International Journal of Computational Intelligence Systems
Vol. **13(1)**, 2020, *pp.* 44–55
DOI: https://doi.org/10.2991/ijcis.d.200117.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

New Ant Colony Optimization Algorithm for the Traveling Salesman Problem

Wei Gao[*]

College of Civil and Transportation Engineering, Hohai University, Nanjing, Jiangsu 210098, PR China

ARTICLE INFO

Article History
Received 09 Oct 2019
Accepted 27 Dec 2019

Keywords
Computational intelligence optimization
New ant colony optimization algorithm
Meeting strategy
Performance
Traveling salesman problem

ABSTRACT

As one suitable optimization method implementing computational intelligence, ant colony optimization (ACO) can be used to solve the traveling salesman problem (TSP). However, traditional ACO has many shortcomings, including slow convergence and low efficiency. By enlarging the ants' search space and diversifying the potential solutions, a new ACO algorithm is proposed. In this new algorithm, to diversify the solution space, a strategy of combining pairs of searching ants is used. Additionally, to reduce the influence of having a limited number of meeting ants, a threshold constant is introduced. Based on applying the algorithm to 20 typical TSPs, the performance of the new algorithm is verified to be good. Moreover, by comparison with 16 state-of-the-art algorithms, the results show that the proposed new algorithm is a highly suitable method to solve the TSP, and its performance is better than those of most algorithms. Finally, by solving eight TSPs, the good performance of the new algorithm has been analyzed more comprehensively by comparison with that of the typical traditional ACO. The results show that the new algorithm can attain a better solution with higher accuracy and less effort.

1. INTRODUCTION

As a computational intelligence optimization method, ant colony optimization (ACO) [1] is a swarm intelligence algorithm inspired by the foraging behavior of real ants. The first proposed algorithm of ACO, called the ant system (AS) [2], was developed by three Italian scholars (Dorigo, Colorni, and Maniezzo) in 1991. As the original version of various ACO algorithms, the AS has several obvious merits, including its robustness and its status as a constructive greedy heuristic [1]. Therefore, the AS has been a heavily researched topic from its creation and has been used to solve difficult combinatorial optimization problems, such as the traveling salesman problem (TSP). However, as a heuristic algorithm, the AS has several shortcomings [1], including slow convergence and difficulty in expanding the search space. To solve the TSP well, several scholars have proposed many corrective algorithms for ACOs, such as the elitist AS [3], the ant colony system [4], the rank-based AS [5], the max–min AS [6], the novel max–min AS [7], the adaptive dynamic AS [8], the moderate AS [9], an improved ACO algorithm [10], the cooperative genetic AS [11], a hybrid method of ant colony optimization and the genetic algorithm (ACO-GA) [12], a hybrid method of ant colony optimization and the cuckoo search algorithm (ACO-CS) [12], a hybrid max–min ant system integrated with an inequality constraint on four vertices and three lines (HMMAS) [13], a modified AS [14], a nearest neighbor ant colony system (NNACS) [15], a hybrid elitist-ant system (HEAS) [16], a hybrid method based on ant colony optimization and the 3-Opt algorithm (ACO-3Opt) [17], and a parallel ACO algorithm based on a quantum dynamic mechanism [18], and so on. The details of these algorithms are as follows. The elitist AS [3] introduced the idea of the elitist strategy in the context of the AS to give extra emphasis to the best path found so far to improve the global searching of AS. However, the elitist AS cannot provide a good solution to the slow convergence of AS. The ant colony system [4] was built on the AS to improve efficiency by using a local pheromone update in addition to the pheromone update applied at the end of the construction process. On the other hand, it cannot greatly decrease the probability of being trapped in a local minimum. To avoid premature convergence, the rank-based AS [5] introduced a ranking strategy into AS by weighting the contribution of an ant to the trail level update. However, this system does not provide a good solution to the slow convergence of AS. The max–min AS [6] improved the pheromone update method of AS by only allowing the best ant to update the pheromone trails and by limiting the value of pheromone on the paths to avoid premature convergence of the search. Nevertheless, this system also does not solve the problem of slow convergence, and more parameters need to be determined. To select suitable parameters of the max–min AS, the novel max–min AS [7] used the objective function value to set the pheromone trail value. However, its convergence rate needs to be improved. The adaptive dynamic AS [8] improved AS by modifying the pheromone updating rule and the transition rule with evenness of solution, interesting and acceleration to avoid premature convergence. However, this does not solve the problem of slow convergence. Inspired by adaptive behavior, the moderate AS [9] was developed with a new transition rule that can enhance the capability of AS to find

[*]*Email: wgaowh@163.com*

new solutions to avoid premature convergence, but its convergence rate needs to be improved. An improved version of the ACO algorithm [10] was proposed based on a candidate list strategy for rapid convergence speed and a dynamic updating rule for heuristic parameters using entropy to improve performance. However, this algorithm is very complex, and its performance cannot be verified well. The cooperative genetic AS [11] combined both the genetic algorithm (GA) and ACO in a cooperative manner to improve performance: in this system, the mutual information exchanged between two algorithms at the end of the current iteration. However, this algorithm is also very complex, and more parameters are required. In a hybrid method of ACO and GA (ACO-GA) [12], the GA utilizes all effective paths of ACO and then identifies an effective and efficient way to obtain a better solution in the search space. Moreover, similar to ACO-GA, in the hybrid method of ACO and the CS algorithm (ACO-CS) [12], the CS was used in ACO to search for a better solution. Nevertheless, the above two hybrid algorithms are very complex, and more verification of their performance is needed. In the hybrid max–min AS integrated with an inequality constraint on four vertices and three lines (HMMAS) [13], the MMA was used to search for an approximate result, and with the approximate one, the inequality constraint on four vertices and three lines was used in a local search to obtain a better approximation. However, the performance of this algorithm is not very good, and it cannot be used for large-scale problems. For the modified AS [14], the adaptive tour construction and adaptive pheromone updating strategies are embedded into the AS to achieve better solutions. Nonetheless, deep verification of it is lacking, and the convergence rate should be improved. To reduce computing time and without sacrificing the optimality properties of the solutions, a NNACS [15] was proposed by implementing enhanced pheromone updating rules in the ant colony system for which a large number of inefficient solutions are eliminated at the outset on the basis of proximity-based neighborhoods. However, currently, the NNACS only focuses on problems of small to moderate sizes. The HEAS [16] introduced an external memory into an elitist-AS, called an elite pool, which contains high-quality and diverse solutions to maintain a balance between diversity and quality of the search to enhance the performance of the algorithm. However, this algorithm is very complex, and more parameters are required. In the hybrid ACO-3Opt [17], the 3-Opt algorithm was combined with ACO to avoid local minima, and it can improve the quality of the obtained solutions and the robustness of the algorithm. However, the ACO-3Opt is very complex, and its performance is affected by many factors. Finally, for the parallel ACO algorithm based on a quantum dynamic mechanism [18], to improve the performance of ACO, the improved 3-opt operator was used to provide superior local search ability, and several antibody diversification schemes were incorporated to improve the balance between exploitation and exploration. However, this algorithm is very complex, and finding a suitable parameter setting for it is hard.

