IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 4, 2018, accepted June 19, 2018, date of publication July 5, 2018, date of current version July 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2853129

A Hierarchical Algorithm Based on Density Peaks Clustering and Ant Colony Optimization for Traveling Salesman Problem

ERCHONG LIAO[1], (Member, IEEE), AND CHANGAN LIU[2]
[1]School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China
[2]School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China

Corresponding author: Erchong Liao (ecrobot@ncepu.edu.cn)

This work was supported by the Fundamental Research Funds for the Central Universities under Grant 2016MS121.

ABSTRACT This paper proposed a hierarchical hybrid algorithm for traveling salesman problem (TSP) according to the idea of divide-and-conquer. The TSP problem is decomposed into a few subproblems with small-scale nodes by density peaks clustering algorithm. Every subproblem is resolved by ant colony optimization algorithm, this is the lower layer. The center nodes of all subproblems constitute a new TSP problem, which forms the upper layer. All local tours of these subproblems are joined to generate the initial global tour in the same order that the center nodes are traversed in the upper layer TSP problem. Finally, the global tour is optimized by k-Opt algorithms. Thirty benchmark instances taken from TSPLIB are divided into three groups on the basis of problem size: small-scale, large-scale, and very large-scale. The experimental result shows that the proposed algorithm can obtain the solutions with higher accuracy and stronger robustness, and significantly reduce runtime, especially for the very large-scale TSP problem.

INDEX TERMS Ant colony optimization, density peaks clustering algorithm, k-Opt algorithm, traveling salesman problem.

I. INTRODUCTION

The Traveling Salesman Problem (TSP) attempts to find the shortest tour that traverses all nodes once and only once. TSP is also a typical combinatorial optimization problem. The symmetric TSP means that the distances are the same between from node i to node j and from node j to node i. The problem with n nodes has a set of $(n-1)!$ feasible solutions. The time complexity is $O(n!)$ [1]. The computational complexity of the TSP increases exponentially with the problem size. Thus, it is a well-known NP hard problem.

According to solution accuracy, the TSP algorithms are divided into two categories. One is the exact algorithms which can find the optimal solution in thousands of nodes, the typical algorithms are branch and bound [2], branch and cut [3], branch and price [4], and dynamic programming [5]. The other is the heuristic intelligent algorithms, e.g., Genetic Algorithm (GA) [1], [6], [7], Simulated Annealing (SA) [8], [9], Artificial Bee Colony algorithm(ABC) [10], Ant Colony Optimization (ACO) [11], Particle Swarm Optimization (PSO) [12], neural network [13], [14], tabu search [15], shuffled frog-leaping algorithm [16], wedging insertion method [17].

Every algorithm has some superiority and weakness. Exact algorithms can obtain the optimal solution, but usually consume long runtime, particularly for large-scale TSP. The intelligent algorithms can find the near optimal solution in a relatively short time. Some hybrid intelligent algorithms synthesizing multiple methods can have a better and more competitive performance.

Othman *et al.* [18] introduces the performance and behavior of Water Flow-like algorithm (WFA) applying using 2-Opt and 3-Opt in TSP.

A parallelized neural network system (PMSOM) is proposed to solve the TSP problem [19]. It consists of dividing all cities into municipalities, finding the best overall solution for each municipality, joining neighborhood municipalities to generate the final solution. Test results show that the approach can obtain a better answer in terms of solution quality and runtime.

A hybrid genetic algorithm (HGA) with two local optimization strategies is introduced to solve TSP [20]. The computation results show that it has better performance than GA.

A discrete symbiotic organisms search (DSOS) algorithm is proposed to find a near optimal solution for TSP [21]. The symbiotic organisms search is improved and extended by using three mutation-based local search operators to reconstruct its population. Numerical results show that the proposed method can achieve results close to the theoretical best known solutions (BKS) within a reasonable time.

In improved fruit fly optimization algorithm (IFOA), three improvements are incorporated [22]. The results show a very competitive performance in terms of the convergence rate and precision.

A hybrid method between dynamic programming and memetic algorithm is introduced to solve the TSP with hotel selection which is a variant of the classic TSP [23]. Experiments show the proposed approach has remarkable performance.

