Ant Colony Optimization for Traveling Salesman Problem with Modified Pheromone Update Formula

Rahil Parmar, Naitik Panchal, Dhruval Patel, and Uttam Chauhan

Abstract Traveling Salesman Problem is a combinatorial problem from which various other problems have been derived in the real-world application. It is a well-known NP-complete problem. Its instances are used in various fields around the globe. There have been various optimization techniques that are used to solve this problem. The Ant Colony Optimization (ACO) is an optimization method that is very useful in solving various artificial intelligence problems and obtaining the optimized solution. There have been methods proposed after its introduction in 1991. When using the traditional ACO pheromone update formula on the large dataset of Traveling Salesman Problem, one might get an optimal solution at the cost of a great amount of time. In this paper, we have proposed a modification in the basic Ant Colony Optimization pheromone update formula for discovering the optimized solution for the Traveling Salesman Problem using the probability from the pheromone value from succeeding nodes. This updated formula also helps in reducing the time to obtain the optimal solution as compared to the traditional formula.

Keywords Artificial intelligence · Ant colony optimization · Traveling salesman problem

1 Introduction

Artificial Intelligence (AI) can be described as intelligence in machines that can resemble human intelligence and can perform some tasks, which requires logical thinking to solve the problem. Some of the tasks can be categorized from everyday tasks like speech, translation, generation of language to expert level tasks like manufacturing planning, scientific analysis, etc. Now for solving these tasks, the AI system needs some dataset, which it can process and find the solution. These processes require searching and optimizing. For example, the AI system which is designed to solve the single-player tic tac toe game, two-player chess or to get the shortest path

R. Parmar (✉) · N. Panchal · D. Patel · U. Chauhan
Vishwakarma Government Engineering College, 382424 Ahmedabad, Gujarat, India

P. K. Singh et al. (eds.), *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security*, Lecture Notes in Networks and Systems 203,
https://doi.org/10.1007/978-981-16-0733-2_2

in the graph, the system needs to search the input dataset to achieve the required outcome. For searching in the dataset, the AI system uses various search algorithms like Depth First Search (DFS), Breadth First Search (BFS), Iterative Deepening DFS, bi-directional search or A* search, etc. [8].

The efficiency of these algorithms may depend on the size of the search state space generated from the given input. As the size of the obtained search space state becomes larger, these algorithms can take a very long time to generate the solution [2]. Besides, the solutions may become less accurate and less efficient. Heuristic methods are used for increasing the efficiency of these algorithms by optimizing the generated path to a solution. It can be done by finding such solutions that decrease the size of the search state space and the time required to achieve these results. Heuristic techniques often generate good-enough solutions, but it does not guarantee the optimal solution [4].

Optimization algorithms are important to the various fields in real life. Examples of the practical implementation of the optimization algorithm include train scheduling, telecommunication, shape formation, routing technique, drone pathfinding, and many more.

1.1 Meta-Heuristic Methods in AI

The term meta-heuristic is composed of two Greek words, the suffix meta means "more organized" and the heuristics mean "to find". A meta-heuristic procedure is used to find the best solutions from the set of feasible solutions. Meta-heuristic can be used in combinatorial optimization to generate a better solution with more efficiency than simpler heuristic methods. Combinatorial optimization is to find the near-optimum solution from the set of feasible solutions. The Traveling Salesman Problem is a good example of NP-hard problems in combinatorial optimization [4].

The Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Simulated Annealing are the fields where the meta-heuristic methods are used widely. Genetic Algorithms are the search-based optimization techniques that are based on the concept of natural selection and genetics. The Genetic Algorithms can be briefly summarized as follows [10]:

1. First, a node is selected at random from the given nodes.
2. Then the pool of possible solutions is evaluated from the current node.
3. These solutions then undergo various recombinations and mutations which produce the new children.
4. This process is repeated for various generations. The fitness value is assigned to each child solution obtained.
5. The node with a better solution is more likely to produce an optimized path.

This way the Genetic Algorithm is implemented on various problem domains.