Because the TSP is a well-known NP-hard combinatorial optimization problem that is computationally difficult, in addition to ACO, many other new metaheuristic optimization algorithms have been applied to solve it, such as the quantum heuristic algorithm (QHA) [19], the discrete artificial bee colony algorithm with a neighborhood operator (DABC-NO) [20], the shrinking blob algorithm (SBA) [21], the discrete cuckoo search algorithm (DCSA) [22], the random-key cuckoo search (RKCS) [23], the African buffalo optimization (ABO) [24], the discrete bat algorithm (DBA) [25], the fruit fly optimization algorithm (FFOA) [26], a hybrid algorithm using a GA and a multiagent reinforcement learning heuristic (GA-MRLH) [27], the artificial atom algorithm (AAA) [28], the greedy flower pollination algorithm (GFPA) [29], the imperial competitive algorithm (ICA) [30], the black hole algorithm (BHA) [31], the simulated annealing-based symbiotic organisms search optimization algorithm (SA-SOSOA) [32], the discrete symbiotic organisms search algorithm (DSOSA) [33], the hybrid discrete artificial bee colony algorithm with a threshold acceptance criterion (DABC-TAC) [34], a minimum spanning tree-based heuristic (MSTH) [35], a genetic algorithm with local operators (GAL) [36], a new hybrid optimization algorithm based on wolf pack search and local search (WPS-LS) [37], discrete spider monkey optimization (DSMP) [38], discrete pigeon-inspired optimization (DPIO) [39], and the parthenogenetic algorithm (PGA) [40], and so on. For those algorithms, many are newly proposed metaheuristic algorithms, such as QHA, SBA, DCSA, RKCS, ABO, DBA, FFOA, AAA, GFPA, ICA, BHA, DSOSA, MSTH, DSMP, DPIO, and PGA. Most of them have simple structure and are easy to be applied. Using those new metaheuristic algorithms, the results for solving TSPs are perfect in terms of the solution quality and speed for most algorithms, except GFPA which mainly considers the robustness of the solutions. However, the theoretical bases of those new metaheuristic algorithms are all lacking. Moreover, the verifications are not enough and more experiments should be conducted. For example, only the small-scale problems have been solved by some algorithms (including SBA, RKCS, ABO, FFOA, AAA, ICA, BHA, MSTH, and DSMP), and only large-scale problems have been solved by some algorithms (including DPIO and PGA). Furthermore, some algorithms have their specific shortcomings. For example, for QHA [19], an open question still remains as to whether the quadratic or faster speedup can still be achieved when employing structural or geometrical information on city-pair costs, and the performance of GFPA [29] will decrease quickly as the number of cities increases. For the hybrid algorithms, DABC-NO [20] has combined neighborhood operator into newly proposed metaheuristic algorithm (discrete artificial bee colony algorithm) to improve solution quality for TSP, and results are perfect in terms of the solution quality and speed. However, only the small-scale problems have been solved by DABC-NO. The GA-MRLH [27] is the combination of genetic algorithm and a multiagent reinforcement learning heuristic, and its results for solving TSPs are good in terms of the solution quality and speed. However, the algorithm is very complex and more parameters are needed. The SA-SOSOA [32] is a new hybrid algorithm based on the simulated annealing and symbiotic organisms search optimization algorithm, and its results for solving TSPs are good in terms of the solution quality and speed. However, its theoretical basis lacks and more experiments should be conducted. DABC-TAC [34] is a hybrid discrete ABC, which uses acceptance criterion of threshold accepting method, and its results for solving TSPs are also good in terms of the solution quality and speed. However, this algorithm is very complex and more parameters are needed. GAL [36] is a hybrid algorithm of genetic algorithm with local operators and its results for solving TSPs are good in terms of the solution quality and speed too. However, the algorithm is also very complex and more parameters are needed. At last, WPS-LS [37] is a new hybrid algorithm based on wolf pack search and local search, and its results for solving TSPs are good in terms of the solution quality and robustness of the solutions. However, this algorithm is very complex and only the small-scale problems have been solved.

As mentioned previously, ACO was introduced by means of a proof-of-concept application to the TSP. Therefore, ACO has been applied to many combinatorial optimization problems [41]. First, classical problems other than the TSP, such as assignment problems, scheduling problems, graph coloring, the maximum clique problem, and vehicle routing problems, were addressed. More recent applications include electrical engineering problems, clustering problems, and civil engineering problems [42].

Intuitively, the most important theme in improvement algorithms for ACO is the balance between intensification and diversification. However, excessive emphasis on intensification can make ants converge to a local optimum, while excessive emphasis on diversification can lead to an unstable state [43]. To overcome the shortcomings of traditional ACO methods, a meeting strategy has been introduced to favor search diversification and maintain intensification at an appropriate level. In other words, this new strategy is to adjust the trade-off between intensification and diversification. Thus, a new ACO algorithm is proposed. To verify the new algorithm, the results of the new ACO algorithm for 20 TSPs are compared to the best known solutions and the results of 16 state-of-the-art algorithms. Furthermore, the computing effectiveness and efficiency of the new algorithm are compared with those of typical traditional ACO when the algorithm is used to solve eight generally used TSPs.

The paper is organized as follows. After this introduction, the typical traditional ACO is introduced in Section 2. The details of the proposed new ACO algorithm are given in Section 3. Section 4 presents the experimental results obtained from the new ACO algorithm and compares it to certain state-of-the-art heuristic methods. Section 5 contains the statistical analysis of two algorithms (traditional ACO and the new ACO algorithm). Finally, the study's conclusions are presented in Section 6.

2. TRADITIONAL ACO

By application to the TSP, the main steps of traditional ACO are introduced as follows. First, when $t = 0$, the ants, of which there are m, are randomly positioned at different cities, of which there are n. At this time, the intensity of pheromone (τ) on all paths is the same; this state can be described as

$$\tau = c \tag{1}$$

where c is a constant.

The next step for each ant is selecting another city that it has not yet visited using some rules. There are two main rules followed by the ants, which are

1. The pheromone intensity on the path between cities i and j at time t is $\tau_{ij}(t)$, which is information provided by the algorithm.
2. The heuristic information controlling the move from city i to j is η_{ij}, which can be determined by a heuristic algorithm according to the problem to be solved. Generally, it can be given as

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{2}$$

where d_{ij} is the length of the path between cities i and j.

By using the two rules, at time t, the ant k at city i selects the next city j, which it has not yet visited, according to the following probability:

$$P_{ij}^{k}(t) = \frac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}}{\sum\limits_{j \in N_i^k} \tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}}, \quad if \;\; j \in N_i^k \tag{3}$$

where α and β are two parameters that determine the relative influence of the pheromone trail and the heuristic information, and N_i^k is the feasible neighborhood of ant k when it is at city i, that is, the set of cities that ant k has not visited yet.

However, to avoid visiting a city more than one time, for each ant k, there exists one tabu list set *tabu* (k), which records the cities visited by this ant.

After the ant completes one cycle (visiting all cities), the pheromone intensity on all paths should be updated. That is, after the paths of the ants have been built, the pheromones for all ants are updated using the following equation:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{4}$$

where ρ is the residual ratio of the pheromone. However, to avoid infinite accumulation of the pheromone, ρ must be less than 1. $\Delta\tau_{ij}^k$ is the increase of the trail level on edge (i,j) caused by ant k. Depending on the problem, there are three descriptions of $\Delta\tau_{ij}^k$, as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{T^k}, & \textit{if ant k travels on edge } (i,j) \\ 0, & \textit{otherwise} \end{cases} \tag{5}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{d_{ij}}, & \textit{if ant k travels on edge } (i,j) \\ 0, & \textit{otherwise} \end{cases} \tag{6}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} Q, & \textit{if ant k travels on edge } (i,j) \\ 0, & \textit{otherwise} \end{cases} \tag{7}$$

where Q is the quantity of pheromone laid by an ant per tour and T^k is the length of the tour that ant k has found.

In the three descriptions above, global information is used for the first, while local information is used for the other two descriptions. Generally, the first description is used.

The procedure of traditional ACO is a kind of iterative process, and its pseudocode is as follows.

```
Begin
  Initialization of parameters: $\alpha, \beta, \rho$, Q, c, m and n
  $\tau_{ij} = c$, $\Delta\tau_{ij} = 0$, $\eta_{ij} = 1/d_{ij}$ are determined by a heuristic algorithm, $tabu_k = \varphi$
  For every cycle, do:
   Random assignment of ants to cities
   While $tabu_k$ is not full, do:
    For every ant k, do:
   Select the next city by probability $p^k_{[tabu_{[k][index-1]}][j]}$ according to Eq. (3)
   Move the ant to the selected city
   Save the selected city in $tabu_k$
      For every ant k, do:
      Compute the length of the tour
      Save the shortest path
      For every path, do:
      Calculate $\Delta\tau^k_{ij}$ (index) according to Eqs. (5–7)
       For every path, do:
       Update pheromone $\tau_{ij}$ according to Eq. (4)
       $\Delta\tau_{ij} = 0$
      $tabu_k = \varphi$(empty the visited list)
     End cycle
   Output the optimal path and its length
End
```

By using *C*++, the traditional ACO can be implemented in this study.

