTM, the index column considered is {DateTime} to make the problem a time-series prediction problem. The dataset is for LSTM is available as the individual files, which consist of a reading from each junction.

The data for the collection of pollution measurements designed to complement the vehicle traffic dataset available from the smart city pulse project [24]. The pollution is directly proportional to the traffic level and industrial area, if any. Due to the unavailability of the pollutant sensor, synthesized pollution data for Aarhus, Denmark, is derived through the traffic dataset.

To derive the pollution dataset of the Ahmedabad city, Gujarat, India, we have used the same approach followed in the "city pulse" project (Aarhus, Denmark). The dataset contains Junction Id (mapped from latitude/longitude of the respective junction) and measurement of the individual pollutant {ozone, particullate_matter, carbon_monoxide, sulfure_dioxide, nitrogen_dioxide}. For this individual pollutant, the sub-index is required to be calculated to identify the understandable form of the air quality index (Now referred to as AQI). AQI for the respective junction calculated using:

$aqi = Max\ (I1, I2, I3 \dots In)$ where the sub-indexes are computed from the observed values of individual pollutants [27].

For individual pollutants, sub-indexes will be calculated individually. By and large, AQI is determined just if the information is accessible for at least three contaminations, out of which one ought to be either PM2.5 or PM10 fundamentally. Else, data is viewed as inadequate for figuring AQI [27].

MODEL USED:

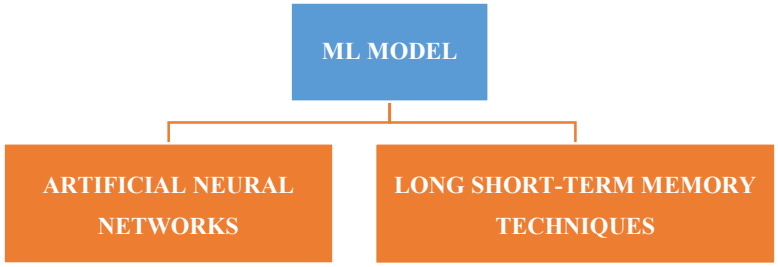


Figure 4: Model used

In the proposed approach we have used artificial neural network (ANN) and long short-term memory techniques (LSTM).

CONFIGURATION OF MODEL:

In the implementation part, firstly, we have implemented ANN and LSTM. Then the comparison of both is made. We have inferred that LSTM works better with improved RMSE and R-Square scores over ANN. So we have picked the LSTM for additional investigations. Finally, we have designed different hyperparameters for the LSTM to show signs of improvement results.

Several hyperparameters have been evaluated before building the LSTM and ANN model, including several layers, neurons, activation functions, and optimizers. Then, the effect of each parameter has been examined and has been finalized as the configuration mentioned in Tables 1 and 2.

Table 1 Configuration parameter for LSTM

Neurons in the first Hidden Layer	50
Dropout	0.3
Output Layer	kernel_initializer='normal' activation='sigmoid'
Optimizer	Stochastic gradient descent
Loss function	Mean absolute error

Table 2 Configuration parameter for Artificial Neural Network

Model	Sequential
Optimizer	Stochastic gradient descent
Loss function	Mean absolute error
Input Layer : 3 Neurons First Hidden layer: 128 Neurons Hidden Layers with 256 Neurons Output Layer: 1 Neuron	
kernel_initializer='normal' activation= relu’	

TRAINING AND TESTING:

The training and prediction procedure is described in Figures 5 and 6.

Figures 5 & 6 depict the working of the LSTM and ANN model as above. Where i={1 to 145} i.e., 145 Junctions of Ahmedabad cities are in consideration.

Based on the model described in the flow chart given in figure 5 and 6, we have split the dataset into two-part:

Training data shape X, y => (14400, 1, 3) (14400,)

Testing data shape X, y => (3158, 1, 3) (3158,)

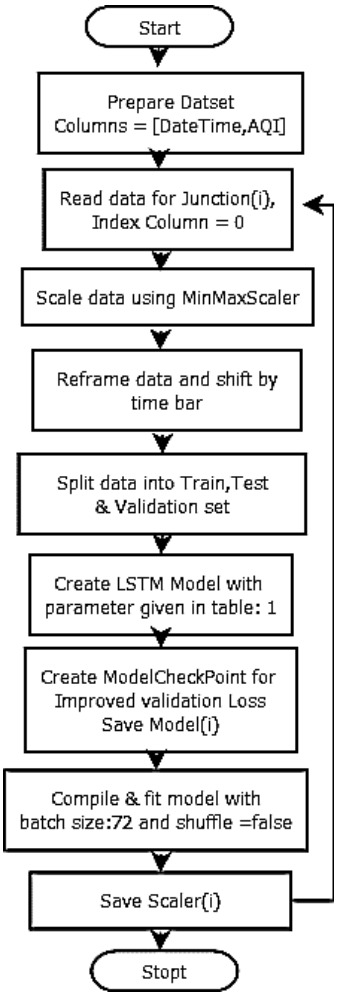
Each junction contains 17558 measurements for the regular intervals of 5 minutes for three months.

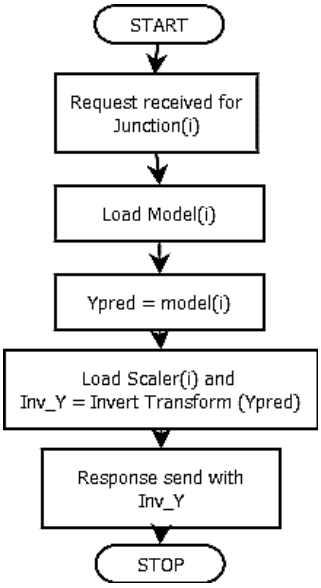
Our proposed system assumes that the sensor to measure air pollutants lies at every 145 crossways. These sensors measure the pollutant level and send it to the server dataset. Our model is trained to consider these updated values, periodically.

(1) Training Phase

   a. LSTM: Figure 5 (a) represents the LSTM training model used to predict (figure 5 (b)) the AQI level for all the 145 crossways.
   b. ANN: Figure 6(a) represents the ANN training model used to predict (figure 5 (b)) the AQI for all the junctions.

(2) Prediction Phase for ANN & LSTM: fig. 5 (b) depicted the prediction phase’s flow for ANN & LSTM.





(a) (b)

Figure 5. LSTM Model Train and Test For Junction(i)

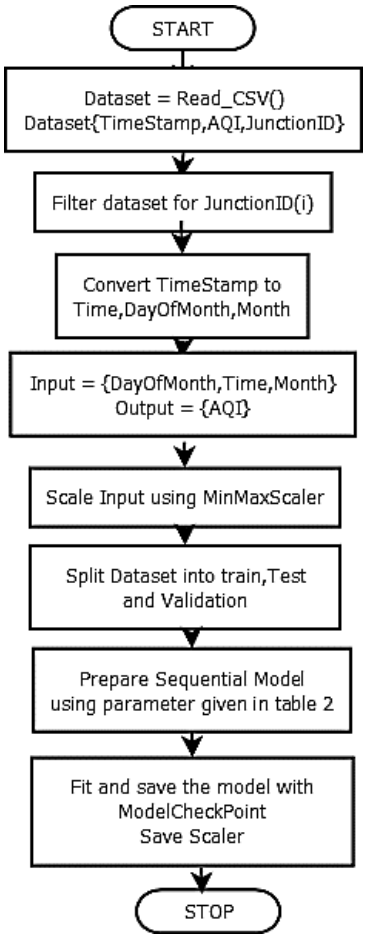


Figure 6. Neural Network Training for Junction(i) & Testing will be the same as mentioned in 5 (b)

As shown in figure 7, we acquire the request from the client by input the parameter:

a. In case of LSTM: {Junction_ID, Time t for the prediction}, Here, the Junction_Id represents the query for the crossway out of the 145 crossways. Time t is the time interval to predict what might be the AQI following t minutes.
b. In case of ANN: {DayofMonth, Time, Month, JunctionId}: DayofMonth ranges from 1 to 30 /31 /28 /29. Time ranges from 00:00 hour to 23:59 Hours with a period of 5 minutes.