The Simulated Annealing has been developed by taking inspiration from the metal annealing. The annealing method involves the heating and cooling of the metal to

change its physical property by changing its inner structure. As the metal cools, its new structure becomes fixed and retains its new property. Suppose S is a set of obtainable solutions. The initial start point is selected at random from the set of obtainable solutions. Now from the currently selected solution, the cost function to other solutions is calculated using the gradient descent which is, in this case is performed on the temperature. If the calculated cost function is reduced, then the current solution is replaced by the generated solution. Otherwise, the generated solution is rejected. This process is carried out until the optimal solution is obtained. This technique is used in various other fields like Machine Learning and Deep Learning to train the model to predict the result and learn to optimize.

Particle Swarm Intelligence is an algorithm that is derived from the social behavior of the flock of birds, schools of fish, and the ant colony [17]. This algorithm emulates the interaction between the members to share the information among them. The individual solution in these methods is considered as a particle. Each individual in PSO flies with the velocity that is dynamically adjusted according to its own experience and the experience of its companion [22]. Given the solution set S with the position of each solution in sample space, we can obtain the optimal solution using PSO. From the current position of the swarm, the fitness of the particle can be calculated by using the objective function. The solution with a better fitness value is selected, and the optimal solution is created. While creating the optimal solution, the previous best solutions are remembered for the backtracking.

This paper is further organized as follows: Sect. 2 explains the famous Traveling Salesman Problem, it also involves how to solve Traveling Salesman Problem using Ant Colony Optimization. Section 3 describes the variants in the Ant Colony Optimization. Following this, we have listed the current state-of-the-art techniques in Sect. 4. Next, we have presented modified formula along with the necessary pseudocode in Sect. 5. In Sect. 6, we have showcased our experimental results in the form of a graph, where we have compared the proposed approach with Ant Colony System. Finally, we present some concluding remarks in Sect. 7.

2 Traveling Salesman Problem (TSP)-NP-Hard

The TSP is a very well-known NP-hard problem in computer science. The TSP is an NP-Hard problem so it means that to solve this problem, there is no direct efficient way. If the solution of TSP is found, then it is possible to find the solution of the famous P versus NP problem. Besides, there are other applications of TSP, so we decided to apply Ant Colony Optimization (ACO) on TSP.

The problem can be described as there is a list of cities which are to be visited by the traveling salesman, the distance between the cities are given. The traveling salesman has to visit every city on the list and return to the original city, but he can only visit each city only once and after visiting a city he cannot re-visit the city. The main aim of the TSP is to find the shortest route to visit each city once. TSP

is a problem of combinatorial optimization [19]. The TSP can be defined by the Hamiltonian cycle problem.

In a directed or undirected graph, there may exist a path that visits each node in the graph exactly once this type of path is called the Hamiltonian path, when the Hamiltonian path forms a cycle it is called the Hamiltonian cycle. In TSP, we need to find the Hamiltonian cycle with the smallest cost. Many problems are similar or related to the original TSP such as Generalized TSP, Bottleneck TSP, Steiner TSP, etc. The generalized TSP is known as the SetTSP [1]. In the generalized TSP, there are sets of nodes of the graph and from each set, we have to visit at least one node and we need to visit each set to find the smallest cost Hamiltonian cycle. The TSP can be called the specialized case of the Generalized TSP in which each set contains exactly one city [12]. In the figure given below, each node represents the city in a graph. The ACO is implemented on the symmetric TSP here in this case. In symmetric TSP, the distance from city A to city B and from city B to city A is the same. So, it can be represented as $d_{AB} = d_{BA}$, where d is the distance between these two cities, which forms an undirected graph. In asymmetric TSP, the distance from city A to city B and from city B to city A can be different because the graph for asymmetric TSP is directed graph so the path between two cities may be in one direction only (Fig. 1).

The TSP is a problem of combinatorial optimization. It is observed that TSP having 20 or fewer numbers of cities can be solved using the specific techniques like dynamic programming or branch and bound method. It can provide the optimal solutions efficiently [20], but when the number of cities increases, the combinations to search the feasible solution also increases. So for a larger number of cities, AI search optimization techniques like Swarm Intelligence, Artificial Bee Colony (ABC), ACO, or genetic algorithm can be applied to solve the TSP efficiently.

