ISSN No:-2456-2165

A Fuzzy Logic for Parameter Adaptation in Ant Colony Optimization Approach

Sawsan A. Al-Agamy[1], Fadl Mutaher Ba-Alwi[2], Abdulqader M. Mohsen[3]
[1,2]Department of Information Systems, Faculty of Computer and IT, Sana'a University, Sana'a, Yemen
[3]Department of Information Systems, Faculty of Computing and IT, UST, Sana'a,Yemen

Abstract:- The purpose of this study is to produce an "adaptive Ant Colony System" in order to establish a balance between exploitation and exploration in terms of solving the Travel salesman problem. First we do a detailed investigation of several Ant Colony System Algorithm's parameters. Second we will incorporate into the algorithm a Fuzzy Logic Controller, which will be utilized to alter the settings based on the algorithm's reliable performance metrics. The parameter customization will proceed throughout the execution of the algorithm, providing for a dynamic parameter settings depending on the algorithm's current performance. The adaptive algorithm will be examined on a set of TSP problems of varying sizes, and the results will be compared to those obtained using the standard algorithm and other studies in the same manner.

Keywords:- *Travel Salesman Problem(TSP),Meta-heusitic Algorithms, Ant Colony Optimization(ACO),Parameter tuning, Fuzzy Logic Controller(FLC), Fuzzy Ant Colony System (FACS).*

I. INTRODUCTION

Since its inception in 1800, the Travel Salesman Problem (TSP) has attracted the interest of several researchers in mathematics, operations studies, computational intelligence, and various branches of engineering, resulting in the focus of multiple research approaches. Consciously, the TSP has evolved into a standard metric for evaluating the search capability of optimization algorithms [1]. A series of techniques to solve TSP have been proposed that produce either an exact or an approximate solution. Algorithms such as branch-and-bound and dynamic programming are used to provide an accurate solution to the TSP with a limited number of cities. For a large number of cities, however, identifying an inexact answer may be sufficient. Numerous metaheuristic methods based on swarm and evolutionary methodologies estimate optimal solutions to the given TSP[2]. The ant colony optimization method (ACO) is a metaheuristic algorithm that is based on the behavior of ant colonies during their mutual search for food. In 1992, Dorigo developed ACO, an approach based on the behavior of ant colonies that use their collective intelligence and the pheromone trail left by actual ants to locate food; the population in ACO is referred to as artificial ants, and Each ant symbolizes a possible solution to the problem[3]. Over the years, ACO has established itself as a successful metaheuristic for combinatorial optimization issues (COPs)[4]. The ACO algorithm's significance stems from its ability to discover high-quality solutions quickly. As a result, ACO algorithms are increasingly being used for problems involving continuous and multi-objective optimization[5]. Intuitively, the core concept of ACO optimization algorithms is the balance between intensification and diversification. However, an overemphasis on intensification can drive ants to a local optimum, whereas an overemphasis on diversity can result in an unstable state[6] . The parameters setup of swarm algorithms has a significant impact. Significant involvement in defining behavior, leading the search, and biasing the quality of final solutions throughout resulted in an appropriate balance of intensification and diversification. Additionally, an ideal parameter value may vary during the optimization process, complicating this task[7]. In ACO, additional research is needed into the sensitivity of parameter selection and its relationship to convergence. Existing research on the behavior of state-of-the-art parameter adaption algorithms focused on only a few ACO variants and neglected the others[8]. The use of fuzzy logic to metaheuristics has lately become a significant area of research, as evidenced by several publications. A Fuzzy Dynamic Adaptation of Parameters is one of these instances when fuzzy logic is used to dynamically make a parameter of an algorithm fuzzy[9]. We suggest a dynamic parameter adaption strategy for ACO in this paper, utilizing a fuzzy logic controller (FLC), in order to establish a balance between exploitation and exploration in terms of solving the TSP problem. By adjusting the settings over time, while taking into consideration certain metrics concerning the ACO algorithm's performance. We do a detailed investigation of several ACO parameters. The suggested algorithm's efficacy is compared to[10] and [11] and other recent literature related works.

II. IMPROVEMENT TECHNIQUES IN METAHEURISTICS

The TSP is based on the notion of a traveling salesman visiting a specified number of cities (nodes) and determining the shortest path between them. A TSP has a maximum of C nodes linked by edges E. The edge E_{ij} that connects nodes i and j has a cost n_{ij} and is frequently equal to the Euclidean distance between nodes [12]. The TSP is useful in a variety of practical applications, including transportation (by air, land, or sea), vehicle routing work scheduling , logistics planning, motherboard drilling, collision avoidance in robotics engineering, industrial robot control, layout of computer motherboard components, manufacturing microchips, DNA sequencing and more recently, automotive vehicle engineering science[13]. TSP is characterized as NP-hard problems that cannot be solved in a finite amount of time. Unless NP equals P, a polynomial time area exists.

Researchers considered alternative workarounds (approximation approaches) that may produce a satisfactory result in an acceptable amount of time; these methods are classified as heuristics and metaheuristics. The critical distinction between the two is that heuristics are problem-dependent, but metaheuristics are not. There have been an immense number of metaheuristics developed to date for adapting to varied issue kinds (continuous, discrete, unconstrained, multi-objective, etc.). Although the fundamental metaheuristics may be rather efficient, additional transformations and improvements are periodically presented to boost the algorithm's and solution's convergence[14]. presented a hybrid Artificial Bee Colony ABC to solve the scheduling problem for identical parallel batch processing machines. The authors assert that the time required to solve the scheduling problem is reduced[15]. suggested a hybrid discrete Grey Wolf Optimizer GWO, Crow Search Algorithm (CSA) and Extreme Learning Machine (ELM) to solve the multi-objective and multi-constraint scheduling problem for casting manufacturing.[16] Developed a novel hybrid Teaching-Learning-Based Optimization TLBO method combined with Extreme Learning Machines (ELM) for solving data categorization issues. On a collection of UC Irvine (UCI) benchmark datasets, the suggested approach is evaluated. [17] Describes a hybrid Iterated Local Search and Ant Colony Optimization technique. The ILS-AntMiner rules-based method is for increasing classification accuracy and model size. By enhancing the benefit of neighborhood structures in the exploitation mechanism, this hybridization intends to improve the classification performance in terms of accuracy and simplicity. For the Dynamic Traveling Salesman Problem DTSP,[18] presented a novel a hybrid metaheuristic algorithm for the DTSP . This approach is hybrid of two metaheuristic principles: ant colony optimization (ACO) and simulated annealing (SA). [10] Proposed a self-adaptive ACO with novel strategies to improve the algorithm's uncertain convergence time and random decisions The proposed technique (DEACO) dynamically adjusts the ACO parameters. The main idea behind this mechanism is to determine the first city (start point) in order to find the shortest path using clustering. DEACO uses this method to find the cheapest/shortest path for each cluster. [11] proposed a method for dynamically adapting the responsible parameters for the decay of pheromone trails xi and rho using a fuzzy logic controller (FLC) in the travelling salesman problems (TSP). The goal of this method is to understand the effect of both parameters xi and rho on the performance of the ACS in terms of solution quality and convergence speed to the best solutions by studying the ACS algorithm's behavior during this adaptation. [19]Proposed a parallel cooperative hybrid algorithm (PACO-3Opt) based on ant colony optimization To avoid local minima, the 3-Opt algorithm is used. PACO-3Opt has multiple colonies and a master–slave paradigm. [20] suggested a new algorithm for the TSP that combines gravitational particle swarm optimization (PSOGSA) and ACO and is known as PSOGSA-ACO-LS (ant supervised by gravitational particle swarm optimization with a local search). On the other hand, [21] and [22] stated that metaheuristics' performance is dependent on parameter setting and may allow for more flexibility and resilience. Since the turn of the twentieth century, the literature has progressively emphasized the need of systematic methodologies for metaheuristic parameter setting. The following subsections explore and summarize several parameter tweaking procedures.