3. NEW ACO ALGORITHM

The new ACO algorithm is described in detail as follows for application to the TSP. In the traditional ACO, for each ant k, at the beginning, the number of cities in the set *tabu*(k) is small; thus, the ant selects the next cities with high flexibility. In contrast, at the end of the search process, the city number in the set *tabu*(k) is large, and the ant selects the next cities without any diversification. Thus, the ants will move along certain paths and lose the ability to explore other potential paths. Additionally, the convergence of the algorithm will be accelerated, which is biased by positive feedback, but diversification will be weakened. To improve the search diversification and maintain the necessary intensification, a meeting strategy is introduced into the traditional ACO, and a new ACO algorithm is proposed. The meeting strategy is as follows.

When approximately half of the cities have been visited by ants based on the same operations as in the traditional ACO, all ants should be evaluated to determine whether they can meet with other ants. This determination can be made easily based on the number of cities that are stored in the ants' tabu lists. Two ants are a pair of meeting ants if the union of the cities in their tabu lists covers all the cities in the problem. Then, the meeting ants will quit the current search iteration, and a new tour will be created by the meeting ants; this tour can be easily realized by combining the two ants' tabu lists. If the number of cities in the combined tabu list is larger than the number of all cities, the repeated cities in the new tabu list should be eliminated to guarantee that the number of cities in the new tabu list is equal to the number of all cities. Thus, according to the cities in the combined tabu list, the new tour can be constructed. Additionally, the pheromone values will be added to the new combined tours. Unfortunately, in real practice, the number of ants in each iteration is finite, and the number of meeting ants is also limited. To normalize the search process, a threshold constant v is applied. If the number of meeting ants surpasses the threshold constant, the search process will stop, and the pheromone values will be deposited on the new tour. If the number of meeting ants does not exceed the threshold constant, the search process will continue until all tours are generated, and the pheromone update law is the same as that of the traditional ACO.

In this algorithm, the pheromone updates according to the following equation:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^{v} \Delta\tau^k_{ij} \tag{8}$$

To diversify the search process, the pheromone values of all paths should be limited to an interval $[\tau_{\min}, \tau_{\max}]$, which is as follows:

$$\tau_{ij} = \begin{cases} \tau_{min}, & if\ \tau_{ij} \le \tau_{min} \\ \tau_{max}, & if\ \tau_{ij} \ge \tau_{max} \end{cases} \tag{9}$$

where $\tau_{\min}$ and $\tau_{\max}$ are the pheromone extent values, which can be determined from experiments. A detailed description of $\tau_{\min}$ and $\tau_{\max}$ can be found in reference [6].

Because the cities on the new tour generated by the meeting ants are visited by two ants and not by one ant, as in the traditional ACO, the time to create a new tour will be reduced by approximately half. Moreover, a limiting strategy for the pheromone values is used. Because the meeting strategy is used, at the end of the search process, the ant selects the next cities with a certain amount of diversification, and its ability to explore other potential paths will not be lost. Additionally, the convergence of the algorithm will be accelerated, which is biased by the positive feedback. Therefore, in this new ACO algorithm, the search time will be greatly reduced, and the search diversification of the ant colony will be expanded enormously. The flow chart of the new ACO algorithm is shown in Figure 1.

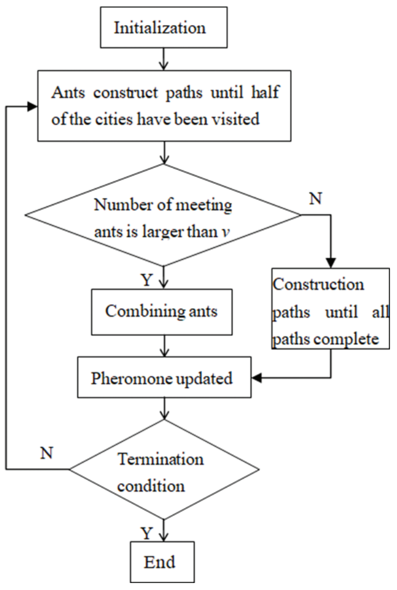
Initialization
Ants construct paths until half of the cities have been visited
Number of meeting ants is larger than v
N
Y
Combining ants
Construction paths until all paths complete
Pheromone updated
N
Termination condition
Y
End

Figure 1 Flow chart of new ant colony optimization algorithm.

In this new algorithm, the termination condition is that the length difference for the optimal paths of neighboring iterations is less than 10^{-5}. In addition, to avoid infinite iteration, the maximum number of iterations N is given.

By using C++, the proposed new ACO algorithm can be implemented in this study.

4. SIMULATION EXPERIMENTS

To verify the computing performance of the proposed new ACO algorithm, a number of TSP instances are used. All of these TSPs are available from the TSPLIB benchmark library, which can be found on the web at

www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

For numerous studies on ACO for the TSP, the selection of algorithm parameters is performed through a series of experiments. By means of these experiments, a law of selecting parameters can be obtained easily. Based on this selection law and using trial calculations, the parameters of the new algorithm can be determined to be:

$\alpha = 1, \beta = 2, \rho = 0.5, Q = 100, m = n, N = 2000, \tau_{\min} = 0.00001, \tau_{\max} = 20, \tau_0 = 1, \nu = 1.$

As shown in this configuration, the number of ants is set to the same as the number of cities, and the threshold constant is set to 1 to achieve peak performance for the algorithm. Because the new ACO algorithm is a random search algorithm, the average results from 30 runs are obtained. Experiments are conducted on a laptop with Intel Core-i5-4200U, 3.4 GHz CPU and 32 GB of RAM. The results are summarized in Table 1.

In Table 1, the first column shows the names of the instances, the second column shows the number of cities, the third column shows the optimal solution taken from the TSPLIB, the column "best" shows the best solution found by the algorithm, the column "average" gives the average solution of the 30 independent runs of the algorithm, the column "PD_{best} (%)" gives the percentage deviation of the best solution from the optimal solution over 30 runs, and the column "PD_{avg} (%)" denotes the percentage deviation of the average solution from the optimal solution over 30 runs. PD_{best} and PD_{avg} are given by the formulas:

$$PD_{best}\,(\%) = \frac{\text{best solution} - \text{optimal solution}}{\text{optimal solution}} \times 100 \tag{10}$$

$$PD_{avg}\,(\%) = \frac{\text{average solution} - \text{optimal solution}}{\text{optimal solution}} \times 100 \tag{11}$$

The column "iterations" gives the average number of iterations over 30 runs, and the column "times" gives the average CPU computing times over 30 runs.

As shown in Table 1, for all 20 selected TSP instances, most of the best solutions found by the new algorithm are the given optimal solutions, except for seven instances (Bier127, Pr136, Ch150, A280, Lin318, Rd400, and Rat575), which are shown in bold in column PD_{best} (%). However, the percentage deviations of the best solutions for those seven instances (Bier127, Pr136, Ch150, A280, Lin318, Rd400, and Rat575) are all very small, less than 0.5% (except for that of Rat575, which is only 1.62%). In other words, 95% of the values of PD_{best} (%) are less than 0.5%, which means that the best solution found in the 30 trials approximates the given optimal solution with a percentage deviation of less than 0.5%. Moreover, most of the average solutions found by the new algorithm in 30 trials approximate the given optimal solutions, except for five instances (including Att48, Eil51, Berlin52, Eil76, and Eil101), the average solutions of which are the same as the given optimal solutions. Furthermore, most of the values of PD_{avg} (%) are less than 1%, except for three instances (including KroA200, Rd400, and Rat575). In other words, 85% of the values of PD_{avg} (%) are less than 1%, which means that the average solution found in the 30 trials approximates the given optimal solution with a deviation of less than 1%. The value of **0** shown in bold in column PD_{avg} (%) indicates that all solutions

Table 1 | Results for 20 TSPs by new ant colony optimization algorithm.