TSP is directly applied in logistics and transportations. Additionally, TSP can be applied in many areas such as the production of ICs and PCBs, where it is used in drilling machines for decreasing the time of these processes. TSP is used in computing the DNA sequencing. In finding the shortest path between airports, the TSP can be

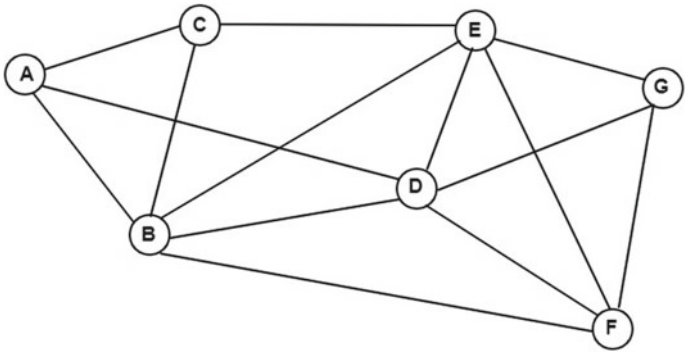

Fig. 1 Traveling salesman problem

applied. Moreover, The TSP can be applied to deliver the power to home using the fiber optics network designing.

There are many state-of-the-art methods to solve the TSP. These methods include some complete algorithms for solving the TSP, most of the algorithms are based on branch and cut methods. The Concorde TSP solver has solved almost all of the TSPLIB instances. The current largest solved TSPLIB instance includes 85,900 cities. Other state-of-the-art methods are the type of Stochastic Local Search (SLS) algorithms, these are Construction heuristic, Hybrid methods, Population-based methods. Population-based algorithms are mostly used as a basis for developing more efficient TSP algorithms. There are a few limitations of SLS algorithms. SLS algorithms can sometimes get into the Stagnation situation, it is difficult to know if the algorithm is in stagnation or not. It is also not guaranteed to find a solution.

2.1 TSP using ACO

ACO was developed from the foraging behavior of the real ants [5]. The first application of ACO that successfully had an advantage over Simulated Annealing and Genetic Algorithm approaches that were aimed to solve this type of dynamic problems. The efficient ACO algorithms adjust between search intensification and diversification.

But in ACO, there exist problems of stagnation situation and premature convergence. Furthermore, the convergence speed of ACO is slow, and as the size of the problem increases, these problems become more obvious. Hence, the ACO algorithm needs more improvement [18]. Initially, the ants move in a random direction in search of the food near their nest. The ants leave behind the special type of chemical known as pheromone due to ant's weak perception about the environment, to communicate to the other ants. The ant travels from its nest to the food source by finding the optimal path and leaving the traces of pheromone for the other ants to follow. As more amount of pheromone is accumulated on the path, the ants will start following the same path [9]. In the ACO algorithm, ants are the agents that search for the solution in the available solutions and then try to optimize the path to reach the solution. The biologists have described the foraging behavior of ants composed of three mechanisms. (1) Selection Mechanism: If more information is on the path, then the probability of choosing that path is higher. (2) Update Mechanism: The amount of pheromone increases with the number of ants and decreases with time. (3) Co-ordination Mechanism: The communication between the ants is carried out in coordination [21] (Fig. 2).

While traveling through the solutions, the (artificial) ants deposit the trail of (artificial) pheromone on the edges. The solution of the problem depends on the quality and distance of the solution of the previous solution [19]. After certain iteration, the ants travel through the same path and show the little deviation in the path, providing the optimized solution to the problem. The Ant System algorithm was originally a set of three algorithms based on their implementation. Those algorithms were Ant-Cycle,

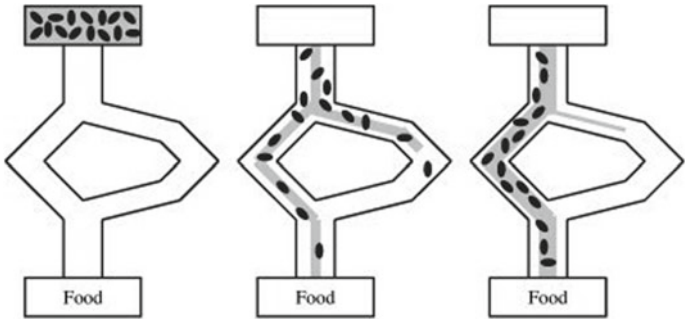

Fig. 2 Ant colony system

Ant-Density, and Ant-Quantity. The difference between these three was the updating of the pheromone at the nodes. In the Ant-Cycle algorithm, the pheromone update was done at the end of each ant's tour while in the latter two, the pheromone was updated at each subsequent step. In Ant-Cycle, the pheromone is updated depending on the length of the tour completed by the individual ant, while in the other two algorithms the pheromone is updated according to distance between two cities [3].