A. *Overview of parameter adaptation approaches*

It is not straightforward to determine the parameter settings. Actually there is no such thing as an ideal values of parameters for a particular metaheuristic. There are two distinct methodologies for parameter tuning, according to [23] which are off-line parameter initialization (or meta-optimization) and the online parameter tuning strategy: Off-Line Parameter Initialization: where appropriate parameter settings are determined prior to the method being applied to the situation at hand. In this situation, the parameter tuning results, i.e. the ideal parameter setting determined during the tuning process, are utilized to solve problems, and these parameter values remain constant throughout the run. Online Parameter Initialization: Where the values of the algorithm's controlled parameters change in real time as a result of various techniques (i.e., during the run). In this case, startup values and suitable control techniques for controlled parameters are provided, which update or adjust the values of relevant parameters during the run. The following are examples of possible control strategies:

- deterministic or pre-scheduled, If the problem is viewed from an offline standpoint: Depending on the computing time or the number of algorithm iterations, static parameters are updated by (deterministic or randomized) functions.
- adaptive parameter settings, where the strategy for parameter adjustment is described in terms of certain statistics about the algorithm's behavior. This online adaptation may employ a variety of measures.
- self-adaptation, This consists in the algorithm modifying the parameters at run time. More precisely, dimensions representing parameters of exploration methods are added to the problem's search space. After that, the optimization procedure is carried out in this new space.
- search-based adaptation, which uses a different search method for parameter adaptation than the underlying algorithm. Local search and EAs for parameter adaptation are examples of this type of strategy.

B. *Ant Colony System (ACS)*

Ant Colony Optimization (ACO) was first introduced by Dorigo in 1992 as part of his doctoral thesis. Ant Colony Optimization is a pure meta-heuristic approach that is capable of solving a broad variety of issues and resulting in several optimizations in the disciplines of science and engineering. Numerous variations of the ACO algorithms have been developed in recent years, including the Elitist ant system, the Rank-based ant system, the Max–min ant system, and the Ant colony system (ACS). We will focus our efforts in this study on the last-mentioned form of the ACO algorithm. Following is a brief description of the algorithm for the ant colony system: To begin, randomly place m ants in n cities, with each edge containing an initial pheromone τ_{ij} (0) among two cities. Each ant's tabu list's initial entry is set to be equal to its beginning town. Following that, each ant migrates from town i to town j. Ants choose the next city using the following probability formula[24].

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{h\in N^{k}}[\tau_{ih}]^{\alpha} * [\eta_{ih}]^{\beta}} \qquad (1)$$

where:

- τ_{ij} : Pheromone trace amount in *t* time in the *(i,j)* corners.
- The visibility or heuristic information between a pair of cities is the inverse of their distance $\eta_{ij} = 1/d_{ij}$, where d_{ij} is the distance between cities *i* and *j* . Hence, if the distance on the arc (*i, j*)is long, visiting city *j* after city *i* (or vice versa) will be less desirable.
- α (alpha) is the parameter that shows the relative importance of the pheromone trace.
- β (beta) is the parameter that shows the importance of the visibility value.
- N^k denotes its practicable neighborhood. Nodes that have previously been visited as part of the ant *k* partial tour are not included in the viable neighborhood . The pseudo-random proportionality rule is used by ACS in the solution construction, though the next city to visit is picked as follows with a probability *q0*:

$$j = arg\ \max_{h\in N^{k}}\{\tau_{ih} * \eta_{ih}^{\beta}\} \qquad (2)$$

That is, with a certain probability, the most appealing edge is chosen greedily. The ACS pheromone deposit modifies just the pheromone levels of adjacent solutions. This is either the best-iteration-so-far solution or the best-so-far solution. The formula for pheromone updates is as follows:

$$\tau_{ij} \longleftarrow \begin{cases}(1-\rho) * \tau_{ij} + \rho * \Delta\tau_{ij}\text{if}(i,j)\text{is a part of BS } s^{best}, \\ \tau_{ij} \qquad \text{otherwise,}\end{cases} \qquad (3)$$

$$\text{with } \Delta\tau_{ij} = F(s^{best})$$

During the ants' solution construction process, a local pheromone update rule is used. Each time an ant makes its way across an edge *(i,j)*, τ_{ij} is modified as

$$\tau_{ij} \longleftarrow \ (1-\xi) * \tau_{ij} + \xi * \tau 0, \qquad (4)$$

where ξ ∊ (0, 1) denotes the pheromone decay factor and *τ0* denotes the pheromones' initial value. In ACS, *τ0* is a very small constant with the value *1/n * L_{nn}* , where L_{nn} is the duration of the nearest neighbor tour; it lowers the pheromone value of previously utilized edges, making them less interesting to other ants. Table I summarize some of the main approaches that have been used in the ACO literature to adapt parameter values. As viewed in Table 1 it's clear that is in the recent years, fuzzy systems have been used as a strategy to find the best parameters in the ACO variants and the obtained results are significance. However we can observe the importance of use fuzzy systems for parameter adaptation. **Even so, with well-designed fuzzy controllers and more sensitive fuzzy rules, the algorithm achievement will be optimized more and more to achieve the best results. This is the main objective of our research.**

III. METHODOLOGY

ACO algorithms' behavior is very dependent on the values assigned to parameters [39]. However, altering the parameters during calculation, either on a pre-determined timetable or in response to the search progress, might improve an algorithm's performance. ACO algorithms are increasingly being used to solve problems involving continuous and multi-objective optimization[40]. According to[21], the behavior of ACO algorithms is highly dependent on the parameter values. In the majority of ACO applications, parameter values are maintained constant during the algorithm's execution. The parameter setting problem has garnered more attention from developers and end-users of metaheuristics, and increased work has been directed toward devising systematic and sophisticated techniques to resolve it [23]. As a result, it is common to discover ACO algorithms that employ fuzzy logic to attain the best outcomes [41]. While other approaches, such as genetic algorithms and neural networks, perform similarly to fuzzy logic in the majority of circumstances, fuzzy logic has the benefit of expressing the answer to the issue in words that human operators can understand. This enables them to incorporate their knowledge into the design of the fuzzy controller, making it easier to automate operations that have been effectively performed by humans previously. As a result, it was proposed to enhance the algorithm's performance by modifying ACO parameters at run-time using a fuzzy logic controller (FLC). ACO algorithms require the proper setting of a number of parameters. Among these:

Paper	Adaptation strategy	ACO variant	Parameters
[25]	pre-scheduled	variant of AS	*β,ρ*
[26]	pre-scheduled	AS	*α*
[27]	Adaptive	ACS	*ρ*
[28]	Adaptive	variant of MMAS	*β*
[29]	Adaptive	ACS	*ρ*
[30]	self-adaptation	MMAS	*α,β*
[31]	self-adaptation	MMAS	*α,β*
[32]	search-based adaptation	variant of ACS	*β,ρ,q0*
[33]	search-based adaptation	variant of ACS	*α,ρ,Q*
[34]	search-based adaptation	variant of ACS	*β,q0*
[24]	search-based adaptation	ACS	*α,β,ρ,Q,S*
[35]	fuzzy logic system	ACS	*β,q0*
[36]	fuzzy logic system	AS	*α,ρ*
[37]	interval type-2 FLC	AS	*α,ρ*
[38]	fuzzy logic system	ACS	*population*
[11]	fuzzy logic system	ACS	*ρ,ξ*

Table 1:- Schematic description of the literature on adaptive ACO

- *α* and *β* which are used to weigh the relative influence of the pheromone and heuristic values in the ants' solution construction. The role of these parameters in biasing the ants' search is intuitively similar. Higher values of *α* emphasize differences in the pheromone values, and *β* has the same effect on the heuristic values.
- The starting value of the pheromones, *τ*0, has a substantial effect on the algorithm's convergence speed;

even so, the ideal value varies according to the ACO algorithm.

- The evaporation rate parameter, ρ, $0 \leq \rho \leq 1$, controls the degree to which pheromone trails degrade. If ρ is low, the effect of the pheromone values would last longer, but high values of ρ allow for rapid forgetting of formerly very attractive alternatives, allowing for a more rapid concentration on newly added information to the pheromone matrix.
- The colony's population of ants, *m*. For a certain computational budget, such as a maximum computing time, the population of ants is essential in defining the trade-off between the number of iterations possible and the breadth of the search at each iteration. Indeed, the fewer rounds the ACO algorithm performs, the larger the number of ants every iteration becomes.
- *Q*, a The amount of pheromone an ant releases is determined by this parameter.
- *q*0, the possibility of making a predictive decision about the next city to visit.

Choosing which parameters to change and how to change them is mostly a random process. Table 2 shows some of the state-of-art attempts to achieve the parameters tuning.

In our work, a deep comparison was utilized to determine what parameters to employ in the solution. Among these:

A. Pheromone and Heuristic parameters settings:

α and β which are used to weigh the relative influence of the pheromone and heuristic values in the ants' solution construction. The role of these parameters in biasing the ants' search is intuitively similar. Higher values of α emphasize differences in the pheromone values, and β has the same effect on the heuristic values. If α is too big, ant colony search may become prematurely imprisoned in local optima because of the increased possibility of re-choosing the path due to stochastic variables. The larger the information heuristic factor α, the stronger the stochastic components are in the path search. However, ant colony's prior knowledge of path search and uncertainty factors, as measured by the relative importance of β, indicates the greater likelihood for ants to select a local shortest path on the local point, even though the search convergence rate increases, the ant colony weakens randomness to fall into the local optimum more easily. The ant colony algorithm's performance and role are both complementary and intertwined[42]. A satisfactory outcome may be achieved by selecting the right α and β ranges; typically, $\alpha = 0.5 \sim 2.0$ and $\beta = 2.0 \sim 5.0$ are used.

B. Global search and convergence speed parameter setting

The colony's population of ants, *m*. For a certain computational budget, such as a maximum computing time, the population of ants is essential in defining the trade-off between the number of iterations possible and the breadth of the search at each iteration. Indeed, the fewer rounds the ACO algorithm performs, the larger the number of ants every iteration becomes. The ACO algorithm uses only one type of ant to generate new solutions, and the ant colony size, selection parameter, and convergence parameter are used to control the solutions[43]. In order to select the ant colony algorithm's m (number of ants), two indications of the algorithm's global search capability and convergence speed must be considered extensively. However, it is interesting to note that the method performs significantly worse at low values (m = 1) than it does at high (m = 100) values. Since the ants' numbers might vary from 30 to 100, we did as suggested and varied the number of them accordingly.

Optimization Method	**α**	**β**	**Ants population m**	**q0**	**ρ**
[44]	**1~2**	**2~3**	**-**	**-**	**0.7~0.8**
[38]	**-**	**2**	**0~100**	**0.9**	**0.1**
[45]	**-**	**2~5**	**10**	**0.9**	**0.1**
[10]	**-**	**-**	**200**	**-**	**-**
[46]	**1~1.7**	**3~6**	**50~150**	**-**	**0.3**
[6]	**1**	**2**	**-**	**-**	**0.5**
[47]	**0~1**	**-**	**number of cities(m=n)**	**-**	**-**
[24]	**-**	**2**	**m=n**	**0.5**	**0.2**
[48]	**1**	**2**	**100**	**-**	**0.1**
[49]	**0.75~1.20**	**0.65~1.50**	**-**	**-**	**-**

Table 2 Optimization methods parameters values samples

C. Solution construction parameter setting

q0, the possibility of making a predictive decision about the next city to visit.

Additionally [39] suggested that q0 values that are close to 1 are considered to be good. A value of 1 causes search stagnation quickly, whereas values less than 0.75 result in very slow convergence forward into good solutions. Therefore we make our work with q0 adapting values between 0.0 and 0.9.

D. Evaporation rate parameter setting

The evaporation rate parameter, ρ, $0 \leq \rho \leq 1$, controls the degree to which pheromone trails degrade. If ρ is low, the effect of the pheromone values would last longer, but high values of ρ allow for rapid forgetting of formerly very attractive alternatives, allowing for a more rapid concentration on newly added information to the pheromone matrix.

IV. THE PROPOSED METHOD

It has been popular in recent years for researchers to use various strategies to dynamically modify important parameters in ACO algorithms in order to obtain better convergence. It is our goal to improve performance by dynamically modifying parameters during algorithm execution solely for ACS algorithms that employ fuzzy logic. Based on the previous discussion about the role of each parameter in the ACS algorithm, we built four Fuzzy Logic Controllers FLCs and utilize them to concurrently adapt the four parameters α , β, $q0$ and population size in the Fuzzy Ant Colony System FACS technique, which uses the fuzzy system to dynamically alter the parameters i.e. they are exactly the outputs of the fuzzy system. This parameter selection has not been taken into account in other studies, as shown in table I. The FLC uses errors and diversity as inputs and then applies a set of linguistic knowledge to them in order to adjust the given parameters. The algorithm 1 shows the four basic phases of the typical ACS. As shown in figure 1, the stochastic greedy rule, which differs from other state transition rules is providing some kind of balance between exploration of new solutions and exploitation of a priori information about the problem. ACS algorithm applied this rule iteratively by each ant in the Construction Solution step to build a feasible solution. The amount of pheromone in the solution is constantly being adjusted. A local search technique is then activated to improve the algorithm's searchability. The amount of pheromone is updated when all the ants have come up with a viable solution. Finally, the method returns the best solution it has identified if it meets the stop condition or finds the ideal solution.

