Abstract—In this paper, efficiency of machine learning algorithm, ACO has been improved for solving Travelling Salesman Problem (TSP) in finding optimal path by varying its parameters (α and β explained later) using Fuzzy inference system. TSP is a NP hard combinatorial optimization problem which means no algorithm is known to solve it in polynomial time. Ant colony optimization algorithm is first applied and then studied for the parameters that highly affect its performance. Fuzzy logic is applied over it and performance in terms of the optimal distance is being improved. It can be extended to similar problems. The FACO (fuzzy controller with ACO) algorithm is tested many times on Travelling Salesman Problem with varying sizes and input sets, and the results are compared with those of the basic ACO algorithm. This comparison showed that the FACO algorithm outpace the basic ACO algorithm and reached near-optimum or ideal solutions wherein the basic ACO could not give best possible or near to optimum solutions.

Index Terms—ACO, TSP, Fuzzy Inference System, Fuzzy Controller, Fuzzy logic.

I. INTRODUCTION

Although ACO has a high potent to find out solutions to combinatorial optimization problems, it suffers from two main problems: stagnation and premature convergence. Also the behaviour of ACO in terms of convergence speed is very sluggish. These problems become more evident as the problem size increases [2]. In basic ACO algorithm the parameters of the ACO algorithm while solving have fixed values throughout the run of algorithm. The parameters vary depending upon the application and the problem size, so there are no fixed known values that can make the algorithm to accomplish best possible results in all situations. The idea being proposed is to set these values mechanically such that their values would alter throughout the course according to certain performance measures, which will yield best performance of the algorithm and closer convergence towards the optimum solutions [1]. ACO for solving TSP is implemented but with a Fuzzy Controller module that is added to the algorithm in which as the algorithm runs, parameters are evolved automatically giving near to ideal solutions. The fuzzy rules are represented by the rule-base of the fuzzy controller which is used for governing the performance of the ACO algorithm corresponding to the variations in the value of parameters (α and β). Genetic algorithm was used to deduce fuzzy rules that produce set of consequents to its corresponding antecedents. Fuzzy Logic takes into account the vague values instead of just the crisp values and using the set of fuzzy rules it deduces a better solution in today’s scenario where human
natural language (not just discrete values) is the most common and appropriate input to any system. The parameters affecting the performance of basic ACO algorithm are taken into consideration, and best optimal solutions are found by running and analysing the algorithm multiple times.

II. PROBLEM FORMULATION

TSP represents a class of problems (NP-hard combinatorial optimization problem set) which is analogous to finding the least-cost sequence for visiting a set of cities, starting and ending at the same city in such a way that each city is visited exactly once. A general description of the Travelling Salesman Problem is given as: Let’s consider a set N of nodes, a set A of arcs, connecting the nodes fully (fully connected graph), where N represents cities and A represents links between the cities of TSP. Let \( r_{ij} \) be the distance between the nodes I and j, \( (i, j) \in A \), representing distance between the two cities \( (i, j) \), where \( r_{ij} \) is the length of arc \( (i, j) \). A graph G = (N,A) could be constructed. TSP could be defined as the problem of finding on the graph G, a minimal length Hamiltonian circuit. Hamiltonian circuit for the graph G is a closed circuit or tour visiting each city once for all the nodes or cities in \( N \) of graph G \( (n = |N|) \), and length of that Hamiltonian circuit (or the solution to the problem) is equal to the sum of the spans of all the arcs of the circuit. TSP is a special case of the travelling purchaser problem [1]. The worst-case running complexity for any algorithm used for solving TSP upsurges in super polynomial time (or imaginably exponentially) as the number of cities increase.

III. SOLVING TSP

The approach used for solving TSP has been showed in the flowchart in fig.1. The ant colony optimization algorithm or ACO is a technique for solving computational problems using probabilistic approach to find optimal solution to a given problem. The standard form of basic ACO is applied to get an optimal solution of best length. The error and variance in value of best length of optimal path (difference between solution obtained and best known solution) is supplied as an input to fuzzy controller for a particular input set. The generated rule-base produces the output the values of \( \alpha \) (heuristic factor describing how greedy the algorithm is) and \( \beta \) (heuristic factor describing how fast ants are going to select path).