Given the weighted graph $G = (V, E)$ of n cities, the TSP can be stated as the smallest tour length of the agent based on the Hamiltonian distance. The Hamiltonian distance between the two cities city i and city j can be given by d_{ij}. The ant k at the city i chooses the city j through probability p_{ij}^k, which is calculated as the function of the city distance and the pheromone amount at that city [15]. The probability for choosing the next city j can be given as:

$$P_{ij}^{(t)} = \begin{cases} \frac{\left(\tau_{ij}^{\alpha}\right)\left(\eta_{ij}^{\beta}\right)}{\sum \tau_{ij}^{\alpha}\left(\eta_{ij}^{\beta}\right)}, & \text{if } j \notin \text{tabu list} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Here, τ is the pheromone value at city j and η is the visibility of the city j from city i (i.e., the inverse of the distance between the cities). The pheromone value τ is updated at every edge according to Eq. (2).

$$\tau_{ij}(t+1) = \rho\left(\tau_{ij}(t)\right) + \tau_{ij}^{k}(t) \tag{2}$$

where ρ is the rate of evaporation and its range is in $0 < \rho \leq 1$. Here the evaporation rate ρ is used for avoiding the unlimited deposition of the pheromone. It allows the ants to improvise the path rather than being stuck up in the local best option. Here $\Delta\tau_{ij}$ is the pheromone the previously visited ants accumulated on the node and it is given as follow:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k}, & \text{if } j \text{ is visited} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where L^k is the distance traveled by the ant so far. Q is the positive constant. From Eq. (3), it can be seen that better the tour of the ant, the more amount of pheromone is deposited by the ant at the node. After some iterations, the path chosen by many ants will contain more amount of pheromone and therefore it has more probability of being chosen by the ants in upcoming iterations.

3 Variants in ACO

Originally, the ACO algorithm was known as Ant System (AS) which was presented by Dorigo et al. in the early 90 s. Later there were some improvements introduced over the AS. These improvements include the Elitist Ant System (EAS), Rank-Based Ant System (RBAS), MAX–MIN Ant System(MMAS), and Ant Colony System (ACS). In these improvements, the methods for solution generation and pheromone evaporation were similar but the pheromone update and pheromone trail management were improved. In all the systems, the evaporation factor is used which evaporates the pheromone at the nodes at the predetermined rate, which in turn helps the algorithm to explore new paths and enables it to escape the premature convergence to the optimal solution. Over the years, many improvements are added to the original algorithm.

Elitist AS: It was the first improvement over the AS. In EAS, the best optimal tour traveled obtained since the start is dispensed with additional reinforcements. This is done by pheromone sublimated by the additional ant called best-so-far ant and the tour is called tbs (best-so-far-tour) [6].

Rank Based AS: The rank-based Ant System improvement was done by Bullnheimer et al. In RBAS each individual ant deposits a certain amount of pheromone which decreases with its range and the ant with the best-so-far tour will deposit the colossal amount of pheromone in each iteration. As a result, RBAS provides a moderately better result than EAS and significantly better results than AS [6].

MMAS: The MIN–MAX Ant System was introduced by Stützle and Hoos. The MMAS has four modifications over the AS. The first modification is that either only the ant which has produced the best tour in the current iteration, which is called iteration-best ant or the best-so-far ant is allowed to change or add the pheromone to the nodes. This may direct result in a stagnation situation. In this situation, all the ants follow only one path or tour because it has an excessive amount of pheromone deposited. This tour may be good or suboptimal. The stagnation situation may halt the exploration of another path that can be optimal. Hence to counteract the stagnation situation, the second modification was implemented in MMAS [6].

As per the second modification, it limits the pheromone trails' possible range in the interval $[\tau_{\min}, \tau_{\max}]$. The third modification was to initialize the pheromone value to the upper limit. Along with a compact pheromone evaporation rate, the exploration rate of other tours from the beginning of the search is increased. The fourth modification was that the pheromone values are to be reinitialized whenever

the system is stuck in a stagnation situation or when no enhanced tour is obtained for some consecutive iterations.