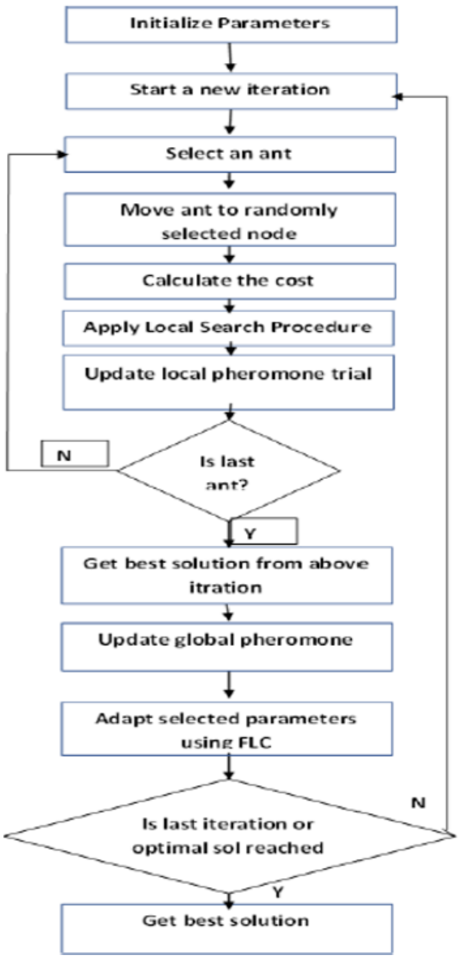
Initialize Parameters
Start a new iteration
Select an ant
Move ant to randomly selected node
Calculate the cost
Apply Local Search Procedure
Update local pheromone trial
N
Is last ant?
Y
Get best solution from above itration
Update global pheromone
Adapt selected parameters using FLC
N
Is last iteration or optimal sol reached
Y
Get best solution

Fig 1:- The FLC steps

A. *FLC Model Components*

In this section, we will go over the FLC design in detail as well as how to use this model to control the parameter adaptation in the ACS algorithm. The Fuzzy Logic Control is generally structured by determining the inputs, outputs, database, and fuzzy rule base as shown in fig.2 .Each FLC component needs to be characterized and implemented. Carefully, in order to get more efficient FLCs of related parameters. As known, the main design of each controller is the same. It consists of similar parts to any FLC. Even so, each FLC has some variations in the way of adaptation and its main components. The designs of these parts differ from one FLC to the other, following the strategy of every controller.

1) *Determine The Membership Functions :*

The definition of membership functions is a particularly delicate aspect of the Fuzzy Controller's design. The number of membership functions used in a fuzzy controller is critical, as is the cardinality of each variable's universe of discourse. Each element of the discourse universe must belong to at least one of these fuzzy sets to be considered, i.e. to fire at least one rule. With a large number of membership functions, more resolution is gained.[50]

Algorithm 1 Proposed FACS Algorithm
```
1.Initialization
repeat
  for Each ant do
    2.Buildasolutionaccordingto1
    3.LocalSearch()
    Local Phermone Update
  end for
  Find Best-Ant-Tour-Length()
  Find worst-Ant-Tour-Length()
  Find Average()
  Calculate distance between ants()
  4.ComputeDiversity
  5.ComputeConvergenceMeasure
  if Itration mod ==50 *to do this step every 50 iteration*
  then
    6.ExecuteFLCModule
  end if
  7.Global Phermone Update
  for Each ant do
    8.Calculate tour − length
  end for
  update statics()
  9.writedatatofile
  Iteration++
until Itrations $\neq$ Max and best − tour $\neq$ OptimalSolutio
10.Return $L_{best}$
```

which motivates us to define five membership functions for each of the proposed FLC's input and output variables and to develop 25 different rules as demonstrated in fig 3. Another factor to examine is the normalization of the FC's membership functions. Normalizing any membership function by having at least one input with a value of one is more of a practical rule than an universal guideline. We'll always assume that we're dealing with normalized membership functions. Furthermore, only a portion of the challenge of converting fuzzy logic language concepts to membership functions has been solved. [51]has been shown that fuzzy control membership functions are often piece wise linear, triangular (three parameters) because they are the most efficient from the point of view of sensitivity where piece wise linear functions are the least sensitive. In this proposed FC we will use only three types of most used types in design which are the triangular, S- shaped membership function (which is spline-based curve and is named because of its S-shape) and the Z-shaped membership function (it is spline-based too and named because of its Z-shape).

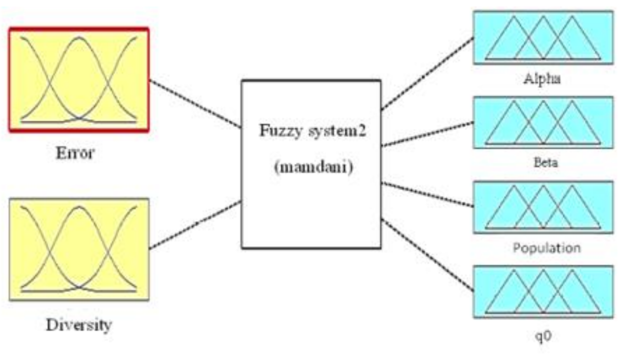

Fig. 2. The Fuzzy System Design

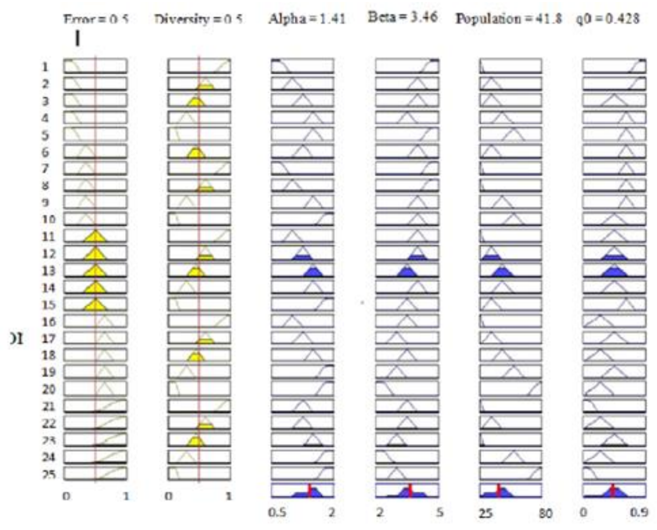

Fig. 3. The Fuzzy Rules Design

2) *Inputs for FLC Model:*
In this FLC model the inputs are:

- The Convergence metric: The Convergence is calculated by the current best length divided by best of tours length as follows:

$$Convergence = \frac{CurrentBestTourLength}{BestofToursLength} \quad (5)$$

- The second input variable, diversity, was acquired by calculating the Euclidean Distance (ED) between the fitness of the ants in the population as shown:

$$ED = \frac{\bar{d} - d_{min}}{d_{max} - d_{min}} \quad (6)$$

where $\bar{d}$ is the average difference in fitness between the best ant and the rest of the population's ants. d_{min} and d_{max} are are the distances between the worst ant fitness and the second best ant fitness and the best ant fitness, respectively[52] . Convergence and diversity are constructed in the fuzzy system as three triangular and two zmf membership functions, respectively, with a range between 0 to 1.

3) *Outputs for FLC Model:* For ACS based on experiments, the parameters α and β from Eq(1) were
chosen to be tuned dynamically. The Member functions of α and β was designed with a range from 0.5 to 2.00 for α and with range from 2.00 to 5.00 for β As mentioned before, the *q*0 range was chosen between 0.0 to 0.9. For the population size (*n*) the range was from 25-80 and all were structured into three triangular and two *zmf* membership functions.

V. EXPERIMENTS AND DISCUSSION

This section will present the experimental results used to assess the efficiency of the proposed algorithm.