An effective local search algorithm based on simulated annealing and greedy search techniques (ASA-GS) is presented to solve the TSP [24]. Test results show that the proposed algorithm provides better compromise between CPU time and solution accuracy for the TSP.

PSO and GA have the shortcomings of premature convergence and poor convergence for high dimensional complex problem, which cannot guarantee to obtain the optimal solution. Neural network is easy to generate unreasonable solution or local optimal solution. ACO is premature convergence, but ACO is a forward feedback method with very few parameters which is easy in adjusting. ACO algorithm has been considered that it is effective for small-scale TSP [25], [26]. Some hybrid algorithms based on ACO have a more competitive performance.

Ant colony extended (ACE) is a novel extensional algorithm of the general ACO framework [27]. Experimental results show ACE has better performance than ant colony system and max-min ant system in almost every tested TSP instance.

A new hybrid method based on SA, GA, ACO, and PSO is presented to solve TSP [28]. First, the initial solutions are generated by using ACO. Then optimize them by genetic simulated annealing. After a given number of iterations, the pheromone information between groups is exchanged by PSO. Experimental results from TSPLIB show that the proposed method has a better solution than the others.

In [29], a parallel cooperative hybrid algorithm (PACO-3Opt) is proposed to solve TSP. Firstly the ACO algorithm generates initial solutions in parallel, and then the 3-Opt algorithm improves these solutions. The algorithm enhances the quality and the robustness of the obtained solutions, significantly decreases the computational time and can solve large-scale TSP problem within a reasonable time.

A hybrid algorithm (ACO-Taguchi) based on ACO and the Taguchi method is proposed to solve TSP [30]. The performance of the proposed algorithm is improved after optimizing the ACO parameters by using the Taguchi method.

Wang presents hybrid max-min ant system (HMMA) [18] with four vertices and three lines inequality to solve TSP. The experimental results show that it can find the better approximate solutions

A hierarchical algorithm based on clustering algorithms and ACO (HCACO) is presented for solving large-scale TSP [31]. Firstly, the large-scale TSP problem is decomposed into several small-scale TSP subproblems by k-means algorithms, while all subproblems could be solved by ACO. Then, the local solutions for all subproblems are merged. Numerical simulation results show that the proposed algorithm has a beneficial effect for large-scale TSP.

In [32], A hybrid method called PSO-ACO-3Opt based on ACO, PSO, and 3-Opt is proposed. In this algorithm, PSO optimizes the parameters of ACO, the initial solution is obtained by ACO, and finally 3-Opt is used to avoid falling in local minimum. Experimental results show that the performance of the proposed method is better than that of other compared methods in terms of solution quality and robustness.

Another hybrid algorithm (C-PSO-ACO-KOpt) based on ACO, PSO, and k-Opt is proposed to solve TSP [33]. ACO is used to generate initial swarm of PSO. Then PSO is made on this swarm to find optimal path which is improved by k-Opt. The benchmark test results show that the algorithm is more efficient with respect to accuracy as well as runtime.

For a TSP problem, a feasible strategy is to firstly decompose into a number of small-scale subproblems which should maintain the main structure of the optimal tour, then solve all small-scale subproblems and finally merge all local tours.

This article proposes a hybrid hierarchical algorithm for TSP. Firstly, the TSP problem is decomposed into small-scale TSP by clustering algorithm based on the superiority of ACO algorithm in small-scale TSP. Then solve the TSP tour by ACO algorithm in each cluster and merge the local tours among clusters. Finally escape from local optima by using k-Opt algorithm to obtain the global tour. Three groups of benchmarks from TSPLIB are used to validate the algorithm. The results show that the algorithm in the article can get the tour closed to the theoretical optimal values in a reasonable and short runtime and has strong robustness.

The rest of the article is organized as follows. Section 2 introduces the Density Peaks Clustering (DPC), ACO, and k-Opt algorithms. The proposed algorithm is detailly explained in Section 3. Experiments and comparisons are revealed in Section 4. The results are analyzed in Section 5. Finally, Section 6 draws some conclusions and provides future work.