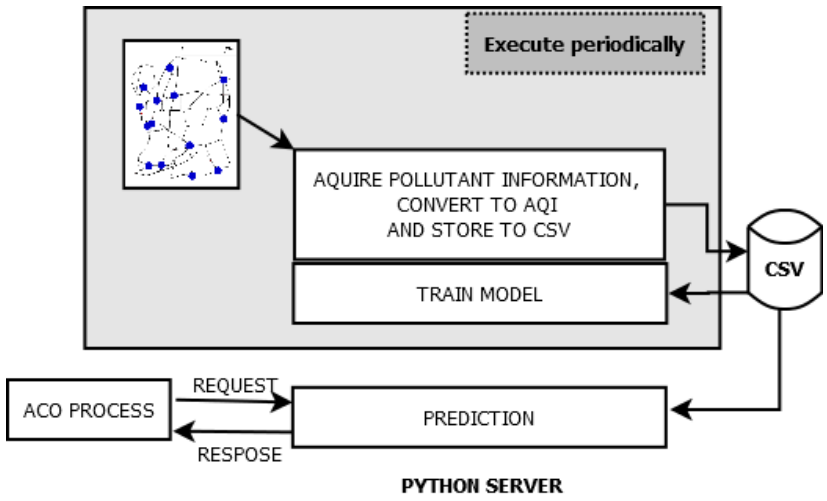


Figure 7 : Client-Server Model configuration to get predicted results for the users

The output of both the models is the AQI. Then the server joins the resultant information as a JSON response and sends back to the Selection block of ACO Process which is described in proposed architecture (figure 7).

For the given dataset, the LSTM is performed better than the ANN. Thus LSTM is the best choice for further experiments.

LSTM has experimented with a different optimizer for the variation in the number of neurons in the first hidden layer, as shown in table 3.

Table 3 represents different configurations for the LSTM and respective RMSE:

Table 3 RMSE analysis of LSTM Optimizers with different neurons

Optimizer	Neurons in the first hidden layer	Average RMSE
SGD	50	9.891
	100	9.236
	150	8.796
	200	8.684
	300	8.595
Adam	50	11.263
	100	11.609
	150	11.689
	200	11.830
	300	11.848

As per Table 3, The SGD has improved RMSE over Adam for different configurations of LSTM. Thus we are considering the SGD optimizer for our experiment. To finalize the number of neurons in the first hidden layer, we have experimented with 50, 100,150,200,300 neurons. Results show that the increasing neurons in the first hidden layer do not improve RMSE significantly. So we use 50 neurons in the first hidden layer. In conclusion, we use an SGD optimizer with 50 neurons in the first hidden layer for the LSTM configuration.

The result for the comparison of ANN and LSTM is discussed in section 6.5.

(5) PROPOSED ANT COLONY BASED ARCHITECTURE FOR AIR POLLUTION AWARE VEHICLE ROUTING IN SMART CITIES USING MACHINE LEARNING

The proposed developed framework is designed to suggest vehicles to move from source to destination, such as riders of the vehicles have the minor effect of air pollutants. This framework is an ant colony optimization algorithm-based architecture supported with machine learning techniques. The architecture is depicted in figure 8 and aims to reduce the effects of air pollution on the human body during traveling through the following building blocks:

1. Client Module
2. Server Module
    i. Adjacency matrix generation
    ii. Ant colony process
        a. Initialization
        b. Finding optimal path
        c. Optimal path suggestion
    iii. Sensor value acquisition and conversion to Air Quality Index
    iv. Recursive Machine learning model creation\ updating block
    v. Prediction block

This framework is the novel ant colony-based architecture for air pollution-aware vehicle routing in smart cities using machine learning. The client module is developed on the Tkinter toolkit shipped with python. The adjacency matrix generator is the block to convert the selected portion of OpenStreetMap [28] to graph using sumoLib [29] and then to adjacency matrix on which the ant colony optimization technique operates. The Ant colony process consists of the initialization phase, finding the optimal path, and optimal path suggestion.

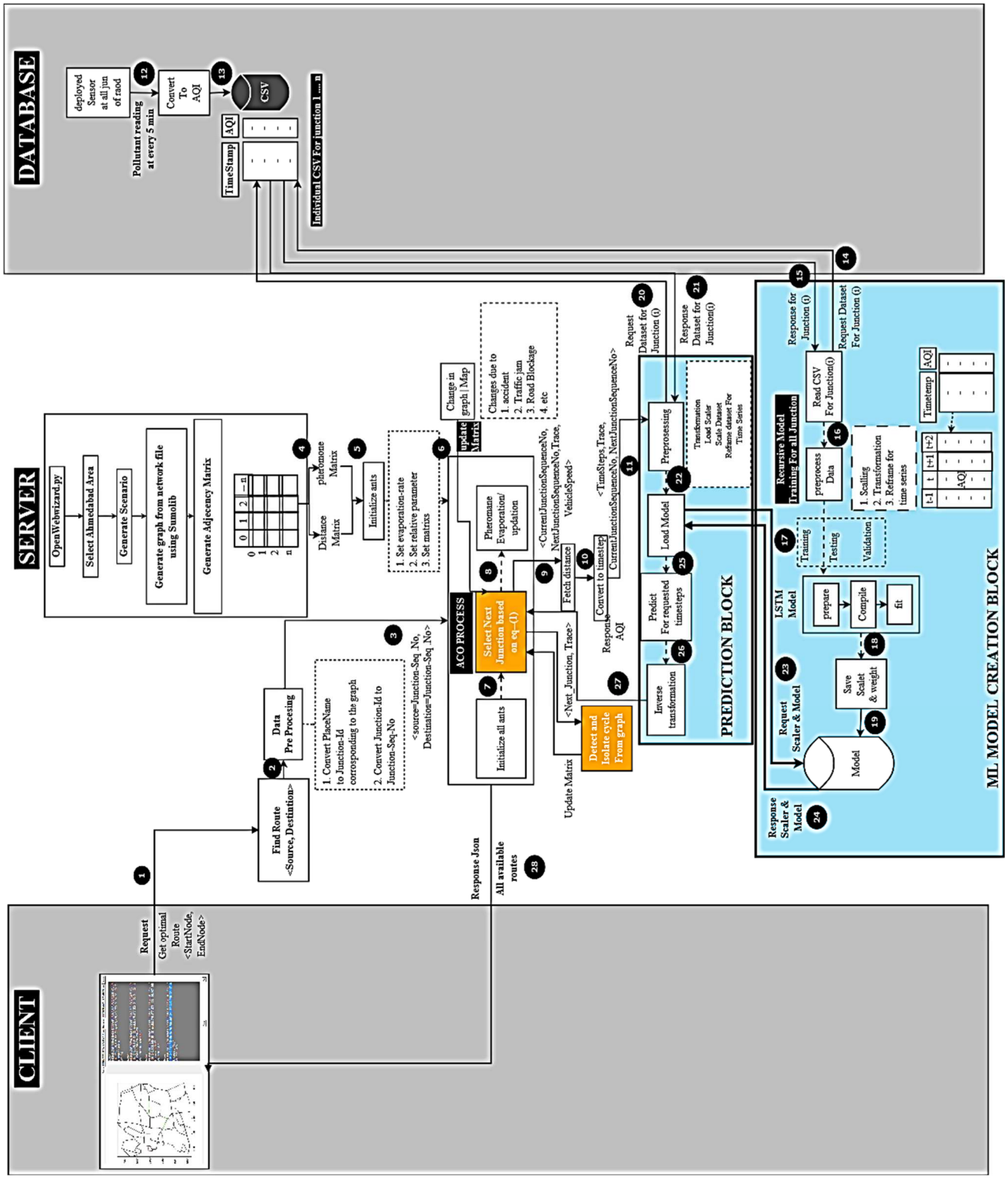


Figure 8. Proposed architecture

The Initialization phase initializes important parameters, Distance matrix, pheromone matrix, and assigns previously created adjacency matrix. In finding optimal path phase, the information initialized in the previous step is used and based on this information, probability selection is calculated and optimal path is evaluated. The optimal path suggestion phase suggests the optimal available route from the previous stage and updates the pheromones. The sensor value acquisition block acquires the pollutant level from the sensor deployed at each junction across the city and gives it to the conversion block. This conversion block converts the available pollutant values to the Air Quality Index (AQI) using the formulas given in [27].