![Fig. 1: Solution design](image_url)
Defuzzification of these values yields crisp values of $\alpha$ and $\beta$. These values are then input to basic ACO again to get near to optimal solution. All the modules are explained in detail in further sections.

**A. Ant Colony Optimization Algorithm**

ACO is a meta-heuristic, implying that it is a general framework that can be referred to produce a specific algorithm to find solution to a particular graph path problem. The description of ACO was given by M. Dorigo in a 1991 doctoral thesis. The detailed explanation and description of the algorithm is credited to a 1996 follow-up publication by M. Dorigo, V. Maniezzo and A. Colorni [2]. Soon after that, ACO has been widely studied and improved. However, very few complete, correct and optimal implementations have been known to exist. The approach of ACO that is being used is shown in fig. 2. Stagnation and Maximum no of iterations are the stopping criteria. Stagnation behaviour is observed when pheromone accumulates on a particular path, which in most cases is a local optimal solution. More ants keep choosing this path over and over again until eventually all the ants in the algorithm choose this path and the algorithm converges prematurely to this native or local optimum solution [1]. To avoid problem of stagnation, ants are initially placed at different cities and each ant is initialized with random trail. Best local trail is determined for each ant which is shortest in terms of total distance covered and its length is determined. Pheromone is initialized at each edge. During the construction of a feasible solution, ants select the following city to be visited through a probabilistic decision rule. When an ant $k$ is at city $i$ and constructs the partial solution, the probability of moving to the next city $j$, which is neighbours with city $i$, is given by equation (1) [2].

![Fig. 2: Structure of ACO Algorithm](image)

\[ p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \text{tabu}_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} \quad \text{if } j \in \text{tabu}_i^k \]  

\[ \text{(1)[6]} \]
where tabu gives the list of feasible neighbours of city-i for the ant k. where $\tau_{ij}$ is the intensity of pheromone trails between edge(i,j) and $\eta_{ij}$ is the heuristic visibility of edge (i, j), and $\eta_{ij}$=1/d_{ij}. The heuristic visibility for an edge tells us how good it is. This gets raised to the power of “beta” which is another heuristic parameter, describing how fast the ants select their path. Here, $\alpha$ and $\beta$ are two flexible positive parameters that control the relative weights of the pheromone trail and of the heuristic visibility [1]. After each ant completes its tour, the local pheromone update is performed by all the ants after each construction step. Each ant applies it only to the chosen city.

$$\tau_{ij}(t+1) = (1-\rho) \tau_{ij}(t) + \rho \tau_0 \cdot \Delta \tau_{ij}(t) \quad ..(2)[6]$$

where $0<\rho<=1$ is a local evaporation factor, $\tau_0=1/n$ (additional pheromone). After all the ants have travelled through all the cities, the amount of the pheromone on the optimal path is updated only as per the equations 3 and 4 [7] (global updating of trail):

$$\eta_{ij}(t+1) = (1-\rho) \eta_{ij}(t) + \rho \Delta \tau_{ij}(t) \quad ..(3)[6]$$

$$\Delta \tau_{ij}(t) = \frac{1}{L_{gb}} \quad ..(4)[6]$$

where $p$ is constant, $L_{gb}$ is the length of global best tour.

**B. Ant Colony Behaviour**

In the natural world, ants (initially) stroll randomly for finding food, and upon finding food they return to their colony while laying down trails of pheromone. If other ants find a path with pheromone, they are probable not to keep wandering in haphazard manner, but instead to follow the trail with pheromone, returning and reinforcing it with more pheromone if they sooner or later find the food. However, over time, the pheromone trail tends to evaporate, thereby reducing its strength for attracting ants. The larger time it takes for an ant to the pathway and return to home, the more the pheromone tends to evaporate. A shorter path, gets trailed over more frequently, which results in higher pheromone on shorter paths than on longer ones. The advantage of pheromone evaporation is that it avoids the convergence to a locally optimal solution as if there were no vaporization of pheromone at all, the pathways chosen by the initial ants would tend to be exceptionally attractive to the subsequent ones. In such case, the exploration of new paths would be inhibited [3]. Thus, when one of the ants finds a suitable path its colony to a food source, that path is more likely to be followed by other ants and a positive feedback ultimately leads to all the ants following the same path. The idea of the basic ant colony algorithm is to imitate this behaviour with simulated ants strolling the graph obtained representing the problem to be solved.