II. MATERIALS AND METHODS

A. DENSITY PEAKS CLUSTERING

There are a lot of clustering algorithms, such as k-means [34], affinity propagation algorithm [35], and density peaks

clustering [36]. The DPC algorithm assumes that the cluster centers are surrounded by neighbors with lower local density and that the centers are at a relatively farther distance from any points with higher local density. The local density ρ_i and the distance δ_i from points with higher local density are computed for each point i. Both two variable quantities only rely on the distances d_{ij} between points. Local density ρ of point i is calculated as (1).

$$\rho_i = \sum_j \chi(d_{ij} - d_c). \tag{1}$$

Where,

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0. \end{cases} \tag{2}$$

Let ρ_i indicate the number of points that are closer than d_c to point i. While d_c is the cutoff distance. The clustering result is not sensitive to the value of d_c for massive nodes. δ_i means the shortest distance between point i and any other point with higher density. It is defined as (3).

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}), \tag{3}$$

If point i is the highest density, δ_i is computed as (4).

$$\delta_i = \max_j (d_{ij}). \tag{4}$$

The cluster centers are the points with especially large δ_i, each remaining point is clustered to the same cluster as its nearest neighbor with higher density. The DPC algorithm need not iterate and is executed in a single step, therefore DPC is robust and effective.

B. ANT COLONY OPTIMIZATION

Ant colony algorithm [26], [37], [38], [39] proposed by M. Dorigo in 1991 is a heuristic search algorithm based on swarm intelligence. Ants can find the shortest tour between the nest and the food source in nature. Ants deposit a chemical substance, called pheromone, on the way they passed. Other ants follow the pheromone trail and search the food source. Furthermore, ants reinforce the pheromone while they return to their nest. Meanwhile, the pheromone evaporates over time and reduces its attractiveness. The pheromone density on the way depends on the times the tour is selected, especially in recent time. Ants can find the shortest tour by choosing the ways on which the pheromone density is the highest. Inspired by the foraging behavior of ants in nature, ant colony optimization is proposed to solve the optimization problem in the discrete system. At present, the algorithm has been widely used to solve traveling salesman problem, the assignment problem [40], the scheduling problem [41], the feature selection [42], and path planning of mobile robots [43]. Ant colony algorithm without prior knowledge is a stochastic optimization method. Ants randomly select the node, gradually optimize the tour with the aid of pheromones, and obtain the global optimal tour at last.

Assume that the TSP has n nodes, m ants. k represents the kth ant in the colony. d_{ij} denotes the distance between node i and node j. The kth ant at time t moves from node i to node j according to the transition probability in (5).

$$P_{ij}^k(t) = \begin{cases} \tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t) / \sum_{s \in a_k} \tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t), & j \in a_k \\ 0, & otherwise. \end{cases} \tag{5}$$

Where a_k is the set of nodes not visited by the kth ant. The variables i, j and s are the identifier of the nodes. $\tau_{ij}(t)$ is the intensity of trail between node i and node j at time t, $\eta_{ij}(t)$ is the visibility at time t and generally given by $1/d_{ij}$. The parameters α and β denote the relative importance between trail and visibility. Over time, the pheromone evaporation occurs and the quantity of pheromone is updated as in (6).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^k(t, t+1). \tag{6}$$

Where $\rho \in (0,1)$ is the pheromone evaporation coefficient. $\Delta\tau$ is the total quantity of increased or decreased pheromone left by all ants between node i and node j. It is calculated in (7).

$$\Delta\tau_{ij}^k(t, t+1) = \sum_{k=1}^{n} \Delta\tau_{ij}^k(t, t+1). \tag{7}$$

In the ant-cycle system, the quantity of pheromones left by the kth ant between node i and node j is determined as in (8).

$$\Delta\tau_{ij}^k(t, t+1) = \begin{cases} Q/L^k, & if\ the\ kth\ ant\ uses\ path_{ij} \\ 0, & otherwise. \end{cases} \tag{8}$$

Where Q is a constant. L^k represents the tour length of the kth ant. The algorithm continues until the maximum number of iterations is met. The tour with the shortest length is regarded as the final solution.