Instances	Size (Cities)	Optimal Solution	Solution by New Ant Colony Optimization Algorithm					
			Best	Average	PD_{best} (%)	PD_{avg} (%)	Iterations	Times (s)
Att48	48	33,522	33,522	33,522	**0**	**0**	105.4	7.1
Eil51	51	426	426	426	**0**	**0**	113.2	7.3
Berlin52	52	7542	7542	7542	**0**	**0**	115.6	7.4
St70	70	675	675	677.45	**0**	0.36	133.6	8.8
Pr76	76	108,159	108,159	108238.12	**0**	0.07	137.4	9.2
Eil76	76	538	538	538	**0**	**0**	138.6	9.3
KroA100	100	21,282	21,282	21312.34	**0**	0.14	157.3	10.4
Eil101	101	629	629	629	**0**	**0**	158.1	10.5
Lin105	105	14,379	14,379	14398.63	**0**	0.14	162.8	10.7
Pr124	124	59,030	59,030	59221.23	**0**	0.32	201.5	12.1
Bier127	127	118,282	118,284	118483.31	0	0.17	206.3	12.5
Ch130	130	6110	6110	6125.52	**0**	0.25	215.8	13.1
Pr136	136	96,772	96,774	97088.62	0	0.33	247.1	13.7
Ch150	150	6528	6531	6564.25	0.05	0.56	405.2	19.7
KroA200	200	29,368	29,378	29682.72	0.03	1.07	557.5	27.1
Tsp225	225	3916	3916	3938.51	**0**	0.57	579.3	31.6
A280	280	2579	2581	2603.81	0.08	0.96	650.1	37.3
Lin318	318	42,029	42,084	42392.79	0.13	0.87	700.3	46
Rd400	400	15,281	15,314	15521.29	0.22	1.57	744.6	55.5
Rat575	575	6773	6883	6933.75	1.62	2.37	865.7	68.7

found in the 30 trials have the same length as the given optimal solutions. Moreover, according to the computing times and number of iterations, the computing speed of the new algorithm is fast. The longest computing time and the largest number of iterations are 68.7 s and 865.7, respectively. There are only five instances whose computing times are longer than 30 s, and only six instances whose number of iterations are larger than 500. Therefore, the numerical values presented in Table 1 show that the new ACO algorithm can indeed provide good solutions for TSPs at a fast speed.

Finally, from Table 1, it can be found that, as the scale of the TSP problem increases, the corresponding computing time will increase too. However, their increasing rates are different. For example, when the scale of the TSP problem increases from 51 to 200, which increases about four times. However, the corresponding computing time increases from 7.3 to 27.1, which increases only about 3.7 times. In this increasing range for the problem scale which increases 149, the computing time increases 19.8. The rate of the increasing time to the increasing scale is 0.13. Moreover, when scale of the TSP problem increases from 400 to 575, the corresponding computing time increases from 55.5 to 68.7. In this range, the rate of the increasing time to the increasing scale is only 0.07 which is much less than 0.13. That is to say, as the problem scale increases, the increase of the computing time will become slow. Therefore, the computing efficiency of new ACO algorithm will improve as the scale of the TSP problem increases, and which is an advantage of the new ACO algorithm.

To verify the computational performance of the new ACO algorithm proposed in this paper, the computational results of the algorithm have been compared with those of 16 state-of-the-art algorithms from the literature, most of which have been proposed in the past three years, as summarized in Table 2.

It should be noted that, in Table 2, algorithms 1 to 17 represent the new algorithm in this paper, HMMAS [13], HEAS [16], ACO-3Opt [17], DABC-NO [20], DCSA [22], RKCS [23], ABO [24], DBA [25], FFOA [26], GA-MRLH [27], AAA [28], ICA [30], BHA [31], SA-SOSOA [32], DSOSA [33], and DABC-TAC [34], respectively.

From Table 2, according to the PD_{best} and PD_{avg}, of the 17 algorithms (including the new ACO algorithm proposed in this study), the BHA and HMMAS are the worst-performing algorithms. Their best solutions and average solutions are both poor. However, due to its limited results, the BHA performs more poorly than the HMMAS. Moreover, the SA-SOSOA, DBA, DCSA, and the algorithm proposed here are the four well-performing ones, and most of their best solutions are equal to the optimal solutions. Furthermore, the average solutions of those algorithms approach the optimal solutions; in fact, a number of their average solutions are equal to the optimal solutions. Therefore, the results of the algorithm proposed here outperform most state-of-the-art algorithms.

To compare the results more clearly, the results of four good state-of-the-art algorithms (1, 6, 9, and 15, which are the proposed new algorithm, DCSA, DBA, and SA-SOSOA, respectively) for the same 15 TSPs (Eil51, Berlin52, St70, Pr76, Eil76, KroA100, Eil101, Lin105, Pr124, Bier127, Ch130, Pr136, Lin318, Rd400, and Rat575) are summarized in Figure 2 and Table 3. In Table 3, the last line denotes the average values of PD_{best} and PD_{avg} for the 15 TSPs, which are shown in bold.

As shown in Figure 2, for PD_{best}, the proposed new algorithm is almost the best, and PD_{best} is zero in most instances. The SA-SOSOA is the best; its PD_{best} values are zero or negative for all instances except one. However, for PD_{avg}, the proposed new algorithm is not a better one, and the SA-SOSOA is the best. Moreover, as shown in Table 3, for 15 TSPs, using the algorithm proposed here, there are only three best results that are not the optimal solutions, and the average value of PD_{best} is 0.13. Moreover, there are only four average solutions that are equal to the optimal solutions, and the average value of PD_{avg} is 0.44. For the DCSA, there are four best results that are not the optimal solutions, and the average value of PD_{best} is 0.21, which is larger than that of the algorithm proposed here. In addition, there are eight average solutions that are equal to the optimal solutions, and the average value of PD_{avg} is 0.42, which is similar to that of the algorithm proposed here. In other words, although the number of average solutions obtained by the DCSA that are equal to the optimal solutions is more than that of the new algorithm proposed here, the average value of PD_{avg} is not less than that of the algorithm proposed here for large values of its PD_{avg}. Therefore, the algorithm proposed here is better than the DCSA. Moreover, for the DBA, there are also three best results that are not the optimal solutions, and the average value of PD_{best} is 0.13, which is the same as that of the algorithm proposed here. There are six average solutions that are equal to the optimal solutions, and the average value of PD_{avg} is 0.34, which is smaller than that of the DCSA or the algorithm proposed here. Therefore, the algorithm proposed here performs more poorly than the DBA. Finally, for the SA-SOSOA, there are three best results that are not the optimal solutions, and the average value of PD_{best} is only 0.005, which is much less than that of the DBA. Moreover, there are only five average solutions that are different from the optimal solutions, and the average value of PD_{avg} is only 0.16, which is also much less than that of the DBA. Therefore, the SA-SOSOA is the best algorithm for those 15 TSPs. Moreover, the new algorithm proposed in this study is a highly suitable method to solve the TSP, and its performance is better than most state-of-the-art algorithms, except for a few algorithms, such as the DBA and the SA-SOSOA.

5. DISCUSSION

To verify the performance of the new ACO algorithm, the comprehensive comparison study of the new ACO algorithm and traditional ACO is discussed.

Based on testing and experience, the parameters of the traditional ACO are as follows:

$\alpha = 1, \beta = 2, \rho = 0.5, Q = 100, m = 50, N = 2000, \tau_0 = 1.$

For comparison, the parameters of the new ACO algorithm are as follows:

$\alpha = 1, \beta = 2, \rho = 0.5, Q = 100, m = 50, N = 2000, \tau_{min} = 0.00001, \tau_{max} = 20, \tau_0 = 1, v = 1.$

It must be noted that to make the comparison fairer, the same parameters are used for both algorithms. However, the parameters used in this study may not be the most suitable values for both algorithms.

Table 2 | Comparison with 16 state-of-the-art algorithms.