**C. Applying ACO to TSP**

A standard input set of 42 cities was taken in the form of a matrix of order 42. The aim is to find the shortest path in which each of these 42 cities is visited exactly once. Certain parameters need to be specified as a requirement of ACO such as the pheromone influence factor (alpha) the pheromone evaporation coefficient (rho) and heuristic parameter (Beta) describing how fast ants are going,. All the ants are initialized with a random trail for the 42 cities. After being initialized, the ant which has a trail of shortest dimension or length is the one which is the best ant. Best trail corresponding to best ant is to be found. The significant notion of ACO is the usage of simulated pheromones, which results in ants getting attracted to better trails through the graph. Mainly the processing loop alternates between two modules. The first module updates the ant trails using the current pheromone values and the second module updates the pheromones using newly explored ant trails. The 42-city graph representing the problem is artificially fabricated so that every city is linked to every other city. The distance between any two cities has been taken as an input which is a predefined matrix of connected cities in units (miles, km and so forth). There’s no definite way to solve the problem of TSP. With 42 cities, assuming any city could be the starting city and ants could either advance or regress, and that all the cities are connected, there comes out to be total $(42 - 1)! / 2$ possible solutions. Assuming that if 1 billion possible solutions per second could be evaluated, it would take about $2.2*10^{63}$ years to analyse all of them, which is many times more than the estimated age of the universe. In ACO algorithms ants are agents which, in the case of TSP, construct tours by moving from one city to another on the problem graph. The expected number of iterations required for an ACO-based algorithm with n ants is $O \left( \frac{1}{\rho} n^2 m \log n \right)$ for graphs with n nodes and m edges, where $\rho$ is an evaporation rate of pheromone. Fig. 3 shows the global best
tour that was obtained using ACO. Graph representing best trail is shown in figure 3, x-coordinate of city (in kilometres/miles) along X-axis and y-coordinate of city along Y-axis (in kilometres/miles).

D. Updating the Ants

The main aspect of the ACO algorithm is the process that updates each ant and trail by constructing a new and better trail based on the pheromone and distance information. Consider an example where we have a small graph with just five cities as shown in Fig. 4 the new trail for an ant is under construction. The trail starts at city 3, and then goes to city 4, and the update algorithm is determining the next city. The pheromone and distance information is as shown in figure 4. For determining next city we would first construct an array called “taueta” (because of Greek letters τ (tau) and η (eta)). The taueta value is determined by value pheromone on the edge being raised to the power of alpha, multiplying it with one over the distance value raised to the power of beta. Recall that alpha and beta are heuristic parameters that must be specified.

Here it is assumed that alpha is 3 and beta is 2. The taueta values for city 3 and city 4 are not computed because they are already in the current trail. The larger values of the pheromone increase taueta, but larger distances decrease taueta. After all the taueta values have been computed, those values are converted to
probabilities and placed in an array labelled probs. The algorithm sums the taueta values, which is 82.26 in this example. Each of these values is then divided by the sum. Here, city 0 has a probability of 0.09 of getting selected and so on. Then the algorithm selects the next city based on the computed probabilities. An augmented array called cumul is constructed, which holds cumulative probabilities, size of which is one greater than the probs array, and cell (0) is seeded with the value of 0.0. Each cell in the cumul array represents the cumulative sum of the probabilities. After construction of cumul array, a random number p is generated between 0.0 and 1.0. Say, p = 0.538 as shown. The value of the random number p falls between the values at (2) and (3) in the cumul array, so (2), or city 2, is selected as next city.