The flow of the proposed architecture is as follows

1. The client sends the request to the server to get optimal routes to the destination, passing the argument <StartNode, EndNode>. Where StartNode is the source node, and EndNode is the destination node.
2. The request FindRoute <Source, Destination> is received at the server, and the received arguments will be preprocessed.
    a) The <Source, Destination> are in the form of the place's name, i.e., Bank Of Baroda, Satellite Area. This place-name needs to map to the associated Junction_id as mentioned in the graph**.**
    b) But the ant colony algorithm is executing on the Adjacency matrix. The Adjacency matrix contains sequence numbers for the rows and columns called Junction_Seq_No. Thus the Junction_id has to be converted to the sequence number**.**

    Therefore, during the preprocessing, conversion (a) and (b) must be performed.
3. Now, this Source_Junction_Seq_No and Destination_Junction_Seq_No is passed to the ACO_PROCESS's Initialization phase.
4. The Adjacency matrix is used to generate Distance_Matrix and Pheromone_Matrix.
5. Evaporation_Rate and Relative parameters ($\propto$,β,γ) are prepared.
6. The parameter prepared in steps 4 and 5 are passed to initialize the ACO_PROCESS.
7. All ants will be initialized with Source_Junction_Seq_No as starting node and Destination_Junction_Seq_No as the destination node.
8. All ants will select the next_node using probability equation (1). After evaluating the optimal path, the pheromone evaporation process takes place.

9. During the selection of the next junction, it is required to fetch the distance. To get the distance from the source node (starting node of the journey) to the next available nodes, we need to pass the

   $< CurrentJunctionSequenceNo, NextJunctionSequenceNo, Trace, VehicleSpeed >.$

   Where

- $Trace$ is the list of $Junction\_Sequence\_No$, which ant has visited, including $CurrentJunctionSequenceNo$.
- $CurrentJunctionSequenceNo$ is the current position of the ant.
- $NextJunctionSequenceNo$ is the node available for the selection as the next node.

Now, the distance between the starting node and the Next_Junction can be obtained using the following function (algorithm 5.8):

Algorithm 5.8 Get distance

```
1: procedure GETDISTANCE(Trace,CurrentJunctionSequenceNo, NextJunctionSequenceNo)
2:     //Initialize distance with the distance between CurrentJunctionSequenceNo and
3:     //NextJunctionSequenceNo
4:     distance ← DistanceBetween(CurrentJunctionSequenceNo, NextJunctionSequenceNo)
5:     for Each i ← Trace do
6:         distance ← distance + DistanceBetween(Trace[i]][Trace[i+1])
7: return distance
```

10. During the selection of the next junction using probability equation (1), the equation contains a pollution parameter.
    a. Our proposed architecture considers predicting the pollution value for the next junction from the starting node (source node) of the journey. We need the actual time to reach the next node from the source node to predict the pollution value.

       The prediction of the pollution value for the next junction will work as follows:

       i. Get time t to reach the next junction after distance d from the source node, and vehicle speed s is

$$time\ (t) = \frac{d}{s}$$

       ii. Now our dataset is spared by five minutes. Therefore we need to convert the time (t) to the time steps using the following function (algorithm 5.9) :

Algorithm 5.9 Convert time to time steps

```
procedure GETTIMESTAMP(distance)
    //Distance is in normalised form ie. convert to actual distance
    distance ← distance * MaxDistance
    //here we have considered average speed of vehicle is 28
    time ← distance/PresumedAverageSpeed
    //12 because data set is sampled at each 5 minutes. and above time is in km per hour. so 60 minutes/5 sample time = 12
    TimeSteps ← time/12
return TimeSteps
```

Note: The paper [30] has shown that the average speed of Ahmedabad city's vehicle is around 25 KMPH to 30 KMPH. Thus we have considered the average speed of vehicle as 28 KMPH.

11. <$TimeSteps, Trace, CurrentJunctionSequenceNo, NextJunctionSequenceNo$> is given to the **PREDICTION BLOCK** to get predicted pollution parameter for the respective junction at given $TimeSteps$. Step 12 and 13 are used to prepare the dataset (CSV) for the individual junction.
12. The pollution sensors are deployed at each junction, and sensor values are acquired every 5 minutes.
13. These values will be converted to Air Quality Index. The format to store information into the dataset is: <Timestamp, Air Quality Index >. The dataset/CSV for individual junction are stored separately.

    Step 14 to 19 are used to generate scaler and weight of our configured LSTM model, which will be used during **LOAD MODEL** phase of **PREDICTION BLOCK.**
14. In **ML MODEL CREATION BLOCK,** the dataset is required for the recursive/periodic training to create the model for the individual junction. The request for the dataset is made to the database server.
15. In response to step 14, the dataset for the junction (i) is available to **ML MODEL CREATION BLOCK.**
16. LSTM model will learn a function that maps a sequence of past observations as input to predict the following sequence number as output. Then, the received series of observations is passed through preprocessing. During the preprocessing following operation carried out :
    i. Scaling
    ii. Transformation
    iii. Reframing of data for time-series

17. To predict the future AQI, it needs to split the data to train models. Therefore, the preprocessed data divided into training, validation, and testing datasets.
18. In turn, the dataset obtained from step 17 fit through our proposed LSTM model.
19. After fitting the LSTM model, the final weight obtained through ModelCheckPoint and scaler obtained through MinMaxScaler, saved to the database for the junction (i).

    Steps 20 to 27 were used to generate the predicted AQI value for the requested junction at the requested timesteps.
20. Request dataset for the junction (i) for the processing of **PREDICTION BLOCK**
21. Response dataset for the junction (i) for the processing of **PREDICTION BLOCK**
22. The response received in step 21 is preprocessed and given to **LOAD MODEL** phase.
23. Now, the **LOAD MODEL** phase of the **PREDICTION BLOCK** initiates the request to load the saved weight and scaler for the junction for which prediction is required to be made (obtained using steps 14 to 19).
24. The response will be the weight and scaler for the requisite junction.
25. The loaded model is used for the prediction of the given timesteps and junction.
26. The predicted value is given for the inverse transformation.
27. Inverse transformed output of the **PREDICTION BLOCK** is the predicted AQI value for the requested junction for the requested timestep. Based on this AQI, ACO_PROCESS decides to select the next junction.
28. All available routes will be sent to the client in JSON format. The response includes following:

Route	Distance	Air Quality Index

During selection of the next junction based on equation (1) in the ACO PROCESS,

- The algorithm 5.4 and 5.5 mentioned in section 5 is used to detect and isolate the cycle from the graph/map.
- If there is a change in graph/map, the matrix mentioned in step 4 will be updated to reflect the changes due to accidents, traffic jams, road blockage, etc. This scenario is simulated using Tkinter.