Instances	Algorithm Numbers											
	1		2		3		4		5		6	
	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)
Att48	0	0	–	–	0	0.53	–	–	–	–	–	–
Eil51	0	0	3.04	5.65	0	0.7	0	0.08	0	0.39	0	0
Berlin52	0	0	0.03	0.03	0	0.11	0	0	0	0	0	0
St70	0	0.36	1.6	4.81	0	0	0.15	0.42	0	0.66	0	0
Pr76	0	0.07	0.36	1.78	0	0.32	–	–	0	0.47	0	0
Eil76	0	0	2.14	3.73	0	2.42	0	0.34	–	–	0	0
KroA100	0	0.14	0.42	0.68	0	0.11	0	0.21	0	1.11	0	0
Eil101	0	0	8.89	11.78	–	–	0	0.25	1.71	2.95	0	0.22
Lin105	0	0.14	0.11	1.29	–	–	0	0.1	–	–	0	0
Pr124	0	0.32	0.84	1.15	–	–	–	–	–	–	0	0
Bier127	0	0.17	1.39	2.24	0	0.91	–	–	–	–	0	0.06
Ch130	0	0.25	1.49	3.39	0	2.54	–	–	–	–	0	0.42
Pr136	0	0.33	4.47	9.13	–	–	–	–	–	–	0.01	0.24
Ch150	0.05	0.56	0.4	3.39	0	0.34	0.64	1.12	–	–	0	0.33
KroA200	0.03	1.07	2.15	6.94	0.17	1.07	0.56	0.94	–	–	0.04	0.26
Tsp225	0	0.57	4.03	7.35	0	9.34	–	–	9.28	15.03	0	1.09
A280	0.08	0.96	11.67	13.75	0	0.54	–	–	25.66	28.44	0	0.51
Lin318	0.13	0.87	7.9	10.75	2.38	5.09	–	–	–	–	0.22	0.96
Rd400	0.22	1.57	8.19	9.44	0	1.44	1.94	2.18	–	–	1.08	1.65
Rat575	1.62	2.37	11.85	14.36	0	0	3.4	3.53	–	–	1.81	2.71

Instances	Algorithm Numbers											
	7		8		9		10		11		12	
	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)
Att48	–	–	0	0.17	–	–	–	–	0	0.18	0	0.4
Eil51	0	0.21	0	0.23	0	0	0	0.36	0	0.33	0	0.52
Berlin52	0	0	0	0.98	0	0	0	0	0	0.12	0	0.28
St70	0	0.34	0.15	0.49	0	0	0	0.33	0	0.66	0	0.67
Pr76	0	0	0	0.22	0	0	–	–	0	1.29	0.17	1.21
Eil76	0	0.2	0	4.65	0	0.14	0.37	1.12	0	1.36	–	–
KroA100	0	0	0.14	4.14	0	0	0	0.35	0	0.34	–	–
Eil101	0	0.33	1.75	1.75	0	0.54	0.95	2.07	0	2.16	–	–
Lin105	–	–	0.28	0.51	0	0	0	0.33	0	0.05	–	–
Pr124	–	–	–	–	0	0.012	–	–	0	0.3	–	–
Bier127	0	0.44	0	0.49	0	0.08	–	–	0.33	0.98	–	–
Ch130	0.26	0.87	0	3.23	0	0.23	–	–	0.36	1.54	–	–
Pr136	0.28	0.97	–	–	0	0.23	–	–	–	–	–	–
Ch150	–	–	0	1.12	0	0.34	0.46	1.38	0	0.71	–	–
KroA200	–	–	0	2.67	0	0.27	–	–	0.23	0.86	–	–
Tsp225	–	–	0	1.69	0	0.73	–	–	0.1	1.67	–	–
A280	–	–	–	–	0	0.3	–	–	0.62	2.96	–	–
Lin318	–	–	0.17	0.73	0.29	1.03	–	–	0.54	2.3	–	–
Rd400	–	–	0.13	0.15	0.35	1.2	–	–	–	–	–	–
Rat575	–	–	0	0.55	1.31	1.93	–	–	–	–	–	–

Instances	Algorithm Numbers									
	13		14		15		16		17	
	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)	PD_{best} (%)	PD_{avg} (%)
Att48	–	–	2.03	2.84	0	0	–	–	–	–
Eil51	0.94	1.64	2.79	7.73	0	0	0	0.45	0	–
Berlin52	0.03	0.03	8.87	12.12	−0.03	0	0	0.01	0	–
St70	–	–	7.15	18.16	0	0	0	0.62	0	–
Pr76	–	–	–	–	0	0.98	0	0.36	0	–
Eil76	1.86	2.49	5.25	50.05	0	0	0.74	1.75	0	–
KroA100	0.01	0.05	–	–	0	0.67	0	0.6	0	–
Eil101	2.7	3.53	14.53	42.67	0	0	1.75	3.43	0.05	–
Lin105	0.02	0.66	–	–	0	0	0.01	0.37	–	–
Pr124	–	–	–	–	−0.08	0	0	0.68	0	–
Bier127	0.5	0.87	–	–	0	0	–	–	–	–
Ch130	1.55	2.5	–	–	0	0	–	–	–	–
Pr136	–	–	–	–	0	0.13	0.69	0.93	0	–
Ch150	0.41	0.55	–	–	–	0	0.21	0.38	0.22	–
KroA200	1.14	1.6	–	–	–	0	0.37	0.96	0.05	–
Tsp225	–	–	–	–	–	–	0.47	–	0	–
A280	–	–	–	–	–	–	–	–	0.24	–
Lin318	3.45	4.75	–	–	0	0	0.41	2.24	0.26	–
Rd400	–	–	–	–	0.19	0.24	–	–	0.26	–
Rat575	4.62	5.48	–	–	0	0.43	4.43	5.08	0.75	–

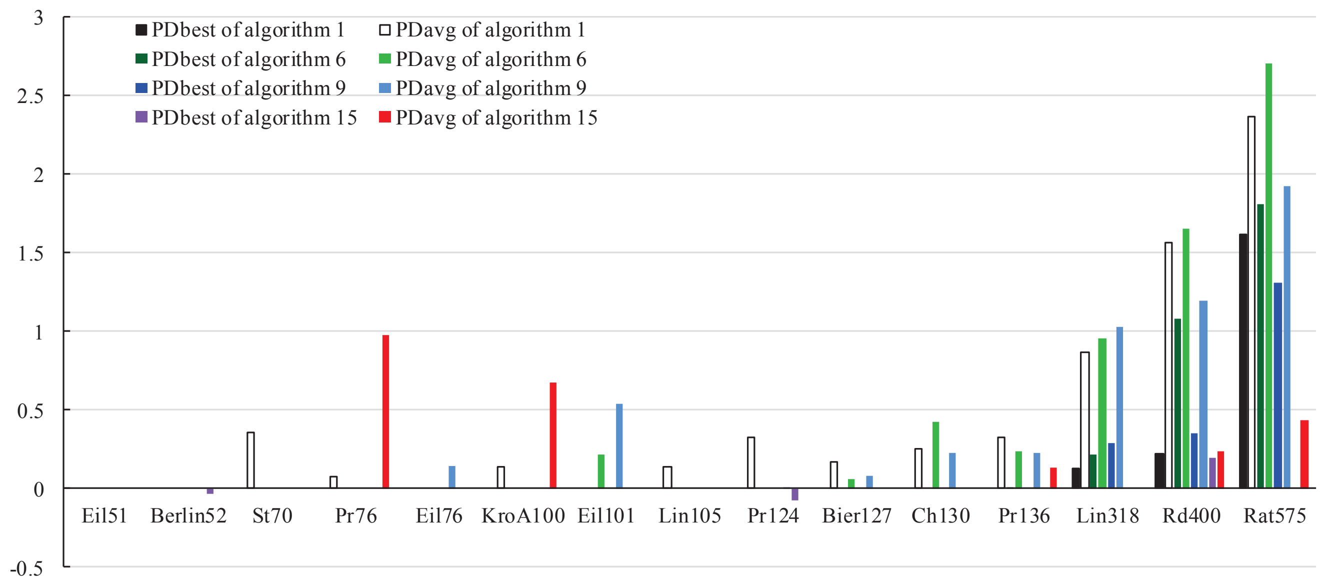

Figure 2 Comparison of four good state-of-the-art algorithms for 15 TSPs.

Table 3 Comparison results of four good algorithms.