IV. ACO WITH FUZZY LOGIC CONTROLLER

Fuzzy logic was first proposed by Lotfi A. Zadeh from University of California at Berkeley in a 1965 publication. He explained his ideas that presented the concept of “linguistic variables”, or a fuzzy set. Fuzzy logic is widely used in controlling of machines. The term “fuzzy” implies that the logic involved can deal with concepts that cannot be expressed in terms of discrete logic (0 or 1) "true" or "false" but somewhat as “partially true” (somewhere between 0 and 1). Alternative methodologies such as genetic algorithms and neural networks are known to perform just the same way as fuzzy logic in most of the cases, fuzzy logic has the advantage of the solution to the problem being cast in the terms that could be understood by human operators. This lets their experience to be used in the design of the fuzzy controller which makes it easier to automate activities that have been successfully executed by human beings already. Fuzzy logic starts with and builds on a set of user-supplied human language rules. These rules are then converted to their mathematical equivalents by the fuzzy systems, simplifying the job of the system designer and the computer, and resulting in much more accurate representations of the way systems behave in the real world. A fuzzy system can be created to match almost any set of values of input-output sets. For this purpose Fuzzy Logic Toolbox is used. The Fuzzy Logic supplies adaptive techniques such as Adaptive Neuro-Fuzzy Inference Systems (ANFIS) and Fuzzy Subtractive Clustering. Fuzzy logic is fabricated on to using the knowledge of experts who understand a particular system. It counts on the experience of the ones who comprehend the system.

A. Fuzzy Inference System

A Fuzzy Inference System (FIS) is a system in which an input space is mapped to an output space by making use of fuzzy logic. In other words, an FIS tries to validate the reasoning process of human language using fuzzy logic. For this purpose fuzzy IF-THEN rules are built. FIS is used to solve decision problems. Fig. 5 depicts the components of fuzzy inference system.

In general a fuzzy system consists of 4 modules:

- Fuzzification module: In this module, the system inputs which are crisp numbers are transferred into fuzzy sets using fuzzification function.

- Knowledge base: This module contains IF-THEN rules are, which are provided by experts (in human understandable form).

- Inference engine: This module simulates the process of human reasoning by constructing fuzzy inference on the input set and IF-THEN rules in the rule list.
- Defuzzification module: This module transforms the fuzzy set acquired by the inference engine into the form which could be used outside the system i.e. a crisp value (reverse of fuzzification module). It is easy to amend an FIS just by adding or deleting rules. There is no requisite to make a fresh FIS from scratch.

**B. Type- I Fuzzy Sets**

Type-I fuzzy rules, is an efficient tool for quantitative modelling of words or sentences in a natural or artificial language. Although type-II fuzzy sets, known for handling uncertainty could be used as well but in this case it would have yielded similar results. So, for the sake of simplicity type-I fuzzy sets were used. The basic structure of a type-I fuzzy inference system is composed of three intangible modules:-

- Rule base: containing selection of fuzzy rules defining value of consequents for different combinations of antecedents. Different combinations were generated using Weka, a data mining tool.
- Database: defining the membership functions used in the fuzzy rules. It is also termed as dictionary.
- Reasoning mechanism: operating upon the rules and facts thereby, deriving a reasonable conclusion or output from input.

Fig. 6 depicts the fuzzy inference system from inside. In general it can be said that a fuzzy inference system implements a nonlinear mapping from its input space to output space. Inputs of the database that is the “If...Then” rules are called the antecedents and the outputs are called consequents. Input values are supplied to the fuzzification module which evaluates those using rules from knowledge base. Knowledge base consists of input membership functions, rule list and output membership functions. The values obtained after rule evaluations are then defuzzified using a particular technique and crisp values are yielded. These obtained values can serve as input to the existing system for yielding optimal solutions. [4] [5] [7].

![Fig. 6: Inside Fuzzy Inference System](image)

**C. Defuzzification**

Defuzzification is the process of generating a finite or quantifiable result in fuzzy logic, with the given fuzzy sets and corresponding membership degrees. Membership degrees are described by membership functions corresponding to each value of the input as well as output sets. Membership functions can be of different types. Triangular membership functions were used in the system. For e.g. in the proposed system, the input value of error could be classified into high, average and low. For all these categories 3 values exist which makes a triangle when this fuzzy set is represented graphically. A Number of variables are transformed into a fuzzy result or output which is dependent on membership in fuzzy sets. A common and useful defuzzification technique is centre of gravity. Firstly, the results of the rules are summed up together. As explained above the concept of triangular membership functions, the most typical fuzzy set membership function has the graph in the shape of a triangle. Now for the purpose of defuzzification, this triangle is cut in a straight line
horizontally somewhere in the middle of it. After removing the top and the top slice of the triangle, the remaining slice results in a trapezoidal shape. The first step of defuzzification stereotypically cuts off parts of the graphs forming trapezoids (or other if the starting shapes were other than triangles). These trapezoids are then superimposed one over other, resulting in formation of a single geometric shape. The fuzzy centroid is then obtained by obtaining the centroid of this shape. The x coordinate of the centroid represents the defuzzified value [4][5][7].