6. METHODOLOGY OF RESEARCH, RESULTS / COMPARISONS

6.1 RESULTS to EVALUATE THE TRANSITION RULE PARAMETER ∝ AND THE EVAPORATION RATE ρ:

Alpha	*Frequency*
1	27
1.5	238
2	17
3	10
4	8
5	15
More	0
Total	315

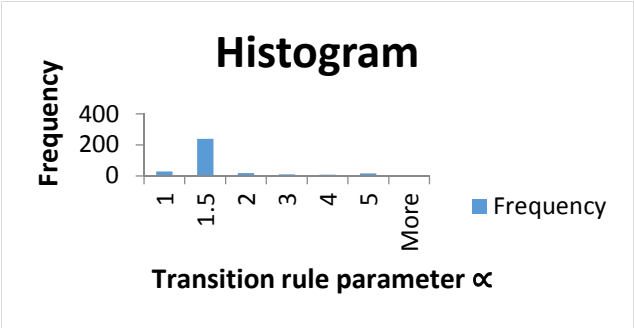


Figure 9 Histogram for Transition rule parameter ∝

Evap_rate	*Frequency*
0.1	159
0.2	32
0.3	34
0.4	20
0.5	13
0.6	23
0.7	17
0.8	17
More	0
Total	315

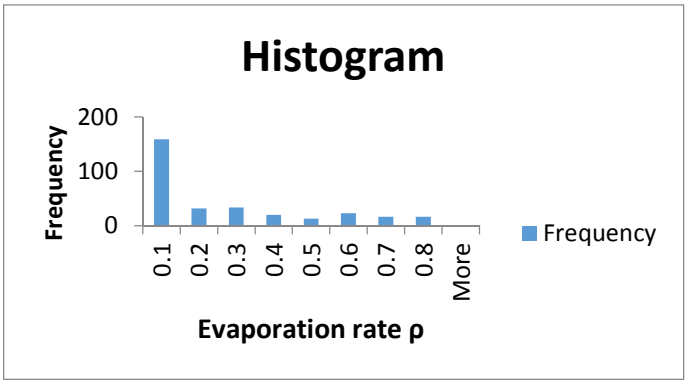


Figure 10 Histogram for Evaporation rate ρ

Table 4 Evaluation of parameters

	∝_ρ							
	1_0.2	**……**	**1.5_0.1**	**1.5_0.2**	**…..**	**5_0.7**	**5_0.8**	
Journey ID	**Distance**							**#**
1	54.72911		65.95629	55.63925		54.79885	56.08853	4_0.8
2	52.44444		43.23238	44.06533		47.26429	43.16264	5_0.8
3	50.06808		50.97078	56.25481		54.65244	49.64809	1.5_0.6
4	29.83227		29.38682	29.83227		28.15622	29.38682	1_0.5
5	32.37563		31.88341	39.90522		36.9577	48.55232	1.5_0.4
6	29.66061		29.66061	29.66061		45.48984	43.74115	1.5_0.4
7	33.98452		34.49795	33.98452		33.13853	33.13853	1.5_0.3
.								
315	64.4952		58.6081	51.5966		56.9904	55.6114	5_0.5

Value of ∝_ρ for the minimum distance of the respective journey

Execution of algorithm 5.1 produces the rows as a trace/output shown in table 4.

From table 4, we have identified the histogram for transition rule parameter ∝ and evaporation rate ρ as shown in figure 9 and figure 10, respectively.

From Figures 9 and 10, we have concluded that the optimum results for the shortest path can be obtained using our implementation of the ant colony algorithm, through considering transition rule parameter ∝ to value 1.5 and the evaporation rate ρ to value 0.1.

6.2 RESULTS FOR ANALYSIS OF EFFECT OF ∝ AND ρ ON THE PERFORMANCE EVALUATION PARAMETERS

As shown in figure (figure 11(a)) and (figure 11(b)), the transition rule parameter ∝=1.5 gives minimum average distance and diffusion. Thus we can conclude that ∝=1.5 gives optimum distance with faster solution convergence. By observing the value of ρ, it is concluded that there is no significant impact of values of ρ on the average length of the constructed path (figure 11(c)). But from (figure 11(d)), the average value of the diffusion increase to double with the increase in the value of ρ.

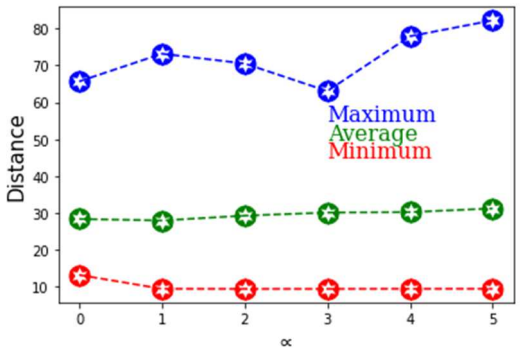


Figure 11 (a) The Transition rule parameter ∝ Vs. Distance(in meter)

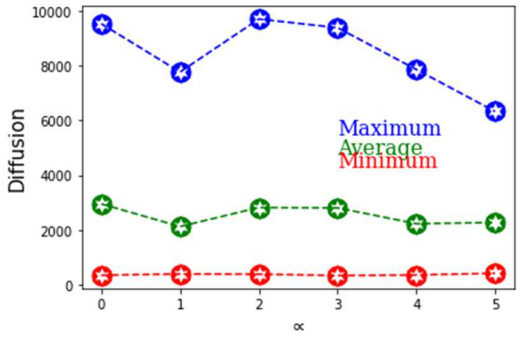


Figure 11 (b) The Transition rule parameter ∝ Vs. Diffusion (count)

Distance	1	**1.5**	2	3	4	5
MIN	13.19368	**9.445801**	9.420347	9.420347	9.445801	9.445801
MAX	65.68695	**73.19637**	70.54499	63.07148	77.93076	82.24898
AVERAGE	28.38128	**27.96989**	29.28301	30.12057	30.24213	31.2339

Diffusion	1	1.5	2	3	4	5
MIN	333	381	371	324	343	406
MAX	9549	7779	9719	9403	7872	6334
AVERAGE	2946.583	2118.8	2813.45	2801.783	2221.65	2266.1

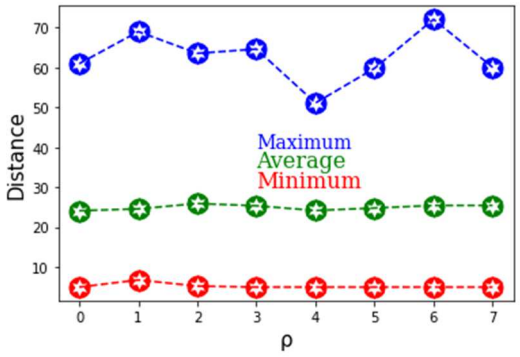


Figure 11 (c) The Evaporation rate ρ Vs. Distance(in meters)

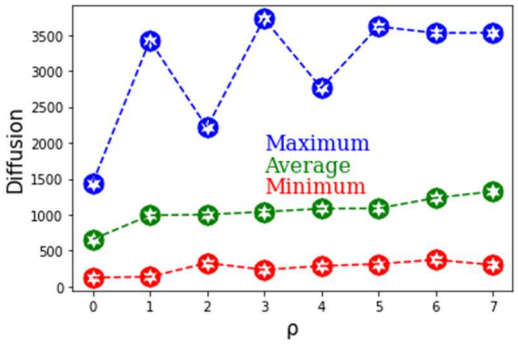


Figure 11 (d) The Evaporation rate ρ Vs. Diffusion (Count)

Distance	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
MIN	4.973304	6.759244	5.221638	4.973304	4.973304	4.973304	4.973304	4.973304
MAX	60.96436	68.99911	63.55107	64.62077	51.11254	59.79885	72.08301	60.03549
AVG	24.1098	24.57365	25.91632	25.34699	24.15822	24.78652	25.42835	25.44209

Diffusion	**0.1**	0.2	0.3	0.4	0.5	0.6	0.7	0.8
MIN	**127**	137	329	236	287	316	374	303
MAX	**1444**	3427	2213	3728	2763	3617	3528	3531
AVERAGE	**658.57**	991.78	1002.5	1040.9	1083.9	1087.42	1233.04	1323.54

From these experiments, we have concluded that the optimum results for the shortest path can be obtained using our implementation of the ant colony algorithm, considering transition rule parameter ∝ to value 1.5 and the evaporation rate ρ to value 0.1.