Instances	**Algorithm Numbers**							
	1		**6**		**9**		**15**	
	PD_{best} (%)	**PD_{avg} (%)**	**PD_{best} (%)**	**PD_{avg} (%)**	**PD_{best} (%)**	**PD_{avg} (%)**	**PD_{best} (%)**	**PD_{avg} (%)**
Eil51	0	0	0	0	0	0	0	0
Berlin52	0	0	0	0	0	0	−0.03	0
St70	0	0.36	0	0	0	0	0	0
Pr76	0	0.07	0	0	0	0	0	0.98
Eil76	0	0	0	0	0	0.14	0	0
KroA100	0	0.14	0	0	0	0	0	0.67
Eil101	0	0	0	0.22	0	0.54	0	0
Lin105	0	0.14	0	0	0	0	0	0
Pr124	0	0.32	0	0	0	0.012	−0.08	0
Bier127	0	0.17	0	0.06	0	0.08	0	0
Ch130	0	0.25	0	0.42	0	0.23	0	0
Pr136	0	0.33	0.01	0.24	0	0.23	0	0.13
Lin318	0.13	0.87	0.22	0.96	0.29	1.03	0	0
Rd400	0.22	1.57	1.08	1.65	0.35	1.2	0.19	0.24
Rat575	1.62	2.37	1.81	2.71	1.31	1.93	0	0.43
Average	**0.13**	**0.44**	**0.21**	**0.42**	**0.13**	**0.34**	**0.005**	**0.16**

For comparison, eight TSPs selected from Table 1 are used in this study: Eil51, St70, Pr76, KroA100, Lin105, Pr124, Bier127, and Ch150. The best, worst and average results of 30 runs are obtained. The results are shown in Table 4.

In Table 4, the "RE" denotes the percentage deviation of the solution obtained from the new ACO algorithm compared to the solution obtained from traditional ACO, which is defined as

$$RE\,(\%) = \frac{\text{solution from traditional algorithm} - \text{solution from new algorithm}}{\text{solution from traditional algorithm}} \times 100 \qquad (12)$$

As shown in Table 4, for all eight TSPs, the lengths found by the new ACO algorithm (including best, worst, and average ones) are all shorter than those found by traditional ACO. For the best results, the RE is always less than 1, and its average value is only 0.13%. That is, the differences between the best results of the two algorithms are not very large. In other words, both algorithms can find relatively suitable lengths for those eight TSPs. For the worst results, the RE is generally less than 10 except for one, and its average value is 5.27%, which is much larger than that of the best results. In other words, the new ACO algorithm can find much shorter lengths for the worst results compared to traditional ACO. Moreover, for the average results, the RE is always less than 3%, and its average value is 1.77%, which is larger than that of the best results but is much smaller than that of the worst results. However, as for the difference between the best results and the worst results, the RE is very large, generally larger than 70%, except for two results. In addition, the RE's average value is 81.56%, which is still larger than 80%. In other words, the results found by the new ACO algorithm are more aggregated than those found by traditional ACO. In other words, the new ACO algorithm is more stable than traditional ACO. Therefore, the new ACO algorithm can significantly improve traditional ACO.

Moreover, the statistical variation of the results for eight TSPs solved by two algorithms over the 30 independent runs is shown in Table 5, and the comparison of the results for eight TSPs solved by two algorithms is also shown in Figure 3. In this study, two types of statistical indexes are used, namely, the coefficient of variation and the relative error. The coefficient of variation, defined as the ratio between the standard deviation and the mean value, gives information on the uniformity of solutions. The relative error, defined as the percentage deviation of the computing solution from the optimal solution over 30 runs, gives information on the precision of solutions. In this instance, to comprehensively study the problem, three relative error indexes are used: PD_{best}, PD_{worst}, and PD_{avg}. These indexes denote the percentage deviation of the best solution from the optimal solution over 30 runs, the percentage deviation of the worst solution from the optimal solution over 30 runs, and the percentage deviation of the average solution from the optimal solution over 30 runs.

As shown in Figure 3, for all eight TSPs, the results of the new ACO algorithm (including the best, worst, and average ones) are better than those of traditional ACO. Moreover, from Table 5, for traditional ACO, the average values of PD_{best}, PD_{worst}, and PD_{avg} are 0.14%, 6.48%, and 2.07%, respectively. In comparison, those for the new ACO algorithm are 0.0075%, 0.82%, and 0.98%, respectively. Thus, the computational relative errors of the new ACO algorithm are much smaller than those of traditional ACO. In other words, the computational precision of the new ACO algorithm is much better than that of traditional ACO. Moreover, compared with the coefficients of variation of traditional ACO, whose average value is 1.62, those of the new ACO algorithm are much smaller, with an average value of 0.98. In other words, the uniformity of the solutions of the new ACO algorithm is also much better. Therefore, the new ACO algorithm can improve the computational results and computational stability greatly for TSPs. Furthermore, compared with the results of traditional ACO, as the TSPs become more complex (greater length or more cities), the computing results are much better, and the advantages of the new ACO algorithm are better. The reason for this phenomenon may be that when the TSPs become more complex (greater length or more cities), the meeting strategy of the new ACO algorithm will be used more frequently, and thus, the increased advantage of the method of computing results in the new ACO algorithm compared with that of traditional ACO will be more noticeable.

To compare the computing speed of the two algorithms, the iterations with the best results as shown in Table 4 for the two algorithms are shown in Figure 4.

As shown in Figure 4, for all TSPs, the computing speeds of the new ACO algorithm are all faster than those of traditional ACO. Moreover, as the number of cities increases, the difference between the computing times for the two algorithms will also increase. Therefore, as the complexity of the problem increases, the computational efficiency of the new ACO algorithm increases. In other words, the more complicated the problem, the better the computational efficiency of the new ACO algorithm.

To study the computational efficiency more comprehensively, the statistical results of the computing time for eight TSPs solved by two algorithms over 30 independent runs are shown in Table 6. In this study, two statistical indexes are used: the average computing times and the coefficient of variation for computing times. The average

Table 4 | Comparison results of the new ant colony optimization algorithm and traditional ant colony optimization for eight TSPs.

Instances	Solutions by Traditional Ant Colony Optimization				Solutions by New Ant Colony Optimization Algorithm			
	Best	Worst	Average	Worst–Best	Best (RE/%)	Worst (RE/%)	Average (RE/%)	Worst–Best (RE/%)
Eil51	426	447	437.2	21	426 (0)	426 (4.7)	426 (2.25)	0 (100)
St70	675	725	695.3	50	675 (0)	684 (5.66)	677.51 (2.57)	9 (82)
Pr76	108,316	121,574	109445.6	3258	108,159 (0.14)	108,527 (10.73)	108238.12 (1.1)	368 (88.74)
KroA100	21,282	22,163	21591.4	881	21,282 (0)	21,427 (3.43)	21316.46 (1.29)	145 (83.54)
Lin105	14,425	15,326	14746.1	901	14,379 (0.32)	14,522 (5.25)	14398.63 (2.36)	143 (84.13)
Pr124	59,112	61,361	59742.3	2249	59,030 (0.14)	60,167 (1.95)	59228.23 (0.87)	1137 (49.67)
Bier127	118,317	129,574	120862.7	11,257	118,284 (0)	118,742 (8.37)	118493.63 (1.97)	458 (95.99)
Ch150	6558	6732	6678.2	174	6532 (0.41)	6587 (2.17)	6568.25 (1.71)	55 (68.39)
Average of RE					**0.13**	**5.27**	**1.77**	**81.56**

Table 5 | Comparison of statistical results for the new ant colony optimization algorithm and traditional ant colony optimization for eight TSPs.