Fig. 7 shows fuzzy graph for temperature conditions of hot and not hot for corresponding fuzzy sets (using triangular membership function). In this system solutions obtained using basic ACO were compared with that of best known solutions. The error and variance so calculated were fed as antecedents (inputs) into FIS. FIS rules were generated using membership function which tells the degree of associativity of crisp values. This is done using data mining tool, Weka which yielded values of parameters $\alpha$ and $\beta$, defuzzification of which yielded crisp values of $\alpha$ and $\beta$ to be used in ACO for increasing its efficiency. These values were then fed to basic ACO again and optimal solutions were obtained.

![Fuzzy graph](image_url)

Fig. 7: Fuzzy graph

V. COMPARISON AND CONCLUSION

For different problem sets efficiency of both ACO and ACO with fuzzy controller or FACO was compared and results obtained in latter case were closer to best known solution as shown in Table I. The no. of cities (n) and nodes (m) were kept same.

<table>
<thead>
<tr>
<th>ACO</th>
<th>FACO</th>
<th>Best Known Solution</th>
<th>Percentage optimization of the results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>452.1</td>
<td>431.9</td>
<td>426</td>
<td>4.7</td>
</tr>
<tr>
<td>778</td>
<td>708</td>
<td>699</td>
<td>10.01</td>
</tr>
<tr>
<td>562.6</td>
<td>556</td>
<td>538</td>
<td>1.22</td>
</tr>
<tr>
<td>1270.2</td>
<td>1234.2</td>
<td>1211</td>
<td>2.97</td>
</tr>
<tr>
<td>672.1</td>
<td>655.6</td>
<td>629</td>
<td>2.62</td>
</tr>
<tr>
<td>14665.4</td>
<td>14543.1</td>
<td>14379</td>
<td>0.85</td>
</tr>
<tr>
<td>46347</td>
<td>45757.9</td>
<td>44303</td>
<td>1.32</td>
</tr>
<tr>
<td>60559.6</td>
<td>60203.3</td>
<td>59030</td>
<td>0.63</td>
</tr>
<tr>
<td>2449.1</td>
<td>2401.9</td>
<td>2323</td>
<td>2.03</td>
</tr>
<tr>
<td>83521.5</td>
<td>83141.5</td>
<td>80369</td>
<td>0.47</td>
</tr>
<tr>
<td>56131.6</td>
<td>53429.6</td>
<td>49135</td>
<td>5.49</td>
</tr>
</tbody>
</table>

When the output of the ACO is being studied and analysed it is found that it is highly dependent on the
values of alpha and beta. So these values are being focused upon and error and variance is being calculated for each of the input values. A graph was plotted as well for both the algorithms for different values of error and variance depicting comparison in efficiency of two algorithms. Fig. 8 shows comparison of efficiency of ACO and ACO with Fuzzy with length of best trail along X-axis and error (difference between obtained solution and best known solution) along Y-axis. The above comparison yields the conclusion that ACO with fuzzy controller is more efficient in finding optimal solution to TSP compared to standard ACO and yields nearly best optimal known solution for a particular problem set.

![Graph showing comparison of ACO and FACO](image)

**Fig. 8: Comparison of ACO and FACO**

VI. **FUTURE SCOPE**

TSP is an NP-hard combinatorial optimization problem, which implies solving it in polynomial time or finding optimum solution to such a problem will help us solve all similar problems of that category such as quadratic assignment, protein folding, routing vehicles, routing network, circuit design, set problem, scheduling problem or so. TSP is often used as a benchmark in optimization techniques. When altered slightly, it appears as a sub-problem in many problems. One such application is that of DNA sequencing. Furthermore in the cases when the graph may change dynamically, the ant colony algorithm can be run incessantly and acclimatize to the alterations in real time. This lures the researchers in the field of network routing and urban transportation systems to optimize the performance of ACO.

**REFERENCES**