C. k-Opt

Some optimization algorithms (e.g. GA, ABC) are presented to decrease the length of the tour. These algorithms sometimes fall in local minimum when searching the optimal tour for TSP. k-Opt algorithm has been used to avoid local optima. 2-Opt [44] and 3-Opt [45] are the typical subclasses of the k-Opt algorithm.

The 2-Opt algorithm basically removes two edges from the tour and reconnects the two edges to obtain a new tour. The steps continue only when the latest tour is shorter. Removing and reconnecting the tour leads to an optimal tour. The 3-Opt algorithm is similar, but instead of removing two edges, three are removed and reconnected.

If the three removed edges are interval among six nodes, all possible 3-Opt reconnection variants are as shown in Fig. 1. And Fig. 2 demonstrates all possible variants among five nodes. A 3-Opt movement is equal to two or three 2-Opt movements. A 3-Opt movement may provide better solutions, but it is significantly slower.

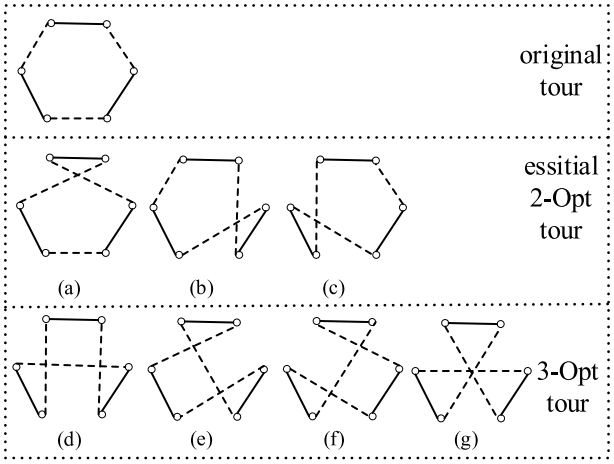


FIGURE 1. All possible 3-Opt reconnection variants among six nodes. There are seven variants. In which, (a), (b), and (c) are essentially 2-Opt movements; while (d), (e), (f), and (g) are 3-Opt movements.

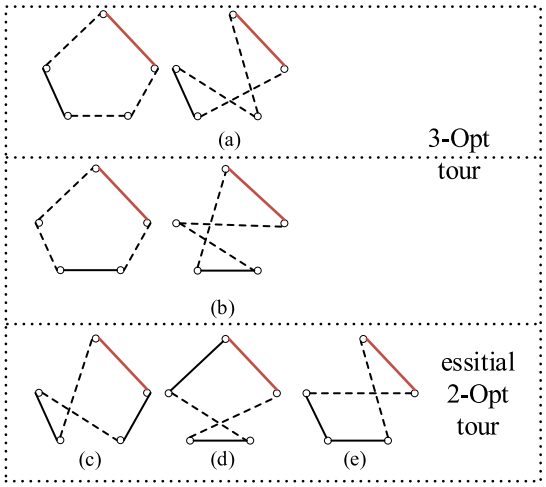


FIGURE 2. All possible 3-Opt reconnection variants among five nodes. The red edge is relatively motionless. There are five possible variants. In which, (c), (d), and (e) are essentially 2-Opt movements; while (a) and (b) are 3-Opt movements, and they are symmetric.

III. PROPOSED ALGORITHM (DPC-ACO-KOpt)

Generally, in the process of solving the TSP problem by using ACO algorithm, computational complexity increases rapidly nonlinearly along with the dimension increase of the TSP problem instance. When the nodes are more than 200, it is difficult to compromise between runtime and solution accuracy. Jiang *et al.* [31] consider that the runtime and solution quality of ACO algorithm are both in the optimal region when the nodes are less than 40, but longer runtime and lower quality when more than 40. In [31], the clustering is conducted by using k-means algorithm. K-means is applicable for the nodes with circle or spherical distribution, but not for linear distribution. DPC which groups all nodes according to the density works effectively in more distribution styles, such as circle, spherical, curvilinear, and linear. This is a bold attempt for DPC in the similar solution.