Instances	Solutions by Traditional Ant Colony Optimization				Solutions by New Ant Colony Optimization Algorithm			
	PD_{best} (%)	PD_{worst} (%)	PD_{avg} (%)	Coefficient of Variation	PD_{best} (%)	PD_{worst} (%)	PD_{avg} (%)	Coefficient of Variation
Eil51	0	4.93	2.63	1.43	0	0	0	0.92
St70	0	7.14	3.01	1.55	0	1.33	0.37	1.04
Pr76	0.15	12.4	1.19	1.79	0	0.34	0.07	0.96
KroA100	0	4.14	1.45	1.62	0	0.68	0.16	1.12
Lin105	0.32	6.59	2.55	1.59	0	0.99	0.14	0.91
Pr124	0.14	3.95	1.21	1.65	0	1.93	0.34	0.89
Bier127	0.03	9.55	2.18	1.74	0	0.39	0.18	1.03
Ch150	0.46	3.13	2.3	1.61	0.06	0.9	0.62	0.93
Average	**0.14**	**6.48**	**2.07**	**1.62**	**0.0075**	**0.82**	**0.24**	**0.98**

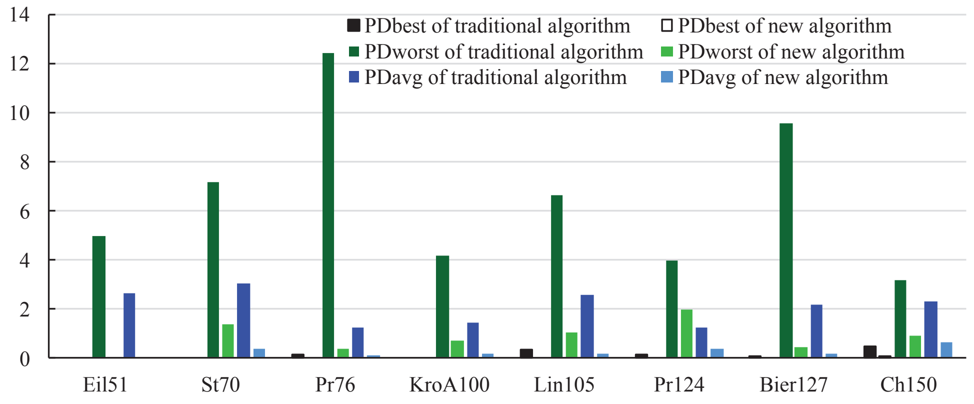

Figure 3 Comparison of new ant colony optimization algorithm and traditional ant colony optimization for eight TSPs.

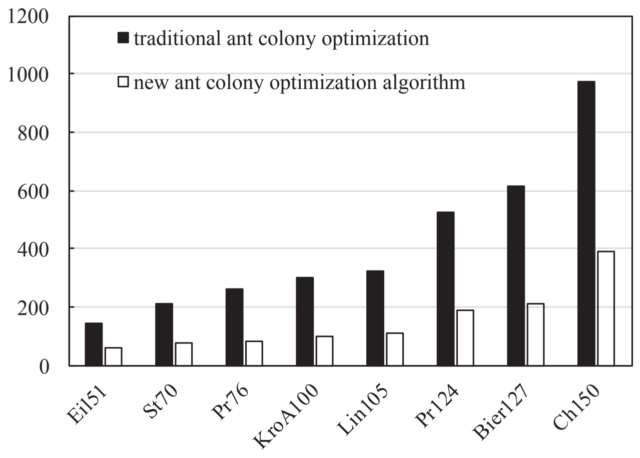

Figure 4 Iterations of traditional ant colony optimization and new ant colony optimization algorithm for eight TSPs.

Table 6 Comparison of statistical results for computing time by the new ant colony optimization algorithm and traditional ant colony optimization for eight TSPs.

Instances	Solutions by Traditional Ant Colony Optimization		Solutions by New Ant Colony Optimization Algorithm	
	Average Computing Time (s)	Coefficient of Variation	Average Computing Time (s)	Coefficient of Variation
Eil51	12.4	1.21	7.3	0.88
St70	15.6	1.32	8.8	0.92
Pr76	16.7	1.43	9.2	0.95
KroA100	19.3	1.51	10.4	1.04
Lin105	20.1	1.45	10.7	0.98
Pr124	23.7	1.56	12.1	1.05
Bier127	25.1	1.59	12.5	1.09
Ch150	41.4	1.55	19.7	1.03

computing times provide information on the specific computing speed. The coefficient of variation, defined as the ratio between the standard deviation and the mean value, gives information on the uniformity of the computing speed.

As shown in Table 6, for all eight TSPs, the computing times of the new ACO algorithm are much less than those of traditional ACO, and the coefficients of variation for the new ACO algorithm are also less. To analyze the degree of reduction of computing time by both algorithms, the reduction rate, which is the ratio of the reduction value to the computing time by traditional ACO, is obtained. For eight TSPs from Eil51 to Ch150, the reduction rates are 41.1%, 43.6%, 44.9%, 46.1%, 46.8%, 48.9%, 50.2%, and 52.4%, respectively. Therefore, as the complexity of the TSPs increases, the reduction rate of the computing time for the two algorithms also increases. Moreover, using the traditional ACO, as the scale of the TSP problem increases from 51 to 100 and 100 to 150, the corresponding computing time increases from 12.4 to 19.3 and 19.3 to 41.4, respectively. And the corresponding rates of the increasing time to the increasing scale are 0.14 and 0.44. However, for the new ACO algorithm, in the same increasing ranges of the problem scale, the corresponding computing time increases from 7.3 to 10.4 and 10.4 to 19.7, respectively. And the corresponding rates of the increasing time to the increasing scale are only 0.06 and 0.18 which are much less than those of the traditional ACO. And their rates of new algorithm to traditional ACO are 0.43 and 0.4 respectively. That is to say, the computational efficiency of the new algorithm is much high than that of traditional ACO. And as the scale of the TSP problem increases, the superiority of computational efficiency for the new algorithm will increase too. Thus, the new ACO algorithm can be used in complicated combinatorial optimization problems. The reason for this phenomenon may be that as the complexity of the TSPs (greater length or more cities) increases, the meeting strategy of the new algorithm will be used more frequently. Then, the new tour will be generated by the meeting ants quickly, and the number of new tours generated by the meeting ants will increase quickly. Therefore, the rate of increase of the computational efficiency for the new ACO algorithm will also increase.

6. CONCLUSIONS

ACO is highly suitable for performing combinatorial optimization, and is typically applied to the TSP. However, as a heuristic algorithm, it has many shortcomings, such as slow convergence speed and low searching efficiency. To overcome its shortcomings, a new ACO algorithm is proposed that enlarges the ants' search space and diversifies the searched solutions. The main strategy used in this new algorithm is to combine pairs of searching ants to increase the diversification of the solutions.

To compare the results of the new ACO algorithm with the best known solutions and the solutions found by some state-of-the-art algorithms proposed in the literature, 20 typical TSPs are used. The applications indicate that, for all the selected TSPs, the solutions obtained by the new algorithm are better than most of the other algorithms and the new algorithm is a very suitable method for solving the TSP. Finally, a comprehensive comparative study of the new ACO algorithm and traditional ACO is discussed.

Based on the above studies, the following conclusions can be drawn.

1. For all 20 selected TSP instances, most of the best solutions found by the new ACO algorithm are the given optimal solutions, except for seven instances. In addition, most of the average solutions found by the new ACO algorithm in 30 trials approximate the given optimal solutions, except for five instances.
2. The SA-SOSOA, DBA, DCSA, and the new algorithm proposed here are well-performing algorithms, and most of their best solutions are equal to the optimal solutions. Furthermore, the average solutions of those algorithms approach the optimal solutions. In fact, some of their average solutions are equal to the optimal solutions.
3. The results found by the new ACO algorithm are more aggregated than those found by traditional ACO. In other words, the new ACO algorithm is more stable than traditional ACO. Therefore, the new algorithm can significantly improve on the traditional algorithm.
4. The computing precision of the new ACO algorithm is much better than that of traditional ACO, and the computing speed of the new ACO algorithm is also faster.
5. As the complexity of the problem increases, the computational efficiency of the new ACO algorithm increases. In other words, the more complicated the problem, the better the computational efficiency of the new ACO algorithm.

Therefore, by comparison with traditional ACO and some state-of-the-art algorithms, the new ACO algorithm has higher computing precision, better computing stability, and faster computing speed. However, the new algorithm has only been tested on TSPs that are not very large. For large-scale TSPs, computational complexity will increase quickly, and the effect on computation may be somewhat detrimental. Therefore, comprehensive testing, especially for large-scale TSPs and for other combinatorial optimization problems, is our next task.

CONFLICT OF INTEREST

The authors declare they have no conflicts of interest.