A. *Expermints setup*

Initially to conduct this experiments we have used 10 TSP standard benchmark problems, with difference between them on the number of cities from TSPLIB [53]. The fuzzy controller was built in MATLAB in the first step, and the proposed methodology was constructed in Java on an Intel Core-i5 2.40GHz and 4GB RAM PC in the second stage.The symmetric TSP was used in all experiments. When the designed algorithm found the optimal solution or achieved 1000 iterations, it was terminated. Several text scripts and EXCEL sheets are used to correlate the collected results and solutions. To carry out a comparative analysis, we as well choose from the instances in [10] and [11]. Table 3 described the chosen problems. In addition for the ρ parameter which is not adapting within the fuzzy system we've put the initial value of it between 0.01 to 0.9 as [21] was proven. These instances may be categorized into three categories based on their duration. The first group has a short length, ranging from 55 to 101 cities. The second category may be classified as medium, as it has examples ranging from 105 to 124 cities. The last category is the large-scale one, which contains cases with a length of 575, 1323 and 1655 cities respectively.

Problem	Number of cites	Best known
Eil76	76	538
Berlin52	52	7542
eil101	101	629
ch130	130	6110
lin105	105	14379
kroA100	100	21282
pr124	124	59030
rat575	575	6773
rl1323	323	270199
d1655	1655	62128

Table 3 Tsp Benchmark Instances

Thus, the tests were repeated ten times, and the findings were gathered for each instance with the best average and standard deviation being used for comparison. At first, we compared the suggested method to the typical algorithm (ACS). The proposed algorithm was then compared to two state-of-the-art algorithms: [10] and [11]. Following that, it will be compared to three recent works on the ACO algorithm: [20], [19] and [54]. We evaluated the suggested method using the following metrics: best solution (Min), worst solution (Max), average solution, standard deviation, number of iterations, and execution time.

B. *Analysis of FACS Performance*

Consequently, from Table 4, it can be observed that the quality of the solutions acquired by the suggested algorithm was much better than the solutions generated by the conventional ACS for all evaluation metrics. It was proven that the FACS really boosted the accuracy of the conventional ACS. As a consequence, the adaptive ACS (FACS) converged to optimal solutions more quickly than the normal ACS, which may be advantageous for time-constrained problems. For bigger TSP cases, the FACS algorithm identified optimum or near- optimal solutions that the normal ACS method did not, which will be necessary for problems requiring precise answers. Additionally, a local search strategy was added to the speed of algorithm convergence, yielding a significant impact.

TSP	**Algorithm**	**BKS**	**Min**	**Max**	**Ave**	**stddev**	**itration**	**Time(sec)**
Eil76	ACS Proposed	538	538 538	552 538	545 538	4.5 0.0	507 1	0.415 0.028
Berlin52	ACS Proposed	7542	7542 7542	7748 7542	7598 7542	91.5 0.0	121 1	0.047 0.014
eil101	ACS Proposed	629	635 629	655 629	647 629	5.10 0.0	703 3	0.7333 0.045
ch130	ACS Proposed	6110	6178 6110	6377 6110	6267 6110	73.1 0.0	876 3	0.927 0.1
lin105	ACS Proposed	14379	14379 14379	14957 14379	14545 14379	180 0.0	946 1	0.605 0.039
kroA100	ACS Proposed	21282	21281 21281	22523 21281	21619 21281	388 0.0	829 1	0.759 0.021
pr124	ACS Proposed	59030	59167 59030	60158 59030	59676 59030	422 0.0	346 1	0.43 0.02
rat575	ACS Proposed	6773	9632 6773	10100 6786	9892 6778	144 3.49	535 976	5.468 78.01
rl1323	ACS Proposed	270199	314403 270199	331710 270546	320970 270287	5715 138.81	920 180	16.146 76.24
d1655	ACS Proposed	62128	69139 62131	73593 62202	71154 62324	1421.9 71.14	994 529	26.54 338.041

Table 4 Summary of The Results Obtained by Proposed Algorithm And the Standard ACS

Figures 4, 5, 6 and 7 illustrate the proposed method's performance with four various sizes of TSP instances, representing short, medium, and large. However, these findings can be applied to other TSP cases. Figures 4 and 5 represent the outcomes of a single run for the eil101 and pr124 cases, respectively. The figure illustrates the Min, Max, and average solutions acquired during the run, as well as the FACS convergence behavior. As seen in Fig.4 and 5, the superior performance and resilience of FACS resulted in the optimal solution being found for both instances in all 10 iterations. The figure illustrates how the trend rapidly decreases toward the optimal solution during the initial repetitions. It is noteworthy that at the early stage, FACS exhibits a high degree of convergence with diverse solutions. FACS strikes an optimal balance between diversity and intensity during the search process. This stability enables FACS to thoroughly scan the search space and converge on areas of a probable optimum. Moreover, there was no stagnation in the search.

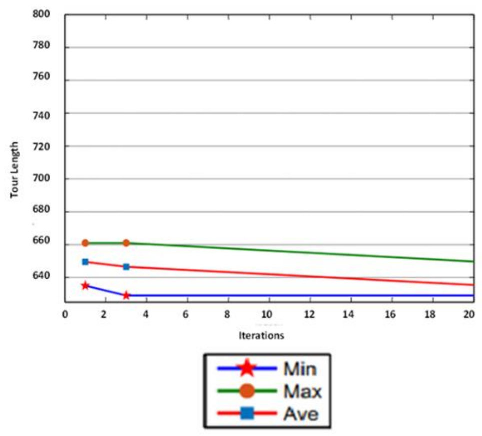
Fig. 4. eil101 instance Representing Short Instances

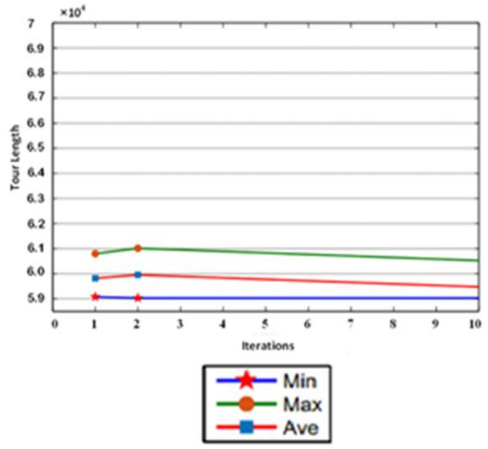
Fig. 5. Pr124 instance Representing Short Instances

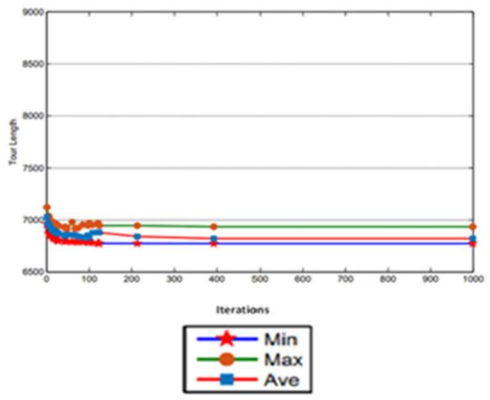
Fig. 6. Rat575 instance Representing Medium Instances