Based on experimental results, it is seen that for TSP having a small number of cities, the iteration-best ant should modify the pheromone values and for TSP having a larger number of cities pheromone on the best-so-far tour is updated only.

ACS: Ant Colony System differs from AS in three respects. Primarily, ACS uses more belligerent action choice rule than AS. Subsequently, global pheromone update and local pheromone update. In tour construction, ACS ants use the pseudorandom-proportional action choice rule, it is given by:

$$j = \begin{cases} -\text{argmax}_{L \in N_i^k} \tau_{il} \left(\eta_{il}^{\beta} \right), & \text{if } q \leq q_0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where q is randomly distributed variable in the range [0, 1]. q_0 ($0 \leq q_0 \leq 1$) is a parameter, J is calculated by ($\alpha = 1$). If the probability is q_0, then the ant selects the best possible solution learned from previous pheromone trails information. Here the ant makes use of the previously acquired knowledge. The tour is constructed based on best-so-far tours or to reconnoiter other tours. If the probability is ($1–q_0$) the biased exploration is done. In global pheromone update, only the ant with a best-so-far tour is allowed to update the pheromone over the best-so-far tour. In the local pheromone update, it is applied immediately after each arc is crossed. After an arc is visited, the pheromone value over that arc is reduced, so the exploration rate of other arcs is increased and it does not result in the stagnation situation.

4 State-of-the-art

Since its development, various research papers and the literature have been published for the improvements in the ACO. The ACO has been applied to various domains in the different forms and with the hybridization with various other techniques, the improvements have been made according to the need. In the latest ACO research, authors mainly focus on the areas of integrating various algorithms with the ACO and the application of ACO in various other disciplines.

4.1 Hybrid Method Based on ACO and 3-Opt Algorithm

One of the most common problems faced during the implementation of the ACO is premature stagnation. The stagnation can be defined as the condition in which all the ants traverse through the same track and construct the same repercussion, again and again considering there is no other optimal path. To overcome this problem, the hybrid algorithm which contains multiple colonies of ants shares the global best tours

occasionally to guide to the better solution. If the colony is stuck in the stagnation condition, the other colonies extricate it from the condition. Thus, this algorithm provides better and more robust solutions. This algorithm requires the colonies to be executed independently at the same time to find the optimal tour [11]. This algorithm considers the tour completed by ant and finds the optimal solution. But it fails to look at the tour which is yet to be performed. For a very large dataset, we might need large number of iteration for the maximum number of tours to be exploited.

4.2 Annealing Elitist Ant System with Mutation Operator

In this algorithm proposed by Abdulqader M. Mohsen, the ant has the two options for selecting the next city to be visited. Given n ants which are to be traveled through m cities, the ants will be distributed randomly across the cities. Initially, the pheromone level at all the cities is initialized to a small positive integer. At every iteration, an ant will have to choose either mutation operation or simulated annealing, based on the multifariousness of elitist ant system, to improve the performance of the algorithm. Mutation Operator is an algorithm in which each ant is provided with the chance to amend according to the predetermined probability. This operator helps the algorithm to survey different scope in the search space. If the diversity is greater than some value, the algorithm needs intensification which can be achieved through annealing on the ratio of solution pool. If the diversity is less than some value, it means the algorithm is losing its diversity and there is a probability to be stuck in local minima. Therefore, the algorithm needs to increase diversity, which can be attained by applying the mutation operator [16]. The variousness in the fitness of the ants in the algorithm can be obtained by the Euclidean Distance (ED) which is given as:

$$\mathrm{ED} = \frac{\overline{d} - d_{\min}}{d_{\max} - d_{\min}} \tag{5}$$

Here,

Here $\overline{d}$ is the mean of the fitness value of the best ant and fitness value of other prevailing ants in the solution. $d_{\min}$ is the distances of the worst ant fitness and $d_{\max}$ is the second-best ant fitness from the best ant respectively. In this algorithm, we need to tune diversity parameter in order to decide whether the algorithm requires diversification or intensification based on the dataset provided. This makes the algorithm very tedious to execute on different datasets as we need to reset the value for different datasets.