6.3 RESULTS FOR PROPOSED ALGORITHM FOR ANT STUCK

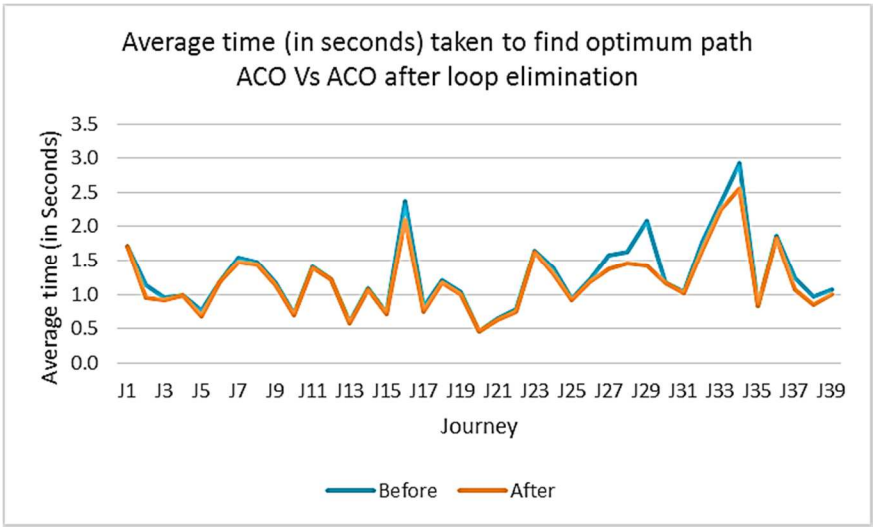


Figure 12 Average time t (in Seconds) ants spent to find the solution for all the 39 journey

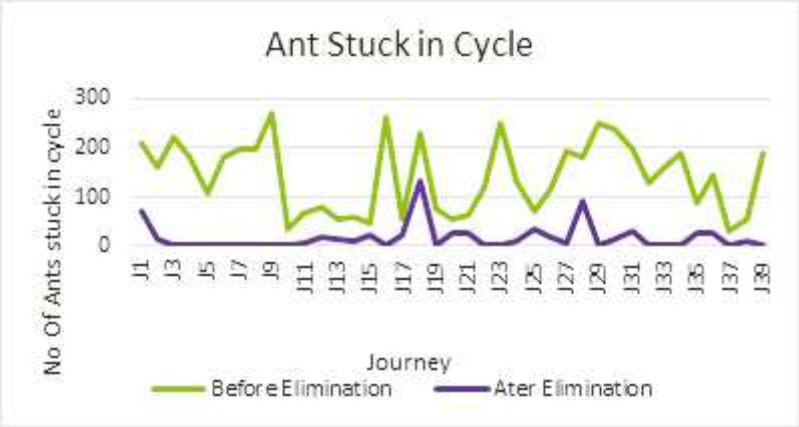


Figure 13 Total number of ants stuck in the cycle before and after eliminating the cycle for all the journeys.

Figure 12 shows the average time t (In seconds) ants spent to find the solution in the journey Ji. After modification in the selection process, it is observed that the average time spent by each ant gets reduced by 5.26% to solve the problem.

As shown in figure 13, the proposed algorithm reduces the ant stuck in the cycle of the graph. Due to the implementation of this algorithm, the ants contribute more to generate optimum solutions instead of getting stuck in the cycle. As per the experimental statistics, the proposed algorithm reduces 85.4677% of ants to get stuck in the cycle.

Figure 14 shows the number of cycles detected in the graph before and after the execution of the proposed algorithm. The proposed algorithm detects almost 76 % of cycles in the graph.

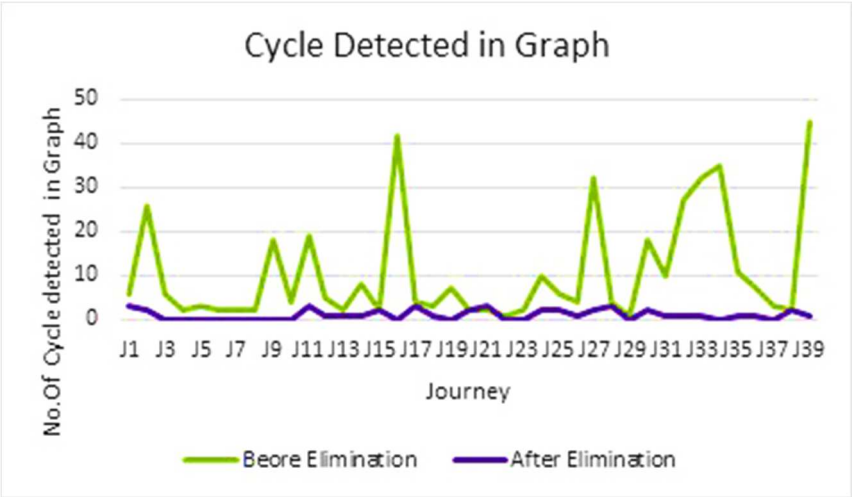


Figure 14 Number of Cycle detected in the graph for all the journey

So we can conclude that the problem of ants being stuck in the cycle during the decision-making process of an ant colony system has been identified. Our proposed solution detects 76% cycles for the experimented journeys and, if eliminated, will result in 5.26% less time taken to solve the problem over the existing Ant Colony system. Furthermore, the proposed calculation reduces the ant to be stuck in the cycle of the diagram; this will let the ants contribute more to generate optimum solutions rather than stall out in the cycle. According to the exploratory insights, the proposed calculation decreases 85.4677% of ants stuck in the cycle and improves computational efficiency.

6.4 RESULTS OF THE PROPOSED METHODOLOGY TO IMPLEMENT A DYNAMIC ANT COLONY ALGORITHM

We have executed this experiment for the procedure shown in algorithm 5.6 for 350 different journeys and find out the results for all the said journeys. To explain the result in better way, we have elaborated the trace obtained for the single source-destination pair and shown in table 5.

Note: *All the values mentioned in Table 5 and Table 6 are the normalized values against the maximum value with four decimal points.*

As presented in algorithm 5.6 mentioned in section 5 (3) , selecting the next node out of the available nodes is done just before reaching the respective junction. Thus iterations are repeated for each junction until the vehicle arrives at its destination. Table 5 shows the ITERATION [1,2…N]. The result of each ITERATION represents node to node cost as well as the total cost. The row of each iteration (start node, end node) represents the distance and pollution cost of that particular road segment.

Suppose we execute the ant colony algorithm before beginning the journey; we will get results as shown in iteration 1 (table 5). As described in our proposed approach, ACO executes at each junction; thus, we need to accumulate the distance cost & pollution cost of the actual edge/road segment that the vehicle has traversed. Thus these statistics are combined and represented in Table 6 and calculated total cost for the actual traversal of the road segment.

Figure 15 is the graphical representation of Tables 5 and 6. Iteration 1 in Table 5 is the representation of the basic ACO algorithm. At the same time, the result of dynamic ACO is shown in table 6. The cumulative cost of pollution at each link is comparatively lower than iteration 1(ACO), i.e., it proves that dynamic decision making (Dynamic ACO), reduces the

impact of air pollution [0.85475 as compared to 0.9901 from iteration 1, Table 5] on the health by slightly increasing in the cost of the distance [1.63163 as compared to 1.5378 from iteration 1, Table 5].