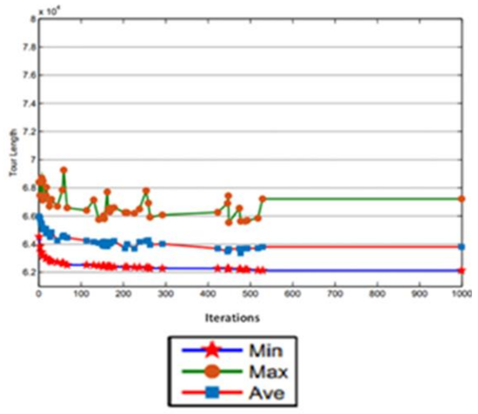
Fig. 7. d1655 instance Representing Long Instances

Figure 6 demonstrates the results of rat575. As shown the suggested method found the best solution on the fifth of ten runs, which equals 6773. As seen, the algorithm achieved a solution that was close to optimal in the majority of iterations until it hit the optimal solution at iteration 393 and then remained constant thereafter. This advantage of FACS may be due to the fact that the inclusion of the fuzzy system preserves the population variance of the algorithm, investigates new regions of the search space, and prevents the algorithm from being stuck in local optima. Figure 7 shows the results of a single run for the d1655 instance, which has 1655 cities. This figure illustrates the algorithm's convergence characteristics after 1000 iterations. Although the search space for that instance was wide, the proposed method avoided stagnation *and* early convergence, as seen by the method's convergence behavior until it reached the sub-optimal solution, 26131 at iteration 539. This smooth convergence occurred as a result of the suggested algorithm's capacity to strike an optimal balance between intensification and diversity.

C. Comparison with other state-of-the-art Algorithms

The first part of this section will demonstrate the results from our FACS algorithm compared with two recent ACS algorithms:[10] and [11]. We examined measures such as Standard Deviation (SD) and Average Solution throughout this comparative phase (Avg). As a consequence, assuring the outcomes' optimality may be deduced from the tiny SD value, which indicates the consistency and efficiency of the associated method. In Table IV, column 1 displays the TSP alongside the best known solutions (BKS), column 2 displays the best solutions through all runs (Min), column 3 displays the average solution across all runs, column 4 displays the standard deviation, column 5 displays the error rate, which is calculated by comparing the percentage deviation of the best results to the best known solution(Error), and The following formula was used to determine the error:

$$Error = \frac{BestSolution - OptimalSolution}{OptimalSolution} * 100 \quad (7)$$

Table 5 shows that FACS was able to achieve shorter distances in the majority of the cases compared to the boldface text. The proposed algorithm produces the best results in terms of solution quality, with an error of zero. for eil76, berlin52,eil101,ch130, lin105, kroA100, Pr124 and rat575, and gives remarkable results for rl1323 and d1655. Furthermore the proposed algorithm was exceed bouzbita2020 [9] method on all instances. Whereas the proposed algorithm and S. Ebadinezhad2020 [8] did not vary much on the small instances. However, we can assert that the suggested method outperforms Ebadinezhad2020 [8] method for medium- and large-scale cities such as pr124, rat575, rl1323, and d1655. , In terms of the five assessment criteria indicated before,. This is due to a well-developed method for changing parameters at run time, which was introduced to the conventional algorithm.

Instance	**BKS**	**Proposed**				**S. Ebadinezhad2020**				**bouzbita2020**			
		Min	**Ave**	**SD**	**Error**	**Min**	**Ave**	**SD**	**Error**	**Min**	**Ave**	**SD**	**Error**
Eil76	538	538	538	0.0	0.0	541.6	541.6	0.61	0.85	548	556	-	1.86
Berlin52	7542	7524	7524	0.0	0.0	7542	7542	0.0	0.0	7544	7589	-	0.03
eil101	629	629	629	0.0	0.0	629	629	-	0.0	646	663	-	2.70
ch130	6110	6110	6110	0.0	0.0	6110	6110	0.0	0.0	6246	6348	-	2.23
lin105	14379	14379	14379	0.0	0.0	14379	14379	0.0	0.0	14383	14525	-	0.02
kroA100	21282	21282	21282	0.0	0.0	21282	21282	0.0	0.0	21285	21612	-	0.01
pr124	59030	59030	59030	0.0	0.0	59074	-	-	0.074	-	-	-	-
rat575	6773	6773	6778	3.49	0.0	6773	6804	10.3	0.0	-	-	-	-
rl1323	270199	270199	270287	138.81	0.0	-	272173	436.2	-	-	-	-	-
d1655	62128	62131	62324	71.14	0.005	-	63432	100.97	-	-	-	-	-

Table 5 comparison of proposed algorithm with a state-of-art algorithms

Figure 8 shows a comparison of the proposed algorithm to Ebadinezhad [10] and Bouzbita [11] based on the percentage deviations of the average solution to the best known solution.

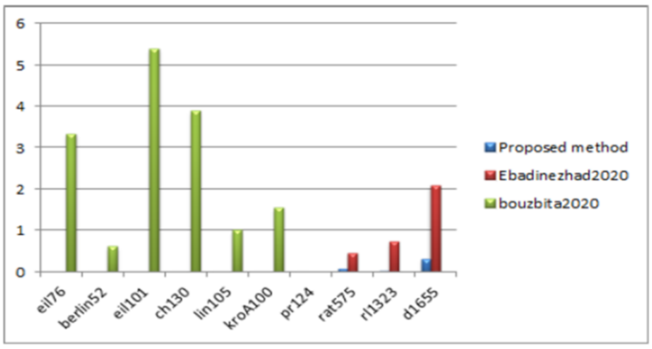

Fig 8. Percentage Deviations for the Proposed Algorithm and the Other two Algorithms.

Clearly, the suggested method outperforms Bouzbita [11] in a significant manner, and gained smaller percentage deviations than Ebadinezhad [10] in the large-scale TSP instances that are rat575, rl1323 and d1655. According to [55], there are numerous researches related to the advancement of ACO algorithm.

Table 6 compares the proposed method to three previously published publications. This part of the experiment involves the execution of eight TSP instances with 100 simulation runs independently. As seen in the table, the suggested method produced the best results in all eight examples in terms of Average Solution (Avg), Standard Deviation (SD), and with an error of zero in all instances. Additionally, the minimal SD value demonstrates the algorithm's consistency and efficiency.

The findings show that the suggested algorithm's architecture, which relies on a fuzzy system and local search technique to adjust the algorithm's most influential parameters during run time, enables the algorithm to exit from local - optima and accelerate convergence.

Additionally, the suggested technique obtained suboptimal/optimal solutions in all circumstances, distinguishing it from the other methods.