4.3 Greedy–Levy Flight ACO

There is always a perplexity in the reinforcement learning of further exploitation and exploration. To address this dilemma, Greedy–Levy Flight ACO was developed. The Greedy–Levy ACO comprises two algorithms, Epsilon Greedy Algorithm and Levy Flight Algorithm. The Epsilon-Greedy policy is technique of exploration which is used in ACS (Ant Colony System) algorithm [14]. The exploitation in Epsilon-Greedy policy is done with the probability of epsilon while selecting the best node available and the exploration is done with the probability of 1-epsilon. In case of 1-epsilon probability, the Levy Flight technique is used to improve the results. In this algorithm, initially a random number P is generated such that $0 < P < 1$. If the $P \leq \varepsilon$, the candidate solution with maximum probability is selected. If $P > \varepsilon$, then a candidate is selected randomly by using the Levy Flight algorithm.

The Greedy–Levy ACO tries to achieve the balance between local search and global search for generating more optimal results which is very crucial in increasing efficiency of the algorithm and it is done by implementing the epsilon-greedy method and Levy Flight technique [13]. The probability of choosing the next node in the Greedy–Levy algorithm can be given as follows:

$$P_{ij}^{(t)} = \begin{cases} \text{argmax}\left\{\left(\tau_{ij}^{\alpha}\right)\left(\eta_{ij}^{\beta}\right)\right\}, & \text{if } P \leq \in \\ 1 - A \times \frac{1-P_{\text{levy}}}{1-P_{\text{threshold}}} \times \left(1 - \frac{\left(\tau_{ij}^{\alpha}\right)\left(\eta_{ij}^{\beta}\right)}{\sum \tau_{ij}^{\alpha}\left(\eta_{ij}^{\beta}\right)}\right) & \text{if } P > \in, \text{ if } P_{\text{levy}} \geq P_{\text{threshold}} \\ \frac{\left(\tau_{ij}^{\alpha}\right)\left(\eta_{ij}^{\beta}\right)}{\sum \tau_{ij}^{\alpha}\left(\eta_{ij}^{\beta}\right)}, & \text{else} \end{cases} \tag{6}$$

Here, A = altering ration of Levy Flight.

P_{levy} = Probability of turning on/off levy Flight altering, $0 < P_{\text{levy}} < 1$.

$P_{\text{threshold}}$ = Parameter for Levy Flight threshold, $0 < P_{\text{threshold}} < 1$.

The Greedy–Levy algorithm improves the program by tuning the exploration and exploitation of the ants, and hence, it requires more iteration to get to the optimal solution.

5 Proposed Solution

Ant Colony Optimization is one of the most popular search methods among artificial intelligence based on the real behavior of ants. There have been various modifications to the algorithm since its formulation by Dorigo. In this paper, we propose a modification in the basic ACO's pheromone update formula which improves the convergence speed of ACO in comparison to the basic ACO model and improves the time required to get the optimal solution. When an ant travels from city i to city j,

the pheromone value of the city j is updated according to the distance traveled by that ant so far. We propose to add a γ amount of pheromone of the next city k from the city j, where city k has the maximum amount of pheromone value from the city j. Hence, the pheromone value at the city j will be updated by the Eq. (5) given below:

$$\tau_{ij}(t+1) = \rho\big(\tau_{ij}(t)\big) + \Delta\tau_{ij}^{k}(t) + \gamma\big(\tau_{jk}(t)\big) \tag{7}$$

Here τ_{jk} is the pheromone value at the city k from the city j and *gamma* is the factor which decides the amount of pheromone value to be added. The city k is selected based on the following equation:

$$k = \text{argmax}\big\{\tau_{jl}\big\} \tag{8}$$

The city k is determined based on the next city from city j which has the maximum pheromone value. By adding this modification to the original algorithm, the convergence speed increases, and the amount of time required to reach the optimal solution decreases greatly. The pseudocode for the ACO is given below:

Algorithm 1: Pseudocode for ACO

```
Distribute ants at random cities;
for i in number of ants do
    while all the cities are not visited do
        Add city to tabu list;
        Calculate the probability for next city;
        Calculate pheromone value;
        increment k by 1;
    end
end
Increment i by 1;
Calculate optimized path;
```

The reason for this convergence speed and the more optimal solution lies in the future path of the solution. In the proposed modification, we are adding a certain portion of the pheromone from the next nearest solution which adds the probability of the next succeeding path to the current solution. As the number of iteration increases and the number of ants visiting the nodes increases, the pheromone value from the succeeding optimal path is accumulated at the node, which increases the probability of choosing the node with the global best solution.