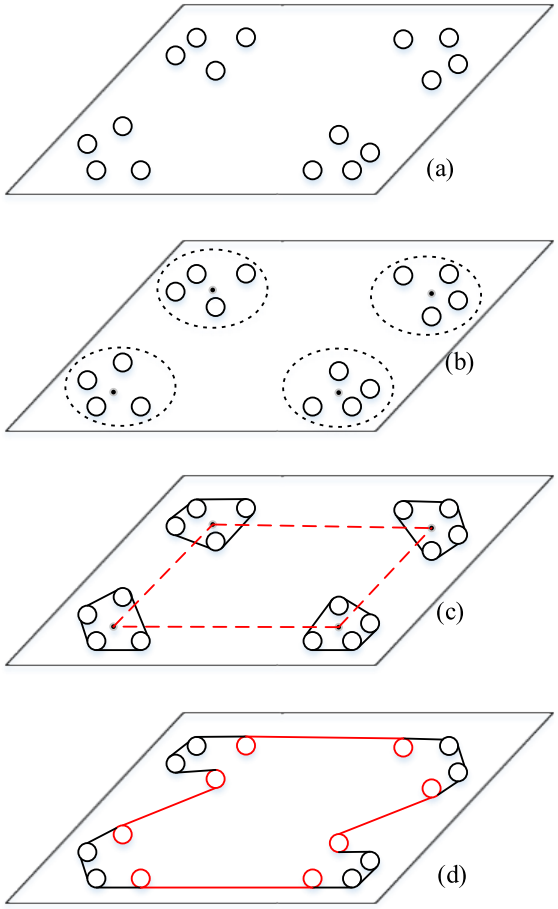


FIGURE 3. Schematic diagram of proposed hierarchical algorithm.

As shown in Fig. 3 and Fig. 4, the paper proposes a hierarchical hybrid method based on ACO. Firstly, group all nodes and find cluster center node inner each group by using DPC algorithm. It takes linear and short time. All nodes are divided into two layers. The lower layer of the proposed approach consists of all clusters. Every cluster is a small-scale TSP problem which could be solved by ACO algorithm. All center nodes comprise the upper layer. Each center node represents one clustering group in the lower layer. Therefore, the upper layer is also a small-scale TSP problem. According to the sequence among groups in the upper layer, the initial global tour which visits all nodes can be constructed by merging local tours for all clustering groups in the lower layer. The final global tour is obtained after optimizing the initial local tour by using k-Opt algorithm.

The local tour of each group in the lower layer would be integrated the global tour. All center nodes in the upper layer constitute a new TSP problem which could also be solved by ACO. The order among center nodes in the upper layer is the sequence of integrating local tours in the groups at the macro level. The local tour in every group is the Hamilton loop. Thus, the local tour should be disconnected at one node, and reconnected with the adjacent group. To obtain the optimal

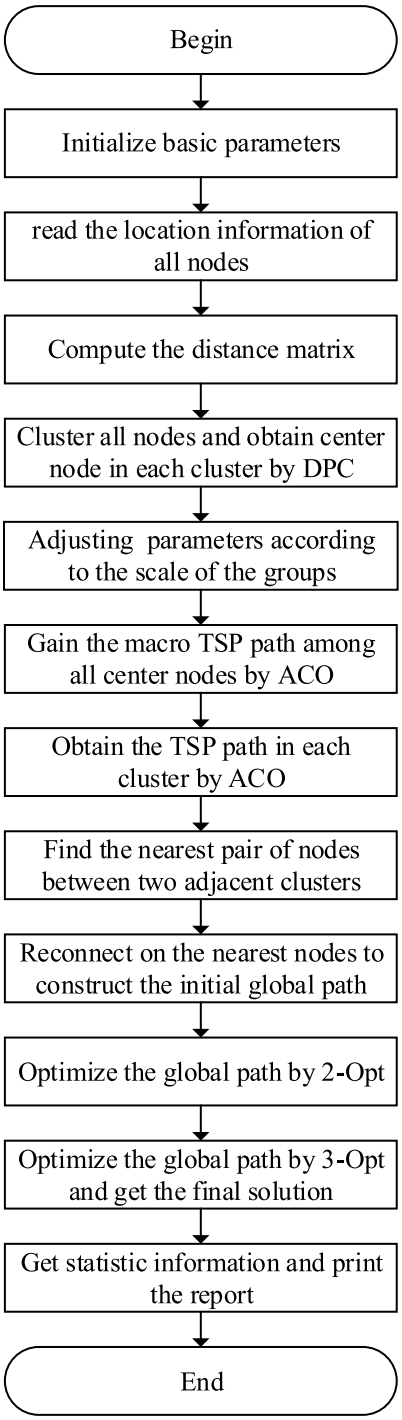


FIGURE 4. The flowchart of the proposed algorithm.