Instance	**BKS**	**Rokbani et al**			**Gülcü et al**			**Khan et al.**			**Proposed method(FACS)**		
		Avg	**SD**	**Error**	**Avg**	**SD**	**Error**	**Avg**	**SD**	**Error**	**Avg**	**SD**	**Error**
eil51	426	428.9	01.85	0.47	426.35	0.49	0.08	427.01	0.46	0.23	**426**	**0.0**	**0.0**
berlin52	7542	**7542**	224.7	**0.0**	**7542**	**0.0**	**0.0**	**7542**	**0.0**	**0.0**	**7542**	**0.0**	**0.0**
st70	675	**675.18**	7.079	**0.0**	677.85	0.99	0.42	675.77	1.17	0.18	**675**	**0.0**	**0.0**
eil76	538	547	12.62	1.67	539.85	1.09	0.3	538.17	0.60	0.02	**538**	**0.0**	**0.0**
rat99	1211	1225	29.24	1.16	1217.10	4.01	0.50	1211.50	0.67	0.041	**1211**	**0.0**	**0.0**
eil1101	629	646	12.42	2.70	630.55	2.63	0.25	630.59	2.37	0.25	**629**	**0.0**	**0.0**
kroA100	21282	21346	695.24	0.301	21326.80	3.72	0.21	**21282**	8.10	0.02	**21282**	**0.0**	**0.0**
kroA200	29368	29850	606.17	1.64	29644.50	53.43	0.94	29469	20.03	0.146	**29368**	**0.0**	**0.0**

Table 6 a comparison of proposed algorithm with a other algorithms

Instance	BKS	[6]			[56]			Proposed method(FACS)		
		Min	Avg	SD	Min	Avg	SD	Min	Avg	SD
eil51	426	426	426	-	426	426	0.0	426	426	0.0
berlin52	7542	7542	7542	-	7542	7542	0.0	7542	7542	0.0
st70	675	675	675	-	-	-	-	675	675	0.0
eil76	538	538	538	-	538	538	0.0	538	538	0.0
eil1101	629	629	629	-	629	629	0.0	629	629	0.0
kroA100	21282	21282	21312.34	-	21282	21282	0.0	21281	21281	0.0
kroA200	29368	29378	29682.72	-	29368	29368	0.0	29368	29368	0.0
ch130	6110	6110	6125.52	-	6110	6110	0.0	6110	6110	0.0
Rat575	6773	6883	6933	-	6348	6367.3	8.48	6773	6778	3.49
rl1323	270199	-	-	-	272.487	273.367.9	484.78	270199	270287	138.81
d1655	62128	-	-	-	63.428	63.707.87	114.84	62131	62324	71.14

Table 7 a comparison of proposed algorithm with a other algorithms

Table 7 shows results of the FACS algorithm by [6]] and [56] for the 11 instances from the TSP. The results indicate that the proposed method's results were statistically significant for three out of eleven datasets, namely, for the large datasets rat575, rl1323, and d1655.

VI. CONCLUSION AND FURTHER WORK

NP-hard optimization problems like TSP have been solved using swarm intelligence techniques like ACO. The ACO has three primary components that influence its execution time and solution quality. The first is the ants' solution creation mechanism for selecting (next) nodes. The second is pheromone memory, which might be costly to maintain. The LS heuristic is the third component, and it is used to increase the quality of the ants' solutions. At least one of the components described is addressed in a major portion of the ACO-related research in the literature[57].Moreover, due to the parameter values in standard algorithm are stable, ACO's performance is unsatisfactory, and this leads to premature convergence. ACO's shortcomings inspired us to develop a new method for dynamically modifying ACO parameters while the algorithm is running. A proposed algorithm based on the Fuzzy Ant Colony System is given in this respect (FACS). TSPLIB examples ranging from 51 to 1655 cities were used to test the FACS algorithm in a variety of ways. Results reveal that FACS has a faster convergence time, avoids the local optimum, and is better suited to solving the TSP issue. The suggested algorithm's experimental results were compared to those of other leading algorithms. This method outperforms others in terms of small/medium sized problems and near optimal solution in terms of large sized problems. Moreover it has smaller percentage deviations in comparison to Bouzbita[11], Ebadinezhad[10], Rokbani et al[20] , Gülcü et al. [19] , Khan et al [54], [6] and [56]. algorithms. In ¨ terms of generality, the suggested approach may be used for a variety of optimization problems. In the future, the suggested approach may be used to create dynamic parameter adaption methods for the Ant Colony System (ACS) using interval type2 fuzzy systems. Furthermore, additional assessment of the proposed fuzzy algorithm's performance may be employed in other applications, such as the optimization of other types of controllers or dealing with more complicated TSP problems.. Furthermore, additional assessment of the proposed fuzzy algorithm's performance may be employed in other applications, such as the optimization of other types of controllers or dealing with more complicated TSP problems.

REFERENCES

[1]. Odili, J.B., A. Noraziah, and R.M. Sidek. *Swarm intelligence algorithms' solutions to the travelling salesman's problem*. in *IOP Conference Series: Materials Science and Engineering*. 2020. IOP Publishing.

[2]. Chaudhari, K. and A. Thakkar, *Travelling Salesman Problem: An Empirical Comparison Between ACO, PSO, ABC, FA and GA*, in *Emerging Research in Computing, Information, Communication and Applications*. 2019, Springer. p. 397-405.

[3]. Olivas, F., et al., *Dynamic parameter adaptation for meta-heuristic optimization algorithms through type-2 fuzzy logic*. 2018: Springer.

[4]. de Oliveira, S.M., et al., *A computational study on ant colony optimization for the traveling salesman problem with dynamic demands.* Computers & Operations Research, 2021. **135**: p. 105359.

[5]. Ragmani, A., et al., *FACO: a hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing.* Journal of Ambient Intelligence and Humanized Computing, 2019. **11**(10): p. 3975-3987.

[6]. Gao, W., *New Ant Colony Optimization Algorithm for the Traveling Salesman Problem.* International Journal of Computational Intelligence Systems, 2020. **13**(1): p. 44.

[7]. Parpinelli, R.S., et al., *A review of techniques for online control of parameters in swarm intelligence and evolutionary computation algorithms.* International Journal of Bio-Inspired Computation, 2019. **13**(1): p. 1-20.

[8]. Alameer, I.A. and R. Sagban, *A Comparative Evaluation of Parameter Adaptation Methods in Ant Colony Optimization.* Journal of Computational and Theoretical Nanoscience, 2019. **16**(3): p. 1182-1189.

[9]. Melin, P., et al., *Fuzzy logic in intelligent system design: Theory and applications*. Vol. 648. 2017: Springer.

[10]. Ebadinezhad, S., *DEACO: Adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem.* Engineering Applications of Artificial Intelligence, 2020. **92**: p. 103649.

[11]. Bouzbita, S., A. El Afia, and R. Faizi, *The behaviour of ACS-TSP algorithm when adapting both pheromone parameters using fuzzy logic controller.* International Journal of Electrical and Computer Engineering (IJECE), 2020. **10**(5): p. 5436.

[12]. Beer, C., T. Hendtlass, and J. Montgomery. *Improving exploration in ant colony optimisation with antennation*. in *2012 IEEE Congress on Evolutionary Computation*. 2012. IEEE.

[13]. Deng, W., et al., *A novel collaborative optimization algorithm in solving complex optimization problems.* Soft Computing, 2016. **21**(15): p. 4387-4398.

[14]. Hancer, E., et al., *Pareto front feature selection based on artificial bee colony optimization.* Information Sciences, 2018. **422**: p. 462-479.

[15]. Fu, W., et al., *Multi-step short-term wind speed forecasting approach based on multi-scale dominant ingredient chaotic analysis, improved hybrid GWO-SCA optimization and ELM.* Energy Conversion and Management, 2019. **187**: p. 356-377.

[16]. SevİNÇ, E. and T. DÖKeroĞLu, *A novel hybrid teaching-learning-based optimization algorithm for the classification of data by using extreme learning machines.* Turkish Journal of Electrical Engineering & Computer Sciences, 2019. **27**(2): p. 1523-1533.

[17]. Al-behadili, H.N.K., K. Ku-Mahamud, and R.J.J.T.A.I.T. Sagban, *Hybrid ant colony optimization and iterated local search for rules-based classification.* 2020. **98**(04): p. 657-671.