Table 5 Link to link Iterative costs for Distance and Pollution

Iteration : 1 [source:0 to destination:3]

	Start node	End node	Distance Cost	Pollution cost
C1	0	1	0.4120	0.3124
	1	2	0.3808	0.0098
	2	3	0.7450	0.6679

Route	Total Distance Cost	Total Pollution cost
[0, 1, 2, 3]	1.5378	0.9901

Iteration : 2 [source:1 to destination:3]

	Start node	End node	Distance Cost	Pollution cost
C2	1	2	0.3808	0.0001
	2	4	0.3666	0.5023
	4	8	0.6020	0.5661
	8	3	0.5220	0.0000

Route	Total Distance Cost	Total Pollution cost
[1, 2, 4, 8, 3]	1.8713	1.0685

Iteration : 3 [source:2 to destination:3]

	Start node	End node	Distance Cost	Pollution cost
C3	2	4	0.3666	0.5423
	4	7	0.5331	0.4812
	7	8	0.3717	1.0000
	8	3	0.5220	0.0001

Route	Total Distance Cost	Total Pollution cost
[2, 4, 7, 8, 3]	1.7934	2.0235

Iteration : 4 [source:4 to destination:3]

	Start node	End node	Distance Cost	Pollution cost
C4	4	3	0.4723	0.00003133

Route	Total Distance Cost	Total Pollution cost
[4, 3]	0.4723	0.00003133

Table 6 Cumulative costs: calculation for the partial links

Partial cost of links		Total Distance Cost	Total Pollution cost
C1	[0,1]	0.41201	0.31237
C2	[1,2]	0.38077	0.00007
C3	[2,4]	0.36657	0.54228
C3	[4,3]	0.47228	0.00003
[0,1,2,4,3]	Total	1.63163	0.85475

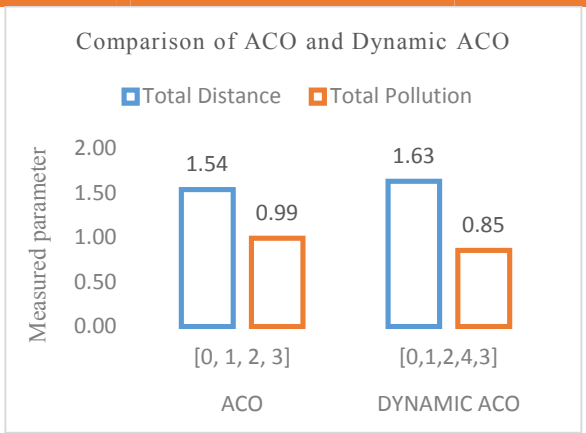


Figure 15 Comparison of ACO and proposed Dynamic ACO for a single journey

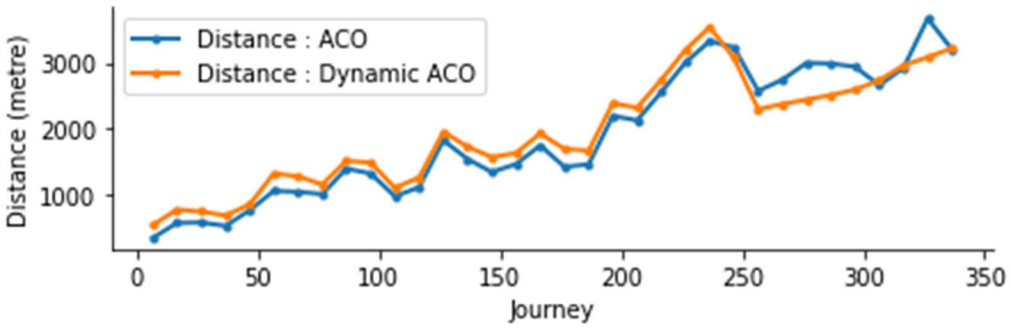


Figure (a) Journey Vs Distance (in meters)

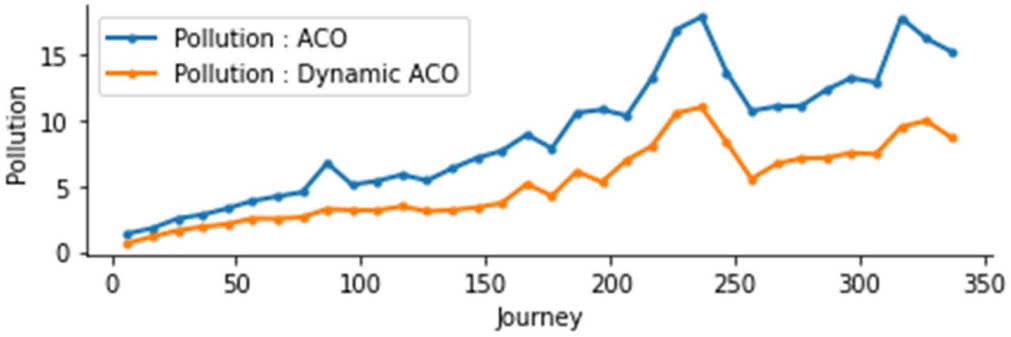


Figure (b) Journey Vs Pollution

Figure 16 Comparison of ACO and proposed Dynamic ACO

The same experiment is repeated for 350 trips, and the result is obtained, as shown in figure 16. Figure 16 (a) compares ACO and Dynamic ACO for the parameter obtained for the length of the 350 journeys. And Figure 16 (b) shows the comparison of ACO and Dynamic ACO for the parameter obtained total pollution affected the human body throughout the journeys.

Concerning figure 16 and Table 6, we conclude that the cumulative cost of pollution at each link is comparatively lower than iteration 1. Moreover, it proves that dynamic decision-making on the route at each junction reduces air pollution impact on the health by slightly increasing or no change in the path length for the 77.14 % of the journeys. The average percentage increment in distance due to the Dynamic ACO is 16.89 %, while the impact of pollution is reduced by 41.74 %.

6.5 RESULTS AND DISCUSSION FOR COMPARATIVE ANALYSIS OF ARTIFICIAL NEURAL NETWORKS AND LONG SHORT-TERM MEMORY TECHNIQUES FOR PREDICTING AIR QUALITY INDEX

This section compares the results of ANN & LSTM and shows that LSTM works better with the tuned hyperparameters to predict AQI. After deciding the best system design for the current

prediction task shown in the previous section, the training set is used to evaluate the LSTM and ANN model convergence. The evaluation of both the model is done based on the root mean square error and R-SQUARE.

First, we evaluated the ANN and then LSTM.

During the experiment, we have mentioned using EarlyStopping and ModelCheckpoint, where EarlyStopiing stops training when a monitored metric has stopped improving. For both ANN and LSTM models, it is observed that convergence saturation reached below 100 epochs. Due to this, the training phase does not reach 100 epochs. This is why it is not required to extend the epoch above 100. Otherwise, it will end up with an Overfitted model. During model fit ModelCheckpoint spare a model or loads at some stretch, so the model or loads can be stacked later to proceed with the preparation from the state saved.

Considering this fact, Figure 17 shows that the epoch is required to train the LSTM and ANN models. It is observed that the average epoch required for the LSTM is 91.56551724, and for the ANN, it is 96.89655172. From these statistics, we can conclude that LSTM converges with fewer epochs than ANN.

The dataset is divided into three sets in machine learning: training set, testing set, and validation set. The validation loss is calculated for each epoch and infers how much our prediction differs from the actual values. Figure 18 shows the validation loss per junction. The average validation loss for the ANN is 0.533562069.

In comparison with ANN, the average validation loss for the LSTM is 0.29661331. The LSTM results in comparatively lower validation loss than the ANN, resulting in better prediction of the AQI using LSTM. To evaluate the optimality of the neural network & LSTM, we use mean absolute error.

Figure 19 shows the comparison of Root Mean Square Error calculated between the actual and predicted AQI for ANN and LSTM. It is observed that the RMSE between 8.237991 and 24.35911 for ANN and 1.68412776 and 16.95798994 for LSTM for 145 crossways.

Figure 20 depicts that the SGD optimizer allows a range of RMSE between 1.68412776 and 16.95798994, which is better than the ADAM Optimizer. So from these statistics, it is observed that LSTM with the SGD performs better over ADAM. And so we use SGD for the prediction.