While in the original Ant Colony System, the ant chooses the next node based on the local optimal solution. Moreover, as the number of ants increases beyond some point, the pheromone value from various nodes is repeatedly accumulated at some nodes, which in turn leads the ants to choose the path with maximum pheromone amount which is the result of the various repeated pheromone from the succeeding nodes.

6 Experiments and Results

In this section, we provide the experiments and their results obtained on the execution the algorithm. The algorithm has been developed in python and has been implemented on the Windows 10 64-bit Operating System, Intel Core i7 2.5 GHz processor and 8 GB RAM. We adjusted the parameters for the comparison of the original algorithm with the proposed algorithm. Here we have varied the number of cities and the number of ants to present the difference in the optimal solution and time provided by the original and the proposed algorithm. In the implementation of the algorithm, we have used the real data of the TSPlib benchmark libraries which is available on the web: https://comopt.ifi.uniheidelberg.de/software/TSPLIB95/XML-TSPLIB/instances/.

According to the survey paper published by Dorigo and Stutzle, in 2019, minmax ACO and Ant Colony System are still the state-of-the-art techniques [7].

Hence we've compared the results obtained with Ant Colony System and used it as a benchmark. To verify the performance of the modified formula, it has been applied to various TSP instances. The algorithm provides the promising results when used with the modified pheromone update formula. We have implemented and compared the results obtained by the ACS algorithm with the traditional pheromone update formula and the modified pheromone update formula. For the experimentation, we have initialized the variables as follows: $\alpha = 1$, $\beta = 2$, $\rho = 0.5$ and $\gamma = 0.4$ (Table 1; Figs. 3 and 4).

It can be perceived from the results that the proposed modification generates an optimized path with a fewer number of ants and less time than the original ACO. It is due to the consideration of the pheromone value of the next node from the traveled node which adds the pheromone value of the succeeding node with an optimal solution from the graph. With the increasing number of iteration, the pheromone value from the succeeding nodes gets accumulated at the node which provides the optimal solution according to the global optimal solution. While in traditional ACO, the next node is selected based on the local best solution. Because of this, the next node

Table 1 Statistical comparison with benchmark algorithm

TSP instances	No. of cities	ACS with original formula (s)	ACS with proposed formula (s)	Percentage change (%)
berlin52	52	8.13	6.29	22.63
eil76	76	17.01	10.3	39.45
kroA100	100	37.27	22.0	40.97
pr124	124	89.41	41.15	53.98
rat195	195	265.34	156.62	40.97
a280	280	813.57	472.56	41.92
rd400	400	2669.75	1464.34	45.15
ali535	535	6385.28	3184.64	50.12

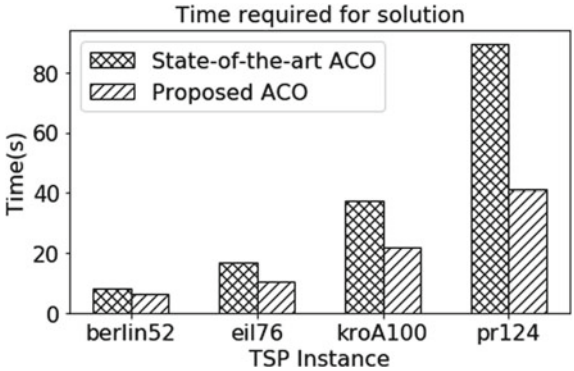

Fig. 3 Comparison of algorithm for less than 150 cities

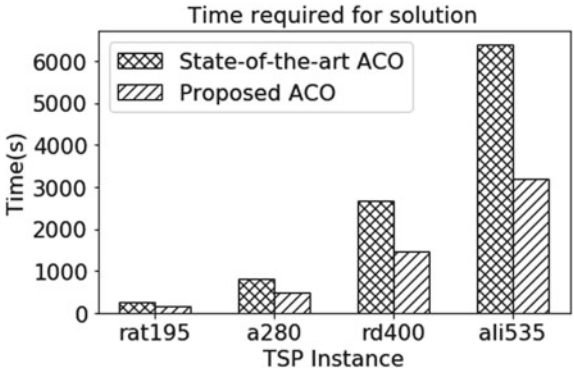

Fig. 4 Comparison of algorithm for more than 150 cities

selected by the ants contains the pheromone value from the optimal node succeeding that node and ant select the node according to the global best solution. As a result, the optimal solution obtained in the proposed modification is better than the traditional ACO algorithm. Also, the experimental results vary when we change the parameters used in the algorithm. Here, the rate of evaporation also plays an important role in providing the optimal solution.