[18]. Stodola, P., et al., *Hybrid Algorithm Based on Ant Colony Optimization and Simulated Annealing Applied to the Dynamic Traveling Salesman Problem.* Entropy (Basel), 2020. **22**(8): p. 884.

[19]. Gülcü, Ş., et al., *A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem.* Soft Computing, 2016. **22**(5): p. 1669-1685.

[20]. Rokbani, N., et al., *A new hybrid gravitational particle swarm optimisation-ACO with local search mechanism, PSOGSA-ACO-Ls for TSP.* Int. J. Intelligent Engineering Informatics,, 2019. **7**(4): p. 384-398.

[21]. Stützle, T., et al., *Parameter adaptation in ant colony optimization.* 2011: p. 191-215.

[22]. Talbi, E.-G., *Metaheuristics: from design to implementation*. Vol. 74. 2009: John Wiley & Sons.

[23]. Huang, C., Y. Li, and X. Yao, *A Survey of Automatic Parameter Tuning Methods for Metaheuristics.* IEEE Transactions on Evolutionary Computation, 2020. **24**(2): p. 201-216.

[24]. Peker, M., et al., *An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method.* 2013. **21**(Sup. 1): p. 2015-2036.

[25]. Merkle, D., M. Middendorf, and H.J.I.t.o.e.c. Schmeck, *Ant colony optimization for resource-constrained project scheduling.* 2002. **6**(4): p. 333-346.

[26]. Meyer, B. *Convergence control in ACO*. in *Genetic and Evolutionary Computation Conference (GECCO), Seattle, WA, late-breaking paper available on CD*. 2004.

[27]. Hao, Z., et al. *An ACO algorithm with adaptive volatility rate of pheromone trail*. in *International Conference on Computational Science*. 2007. Springer.

[28]. Kovářík, O. and M. Skrbek. *Ant colony optimization with castes*. in *International Conference on Artificial Neural Networks*. 2008. Springer.

[29]. Cai, Z., et al., *Ant Colony Optimization Based on Adaptive Volatility Rate of Pheromone Trail.* 2009. **2**(8): p. 792-796.

[30]. Martens, D., et al., *Classification with ant colony optimization.* IEEE Transactions on Evolutionary Computation, 2007. **11**(5): p. 651-665.

[31]. Khichane, M., P. Albert, and C. Solnon. *An ACO-based reactive framework for ant colony optimization: First experiments on constraint satisfaction problems*. in *International Conference on Learning and Intelligent Optimization*. 2009. Springer.

[32]. Hao, Z.-F., R.-C. Cai, and H. Huang. *An adaptive parameter control strategy for ACO*. in *2006 International Conference on Machine Learning and Cybernetics*. 2006. IEEE.

[33]. Ling, W. and H. Luo. *An adaptive parameter control strategy for ant colony optimization*. in *2007 International Conference on Computational Intelligence and Security (CIS 2007)*. 2007. IEEE.

[34]. Anghinolfi, D., et al. *Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling*. in *Asia-Pacific Conference on Simulated Evolution and Learning*. 2008. Springer.

[35]. Amir, C., et al., *A fuzzy logic controller for ant algorithms.* 2007. **11**(2): p. 26.

[36]. Olivas, F., F. Valdez, and O. Castillo. *Dynamic parameter adaptation in Ant Colony Optimization using a fuzzy system for TSP problems*. in *IFSA-EUSFLAT*. 2015.

[37]. Olivas, F., et al., *Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems.* Advances in Intelligent Systems and Computing, 2017. **53**: p. 74-87.

[38]. Bouzbita, S., A. El Afia, and R. Faizi. *Adjusting population size of ant colony system using fuzzy logic controller*. in *International Conference on Computational Collective Intelligence*. 2019. Springer.

[39]. Dorigo, M. and T. Stützle, *Ant colony optimization. Cambridge, Massachusetts: A Bradford Book*. 2004, MIT Press.

[40]. Arunarani, A.R., D. Manjula, and V. Sugumaran, *Task scheduling techniques in cloud computing: A literature survey.* Future Generation Computer Systems, 2019. **91**: p. 407-415.

[41]. Olivas, F., et al., *Interval type-2 fuzzy logic for dynamic parameter adaptation in a modified gravitational search algorithm.* Information Sciences, 2019. **476**: p. 159-175.

[42]. Wei, X.J.I.J.o.u.-a.e.-S., Science and Technology, *Parameters analysis for basic ant colony optimization algorithm in TSP.* 2014. **7**(4): p. 159-170.

[43]. Deng, W., J. Xu, and H. Zhao, *An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem.* IEEE access, 2019. **7**: p. 20281-20292.

[44]. Míča, O. *IMPACT OF PARAMETERS α, β AND ρ ON QUALITY OF SOLUTION TO TRAVELLING SALESMAN PROBLEM BY ANT COLONY OPTIMIZATION ALGORITHM.* in *Proceedings of the 10th CER Comparative European Research Conference-International Scientific Conference for Ph. D. students of EU countries*. 2018. Sciemcee Publishing.

[45]. Stützle, T., et al., *Parameter adaptation in ant colony optimization.* Autonomous search, 2011: p. 191-215.

[46]. Valdez, F., F. Moreno, and P. Melin, *A comparison of ACO, GA and SA for solving the TSP problem*, in *Hybrid intelligent systems in control, pattern recognition and medicine*. 2020, Springer. p. 181-189.

[47]. Olivas, F., F. Valdez, and O. Castillo. *A fuzzy system for parameter adaptation in ant colony optimization*. in *2014 IEEE Symposium on Swarm Intelligence*. 2014. IEEE.

[48]. Dahan, F., et al., *Dynamic Flying Ant Colony Optimization (DFACO) for Solving the Traveling Salesman Problem.* Sensors (Basel), 2019. **19**(8): p. 136-145.

[49]. Gülcü, Ş., et al., *A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem.* 2018. **22**(5): p. 1669-1685.

[50]. Bouchon-Meunier, B., M. Dotoli, and B. Maione, *On the choice of membership functions in a mamdani-type fuzzy controller.* 1996.

[51]. Nguyen, H.T., et al., *A measure of average sensitivity for fuzzy logics.* 1994. **2**(04): p. 361-375.

[52]. Mohsen, A.M., *Annealing Ant Colony Optimization with Mutation Operator for Solving TSP.* Comput Intell Neurosci, 2016. **2016**: p. 8932896.

[53]. Reinelt, G.J.O.j.o.c., *TSPLIB—A traveling salesman problem library.* 1991. **3**(4): p. 376-384.

[54]. Khan, I. and M.K. Maiti, *A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem.* Swarm and Evolutionary Computation, 2019. **44**: p. 428-438.

[55]. Dorigo, M. and T.J.H.o.m. Stützle, *Ant colony optimization: overview and recent advances.* International Series in Operations Research & Management Science, 2019: p. 311-351.

[56]. Dahan, F., et al., *Dynamic Flying Ant Colony Optimization (DFACO) for Solving the Traveling Salesman Problem.* Sensors (Basel), 2019. **19**(8): p. 1837.

[57]. Skinderowicz, R., *Improving Ant Colony Optimization efficiency for solving large TSP instances.* Applied Soft Computing, 2022. **120**: p. 108653.