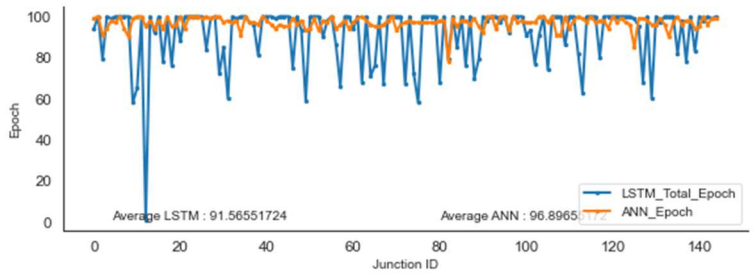


Figure 17 Comparison of the number of epoch required to achieve optimal results

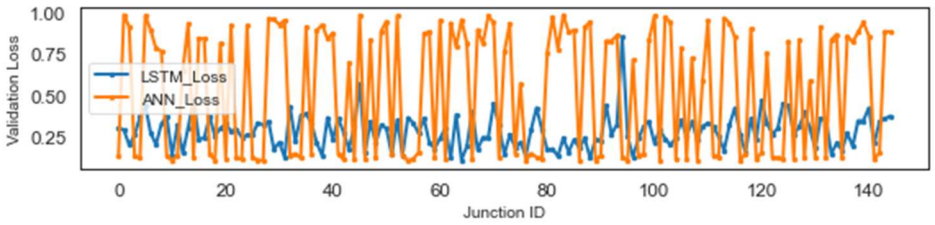


Figure 18 Comparison of Validation loss

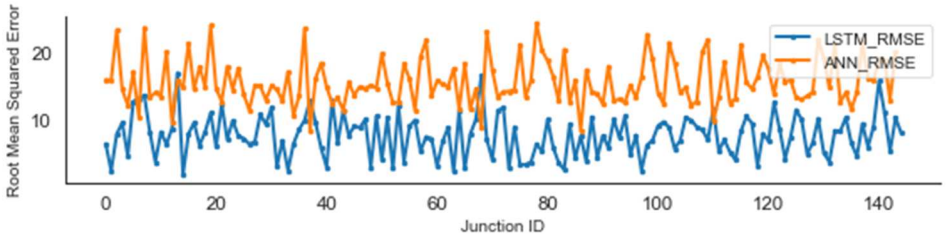


Figure 19 Comparison RMSE

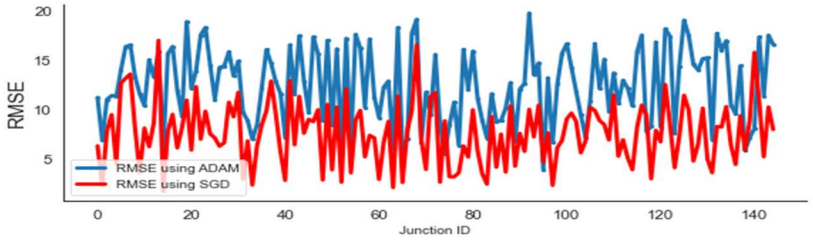


Figure 20 RMSE for the LSTM for 50 neurons in the hidden layer for all the junction using SGD and ADAM Optimizer

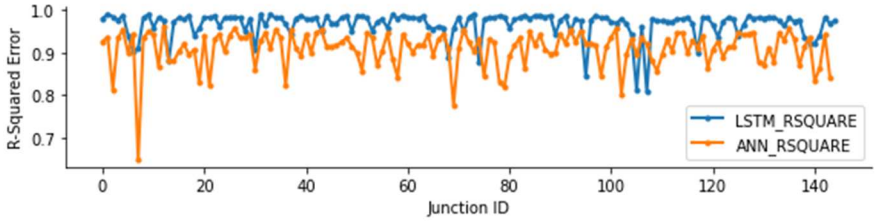


Figure 21 Comparison R-SQUARE

Figure 21 shows the comparison of the R-Square score between LSTM and ANN for the configuration mentioned in table 1 and table 2. From figure 21, we observe that the average value of the R-Square score is 0.909237057 for the ANN while the LSTM R-Square score is 0.966578022. This shows that the LSTM shows a better correlation between actual and predicted values during the testing phase.

Note that the maximum value of AQI is 301 and the Minimum AQI is 21 in the dataset, which at the same time indicates that the model satisfies the variability of the measured values to a great degree. The comparison clearly shows that the LSTM model predicts value with lower root means square error over ANN. And for the LSTM SGD optimizer performs better than ADAM Optimizer.

7. ACHIEVEMENTS WITH RESPECT TO OBJECTIVES

Based on identified problems (section 3) and objectives specified (section 4), we have devised the following:

1. The existing approach to identify and eliminate the cycle in the decision-making process permits all the ants to follow the way where the cycle exists and afterward backtrack the ants. Which thus squanders the computational assets and postpones convergence of the solution. So we find another way to build a novel system. Our novel approach performs better than the existing approach to decrease the number of ants stuck in the cycle and improve computational efficiency.
2. The basic ant colony algorithm will execute at the beginning of the vehicle's tour. Thus in decision making, it will take only the current scenario and suggest the best optimal path. To address this issue, we have proposed executing the ant colony algorithm before reaching the next junction and to find the optimal path towards the destination from the current junction. Thus, our proposed algorithm reduces the impact of pollution on the human body.
3. The experiment related to the prediction of air quality index for the individual junction of the city shows that LSTM performs better than the ANN in terms of the number of epoch required, validation loss, RMSE, and R-SQUARE score.
4. We have devised the novel ant colony-based architecture for air pollution-aware vehicle routing in smart cities using machine learning to find the optimal path.

8. CONCLUSION

In the existing ant colony optimization technique, the ant is stuck into the cycle while constructing a solution, which overuses the computational assets and postpones convergence of the solution. We have proposed modifications in this approach and to modify the selection strategy (Algorithm 5.4 & 5.5). Our proposed solution detects 76% cycles in the routes and decreases 85.4677% of ants stuck in the cycle; this will result in 5.26% less time to solve the problem over the existing approach. Furthermore, the proposed amendment reduces the ant to stuck in the cycle while finding the solution; this will let the ants contribute more to generate optimum solutions rather than stall out in the cycle.

During the execution of the basic ant colony optimization (ACO) algorithm, the algorithm executes at the beginning of the journey. However, we have found that this will not consider the frequently changing parameters like pollution, traffic, etc. Thus we have devised to implement the ACO algorithm dynamically named the dynamic ant colony algorithm (Dynamic ACO). In this approach, we have executed the ant colony algorithm at each junction of the available route. As a result, dynamic decision-making on the route at each junction reduces air pollution impact on the health by slightly increasing or no change in the path length for 78% of the journeys.

The results of Dynamic ACO need computation at each junction. Therefore, the recursive execution of the algorithm requires a reasonable degree of calculation. We have proposed the approach to predict the frequently changing parameter (here in our case, air pollution) before beginning the journey to reduce the recursive computation. The previous studies were based on predicting an aqi for the whole city. The novel part of this study is to predict AQI for each junction individually because air quality varies from intersection to intersection as it depends on several parameters like traffic, industrial area, time of the day, etc. Therefore, we have experimented with artificial neural network (ANN) and long short-term memory (LSTM) machine learning techniques to predict AQI at each junction of the available routes. After the comparative analysis discussed in sections 5 (Point 4) and 6.5, we have concluded that the LSTM predicts AQI accurately than the ANN for the dataset derived for the Ahmedabad City. Moreover, we have proposed the Client-server model to predict AQI for different 145 crossways for the city.

We have devised the novel ant colony-based architecture for air pollution-aware vehicle routing in smart cities using machine learning to find the optimal path using all the experiments and implementation. This architecture is designed to suggest vehicles to move from source to destination, such as riders of the vehicles have the minor effect of air pollutants.