7 Conclusion

In this paper, we are proposing the modification in the pheromone update formula of the ACO. The new pheromone update formula updates the pheromone value of current node with a certain amount of the pheromone value from the next node with maximum pheromone from the current node. It can be deduced that the proposed

method performs less computation with less number of ants used for a large number of cities. Hence, the time for finding the optimal solution decreases notably by 41.9%. The advantage of using the proposed method is that with a low number of ants and iterations, the optimal solution is found in a shorter time. Hence, it can be used in solving the problem which may arise in practical applications that are combinatorial. Also, it can be implemented in all the new methods proposed to find the optimal solution.

References

1. Generalized tsp.https://en.wikipedia.org/wiki/Set_TSP_problem. Accessed 14 May 2020
2. State space search. https://en.wikipedia.org/wiki/State_space_search. Accessed 15 Apr 2020
3. Asmar D, Elshamli A, Areibi S (2005) A comparative assessment of aco algorithms within a tsp environment. Dyn Contin Discr Impul Syst Series B Appl Algorithms 1:462–467
4. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3):268–308
5. Chen H, Tan G, Qian G, Chen R (2018) Ant colony optimization with tabu table to solve tsp problem. In: 2018 37th Chinese Control Conference (CCC). IEEE, pp 2523–2527
6. Dorigo M, and Thomas S (2004) Ant colony optimization algorithms for the traveling salesman problem
7. Dorigo M, Thomas S (2019) Ant colony optimization: overview and recent advances. In: Handbook of metaheuristics. Springer, pp 311–351
8. Knight K, Rich E, Nair SB (2009) Problems and search. In: Artificial Intelligence, 3rd edn. Tata McGraw-Hill
9. Gao W (2020) New ant colony optimization algorithm for the traveling salesman problem. Int J Computat Intell Syst 13(1):44–55
10. David EG (1989) Genetic algorithms in search. In: Optimization, and Machine Learning
11. Şaban G, Mostafa M, Ömer KB, Halife K (2018) A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem. Soft Comput 22(5):1669–1685
12. Wu J, Ouyang A (2012) A hybrid algorithm of aco and delete-cross method for tsp. In: 2012 International Conference on Industrial Control and Electronics Engineering. IEEE, pp 1694–1696
13. Liu Y, Cao B (2020) A novel ant colony optimization algorithm with levy flight. IEEE Access 8:67205–67213
14. Liu Y, Cao B, Li H (2020) Improving ant colony optimization algorithm with epsilon greedy and levy flight. JSP 24(25):54
15. Mavrovouniotis M, Yang S (2013) Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. App Soft Comput 13(10):4023–4037
16. Mohsen AM (2016) Annealing ant colony optimization with mutation operator for solving tsp. In: Computational intelligence and neuroscience
17. Ester ME (2017) Ant colony optimization for predicting gene interactions from expression data
18. Raghavendra BV (2015) Solving traveling salesmen problem using ant colony optimization algorithm. J Appl Comput Math JACM 4(6):260
19. Stützle T, Dorigo M et al (1999) Aco algorithms for the traveling salesman problem. Evolut Algorithms Eng Comput Sci 4:163–183
20. Supaporn S, Deacha P (2012) Solving traveling salesman problems via artificial intelligent search techniques. In: Proceedings of the 11th WSEAS international conference on artificial intelligence, knowledge engineering and data bases. World Scientific and Engineering Academy and Society (WSEAS), pp 137–141

21. Wang J, Yang X (2016) Application of improved ant colony algorithm on travelling salesman problem. In: 28th Chinese control and decision conference (CCDC)
22. Liu Y, Hou Z, Jiang C (2005) Unit commitment by binary particle swarm optimization. In *Proceedings of the 7th WSEAS International Conference on Mathematical Methods and Computational Techniques In Electrical Engineering*, pages 372–377. Citeseer, 2005.