AUTHORS' CONTRIBUTIONS

Literature search, study design, manuscript writing, data collection, data analysis, and software are all conducted by Wei Gao.

ACKNOWLEDGMENTS

This study was funded by Fundamental Research Funds for the Central Universities (grant number 2016B10214).

REFERENCES

[1] M. Dorigo, T. Stutzle, Ant Colony Optimization, The MIT Press, Cambridge, 2004.
[2] M. Dorigo, V. Maniezzo, A. Colorni, Positive Feedback as a Search Strategy (Technical Report No. 91-016), Politecnico di Milano, Milano, Italy, 1991.
[3] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man. Cybern. B. 26 (1996), 29–41.
[4] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1997), 53–66.
[5] B. Bullnheimer, R. Hartl, C. Strauss, A new rank-based version of the ant system: a computational study, Cent. Eur. J. Oper. Res. 7 (1999), 25–38.
[6] T. Stützle, H.H. Hoos, Max-min ant system, Future Gener. Comput. Syst. 16 (2000), 889–914.
[7] Z.J. Zhang, Z.R. Feng, A novel max-min ant system algorithm for traveling salesman problem, in Proceedings of Intelligent Computing and Intelligent Systems (ICIS 2009), IEEE Press, Piscataway, 2009, pp. 508–511.

[8] H.B. Mei, J. Wang, Z.H. Ren, An adaptive dynamic ant system based on acceleration for TSP, in Proceedings of Computational Intelligence and Security, IEEE Press, Piscataway, 2009, pp. 92–96.
[9] P. Guo, Z.J. Liu, Moderate ant system: an improved algorithm for solving TSP, in Proceedings of Seventh International Conference on Natural Computation, IEEE Press, Piscataway, 2011, pp. 1190–1196.
[10] Z.C.S.S. Hlaing, M.A. Khine, Solving traveling salesman problem by using improved ant colony optimization algorithm, Int. J. Inf. Educ. Technol. 1 (2011), 404–409.
[11] G. Dong, W.W. Guo, K. Tickle, Solving the traveling salesman problem using cooperative genetic ant systems, Expert Syst. Appl. 39 (2012), 5006–5011.
[12] A.Q. Ansari, Ibraheem, S. Katiyar, Comparison and analysis of solving travelling salesman problem using GA, ACO and hybrid of ACO with GA and CS, in Proceedings of IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions, IEEE Press, Piscataway, 2015, pp. 1–5.
[13] Y. Wang, Hybrid max-min ant system with four vertices and three lines inequality for traveling salesman problem, Soft Comput. 19 (2015), 585–596.
[14] Y. Yan, H.S. Sohn, G. Reyes, A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem, Appl. Soft Comput. 60 (2017), 256–267.
[15] J.C. Thill, Y.C. Kuo, The nearest neighbor ant colony system: a spatially-explicit algorithm for the traveling salesman problem, in: J.-C. Thill (Ed.), Spatial Analysis and Location Modeling in Urban and Regional Systems. Advances in Geographic Information Science, Springer, Berlin, Heidelberg, 2018, pp. 301–322.
[16] G.M. Jaradat, Hybrid elitist-ant system for a symmetric traveling salesman problem: case of Jordan, Neural Comput. Appl. 29 (2018), 565–578.
[17] Ş. Gülcü, M. Mahi, Ö.K. Baykan, *et al.*, A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem, Soft Comput. 22 (2018), 1669–1685.
[18] X.M. You, S. Liu, Y.M. Wang, Quantum dynamic mechanism-based parallel ant colony optimization algorithm, Int. J. Comput. Int. Sys. 3 (2010), 101–113.
[19] J. Bang, J. Ryu, C. Lee, *et al.*, A quantum heuristic algorithm for the traveling salesman problem, J. Korean Phys. Soc. 61 (2012), 1944–1949.
[20] M.S. Kıran, H. İşcan, M. Gündüz, The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem, Neural Comput. Appl. 23 (2013), 9–21.
[21] J. Jones, A. Adamatzky, Computation of the travelling salesman problem by a shrinking blob, Nat. Comput. 13 (2014), 1–16.
[22] A. Ouaarab, B. Ahiod, X.S. Yang, Discrete cuckoo search algorithm for the travelling salesman problem, Neural Comput. Appl. 24 (2014), 1659–1669.
[23] A. Ouaarab, B. Ahiod, X.S. Yang, Random-key cuckoo search for the travelling salesman problem, Soft Comput. 19 (2015), 1099–1106.
[24] J.B. Odili, M.N.M. Kahar, Solving the traveling salesman's problem using the African buffalo optimization, Comput. Intell. Neurosci. 2016 (2016), 1510256.
[25] Y. Saji, M.E. Riffi, A novel discrete bat algorithm for solving the travelling salesman problem, Neural Comput. Appl. 27 (2016), 1853–1866.
[26] L. Huang, G.C. Wang, T. Bai, *et al.*, An improved fruit fly optimization algorithm for solving traveling salesman problem, Front. Inform. Tech. El. Eng. 18 (2017), 1525–1533.
[27] M.M. Alipour, S.N. Razavi, M.R.F. Derakhshi, *et al.*, A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem, Neural Comput. Appl. 30 (2018), 2935–2851.
[28] A.E. Yildirim, A. Karci, Applications of artificial atom algorithm to small-scale traveling salesman problems, Soft Comput. 22 (2018), 7619–7631.
[29] Y.Q. Zhou, R. Wang, C.Y. Zhao, *et al.*, Discrete greedy flower pollination algorithm for spherical traveling salesman problem, Neural Comput. Appl. 31 (2019), 2155–2170.
[30] M.H. Chen, S.H. Chen, P.C. Chang, Imperial competitive algorithm with policy learning for the traveling salesman problem, Soft Comput. 21 (2017), 1863–1875.
[31] A. Hatamlou, Solving travelling salesman problem using black hole algorithm, Soft Comput. 22 (2018), 8167–8175.
[32] A.E.S. Ezugwu, A.O. Adewumi, M.E. Frîncu, Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem, Expert Syst. Appl. 77 (2017), 189–210.
[33] A.E.S. Ezugwu, A.O. Adewumi, Discrete symbiotic organisms search algorithm for travelling salesman problem, Expert Syst. Appl. 87 (2017), 70–78.
[34] Y.W. Zhong, J. Lin, L.J. Wang, *et al.*, Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem, Inf. Sci. 421 (2017), 70–84.
[35] S. Kumar, E. Munapo, M. Lesaoana, *et al.*, A minimum spanning tree based heuristic for the travelling salesman tour, Opsearch. 55 (2018), 150–164.
[36] K.M. Lo, W.Y. Yi, P.K. Wong, *et al.*, A genetic algorithm with new local operators for multiple traveling salesman problems, Int. J. Comput. Int. Sys. 11 (2018), 692–705.
[37] R.Y. Dong, S.S. Wang, G.Y. Wang, *et al.*, Hybrid optimization algorithm based on wolf pack search and local search for solving traveling salesman problem, J. Shanghai Jiao Tong Univ. 24 (2019), 41–47.
[38] M.A.H. Akhanda, S.I. Ayon, S.A. Shahriyar, *et al.*, Discrete spider monkey optimization for traveling salesman problem. Appl. Soft Comput. 86 (2020), 105887.
[39] Y.W. Zhong, L.J. Wang, M. Lin, *et al.*, Discrete pigeon-inspired optimization algorithm with metropolis acceptance criterion for large-scale traveling salesman problem, Swarm Evol. Comput. 48 (2019), 134–144.
[40] C. Jiang, Z.P. Wan, Z.H. Peng. A new efficient hybrid algorithm for large scale multiple traveling salesman problems, Expert Syst. Appl. 139 (2020), 112867.
[41] B. Christian, Ant colony optimization: introduction and recent trends, Phys. Life Rev. 2 (2005), 353–373.
[42] M.B. Chandra, R. Baskaran, Survey on recent research and implementation of ant colony optimization in various engineering applications, Int. J. Comput. Int. Sys. 4 (2011), 566–582.
[43] Y. Nakamichi, T. Arita, Diversity control in ant colony optimization, Artif. Life Robot. 7 (2004), 198–204.