9. A LIST OF ALL PUBLICATIONS ARISING FROM THE THESIS

Sr. No	Title	Journal	ISSN & Publication details
1	Novel approach for traffic directing in urban areas using ant colony optimization technique to diminish the effect of air pollution on the human body	Indian Journal of Environmental Protection (Scopus Indexed International Journal)	**0253-7147** Published in VOLUME 40 ISSUE 11 JANUARY November 2020
2	Architecture For Air Pollution Aware Vehicle Rerouting in Smart cities Using Machine Learning & Ant Colony Algorithm	Studies in Indian place names (UGC Care Listed)	**2394-3114** Published in VOL-40-ISSUE NO.-9-2020
3	Analysis of the issue of ant being stuck in the cycle by ant colony system algorithm & solution	Multidisciplinary International Research Journal of Gujarat Technological University (Peer Reviewed Journal)	**2581-8880** Published in VOLUME 3 ISSUE 1 JANUARY 2021
4	Comparative analysis of artificial neural network and long short-term memory techniques for predicting air quality in smart cities: Ahmedabad city	Indian Journal of Environmental Protection (Scopus Indexed International Journal)	**0253-7147** Acceptance on February 2021

10. PATENTS

No patent has published.

11. REFERENCES

[1] WHO, "WHO global ambient air quality database (update 2018).," 2018, [Online]. Available: https://www.who.int/data/gho/data/themes/topics/topic-details/GHO/ambient-air-pollution

[2] H. Zahmatkesh, M. Saber, and M. Malekpour, "A New Method for Urban Travel Rout Planning Based on Air Pollution Sensor Data," *Curr. World Environ.*, vol. 10, no. Special-Issue1, pp. 699–704, Jun. 2015, doi: 10.12944/CWE.10.Special-Issue1.83.

[3] P. Phadtare, S. K. Singh, S. Karanjkar, S. P. Kulkarni, and S. R. Hiray, "INTELLIGENT NAVIGATION SYSTEM USING AIR QUALITY INDEX," vol. 05, no. 05, p. 4.

[4] M. H. Sharker and H. A. Karimi, "Computing least air pollution exposure routes," *Int. J. Geogr. Inf. Sci.*, vol. 28, no. 2, pp. 343–362, Feb. 2014, doi: 10.1080/13658816.2013.841317.

[5] B. Zou, S. Li, Z. Zheng, B. F. Zhan, Z. Yang, and N. Wan, "Healthier routes planning: A new method and online implementation for minimizing air pollution exposure risk," *Comput. Environ. Urban Syst.*, vol. 80, p. 101456, Mar. 2020, doi: 10.1016/j.compenvurbsys.2019.101456.

[6] A. Sadiq, A. El Fazziki, J. Ouarzazi, and M. Sadgal, "Towards an agent based traffic regulation and recommendation system for the on-road air quality control," *SpringerPlus*, vol. 5, no. 1, p. 1604, Dec. 2016, doi: 10.1186/s40064-016-3282-2.

[7] M. Shanmugam, L. Jayakumar, T. Anand, D. Rajaguru, C. Dhasarathan, and J. Amudhavel, "Air Pollution Based Vehicular Routing Problems: Using Genetic Algorithm Optimization Approach," p. 13, 2018.

[8] E. Payne, "A Resource Constrained Shortest Paths Approach to Reducing Personal Pollution Exposure," *REU Final Rep.*, p. 6, Jun. 2019.

[9] V. Jindal and P. Bedi, "An improved hybrid ant particle optimization (IHAPO) algorithm for reducing travel time in VANETs," *Appl. Soft Comput.*, vol. 64, pp. 526–535, Mar. 2018, doi: 10.1016/j.asoc.2017.12.038.

[10] B. Ding, J. X. Yu, and L. Qin, "Finding Time-Dependent Shortest Paths over Large Graphs," p. 12.

[11] A. Madkour, W. G. Aref, F. U. Rehman, and M. A. Rahman, "A Survey of Shortest-Path Algorithms," p. 27.

[12] N. Thomas, A. Balagopal, and S. M. Varghese, "Green route: An Ecofriendly Route Suggestion and Description Based on Congestion and Air quality," vol. 8, no. 5, p. 6, 2019.

[13] W. Elloumi, H. El Abed, A. Abraham, and A. M. Alimi, "A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP," *Appl. Soft Comput.*, vol. 25, pp. 234–241, Dec. 2014, doi: 10.1016/j.asoc.2014.09.031.

[14] K. Eshghi and M. Kazemi, "Ant colony algorithm for the shortest loop design problem," *Comput. Ind. Eng.*, vol. 50, no. 4, pp. 358–366, Aug. 2006, doi: 10.1016/j.cie.2005.05.003.

[15] J. Zhao and W. Tong, "Solution to the problem of ant being stuck by ant colony routing algorithm," *J. China Univ. Posts Telecommun.*, vol. 16, no. 1, pp. 100–110, Feb. 2009, doi: 10.1016/S1005-8885(08)60187-9.

[16] P. Bedi, N. Mediratta, S. Dhand, R. Sharma, and A. Singhal, "Avoiding Traffic Jam Using Ant Colony Optimization - A Novel Approach," in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, Sivakasi, Tamil Nadu, India, Dec. 2007, pp. 61–67. doi: 10.1109/ICCIMA.2007.61.

[17] V. Jindal, H. Dhankani, R. Garg, and P. Bedi, "MACO: Modified ACO for reducing travel time in VANETs," in *Proceedings of the Third International Symposium on Women in Computing and Informatics - WCI '15*, Kochi, India, 2015, pp. 97–102. doi: 10.1145/2791405.2791476.

[18] J. Dallmeyer, R. Schumann, A. D. Lattner, and I. J. Timm, "Don't Go With the Ant Flow: Ant-Inspired Traffic Routing in Urban Environments," *J. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 78–88, Jan. 2015, doi: 10.1080/15472450.2014.941758.

[19] R. Doolan and G.-M. Muntean, "EcoTrec—A Novel VANET-Based Approach to Reducing Vehicle Emissions," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 608–620, Mar. 2017, doi: 10.1109/TITS.2016.2585925.

[20] D. Varia and A. M. Kothari, "Analysis of the issue of ant being stuck in the cycle by ant colony system algorithm & solution," *Multidiscip. Int. Res. J. Gujarat Technol. Univ.*, vol. 3, no. 1, p. 9, Jan. 2021.

[21] D. Varia and A. M. Kothari, "Novel Approach for Traffic Directing In Urban Areas Using Ant Colony Optimization Technique to Diminish the Effect of Air Pollution on the Human Body," *Indian J. Environ. Prot.*, vol. 40, no. 11, pp. 1146–1153, Nov. 2020.

[22] D. Varia and A. M. Kothari, "Comparative analysis of artificial neural network and long short term memory techniques for predicting air quality in smart cities: Ahmadabad city," *Indian J. Environ. Prot.*, p. 2.

[23] "CityPulse Project Anual Report," SEPTEMBER '13 - AUGUST '14. [Online]. Available: http://www.ict-citypulse.eu/page/sites/default/files/citypulse_annual_report.pdf

[24] M. I. Ali, F. Gao, and A. Mileo, "CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets," in *The Semantic Web - ISWC 2015*, vol. 9367, M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, Eds. Cham: Springer International Publishing, 2015, pp. 374–389. doi: 10.1007/978-3-319-25010-6_25.

[25] M. Mohan and A. Kandya, "An Analysis of the Annual and Seasonal Trends of Air Quality Index of Delhi," *Environ. Monit. Assess.*, vol. 131, no. 1–3, pp. 267–277, Jun. 2007, doi: 10.1007/s10661-006-9474-4.

[26] B. S. Fakinle, J. A. Sonibare, O. B. Okedere, L. A. Jimoda, and C. O. Ayodele, "Air quality index pattern of particulate around a haulage vehicle park," *Cogent Environ. Sci.*, vol. 2, no. 1, Jul. 2016, doi: 10.1080/23311843.2016.1208448.

[27] "National Air Quality Index - CPCB,India," 2014, pp. 5–12. [Online]. Available: http://www.indiaenvironmentportal.org.in/files/file/Air%20Quality%20Index.pdf

[28] "OpenStreet Map." https://www.openstreetmap.org

[29] *Simulation of Urban MObility(SUMO)*. [Online]. Available: https://sumo.dlr.de/

[30] "Status of Pollution Generated from Road Transport in Six Mega Cities," *Cent. Pollut. CONTROL BOARD*, p. 105, Mar. 2015.