tour, one pair of nodes with the shortest distance between two adjacent groups are disconnected in local tour in the lower layer and reconnected to construct the initial global TSP tour.

The ACO parameters may be different in the process of merging local tours when the number of groups is different with clustering. The parameters need to be determined by experiments for multiplex problem sizes.

In the process of clustering nodes by using DPC algorithm, when the number of nodes increases in each cluster, the number of groups and the clustering time both become less, but it takes more time to find the local tour by using ACO algorithm in each cluster. The total runtime usually becomes longer. In addition, the number of nodes in each group would affect the solution accuracy of the proposed method. Thus, the runtime and the accuracy of the algorithm are closely related to the number of nodes in each cluster.

The method that decomposes the problem into subproblems can improve the solution speed but may produce nonoptimal solution when connecting the local paths among subproblems. Therefore, the solution accuracy should be enhanced by using local path optimization algorithm, such as k-Opt.

IV. EXPERIMENT

The proposed algorithm is carried out by MATLAB 2015b with single thread, the CPU is Intel (R) Core (TM) i7-6700 @3.40GHz, the memory capacity is 8 GB, and the OS is Win7. TSPLIB [46] is the public library of TSP benchmark instances which are commonly used for testing the optimization algorithms. The experiments are divided into three groups according to the problem size: small-scale, large-scale, and very large-scale.

A. THE SCALE OF EACH CLUSTERING

To find the optimal range of the number of nodes in each cluster, the numbers from 30 to 80 with an interval five are candidate scales. These five instances (ch150, kroA200, rd400, d493, and p654) are selected for parameters determination experiments because they are different distribution styles and the nodes are not too many, but wide and uniform distribution. Each candidate number was repeated 20 times for five TSPLIB instances. The changed relative error is computed according to (9).

$$CRE = (L_{ACO} - L_{HHA})/L_{BKS} \times 100. \tag{9}$$

For a certain TSP instance, let L_{HHA} be the average tour length by using the hierarchical hybrid algorithm, oppositely let L_{ACO} be the average tour length by using the ACO and k-Opt algorithms without clustering. Let L_{BKS} be the length of BKS for the same instance. CRE is the percent that the accuracy is improved by the hierarchical hybrid algorithm compared with the algorithm without clustering. CRE greater than zero means that the proposed algorithm has better performance in terms of the accuracy.

$$CT = T_{ACO}/T_{HHA}. \tag{10}$$

Let T_{ACO} be the computational time by using traditional ACO algorithm and let T_{HHA} be the computational time by using hierarchical hybrid algorithm. CT as in (10) is the ratio of the two time. If CT is greater than one, it means the proposed algorithm has less runtime. The runtime will become less as CT increases.

TABLE 1. The relationship between performance and the maximum scale in each cluster.

Candidate scale		30	35	40	45	50	55	60	65	70	75	80
ch150	**CRE**	3.1	**3.43**	2.54	3.02	1.55	1.38	1.45	1.1	1.29	1.8	1.91
	CT	19.33	**19.45**	12.88	12.9	5.54	5.52	5.48	5.42	5.47	3.2	3.19
	EV	329.22	**362.69**	266.49	314.8	160.11	143.86	150.88	115.18	134.85	182.76	194.59
kroA200	**CRE**	**2.41**	1.96	1.55	2.33	1.75	1.73	0.89	1.24	1.26	2.33	2.18
	CT	**26.15**	16.15	15.36	16.99	8.75	8.98	9.03	9.03	6.75	6.67	6.67
	EV	**267.46**	212.09	170.63	250.16	183.3	181.49	97.82	132.6	133.01	239.87	224.51
rd400	**CRE**	1.17	1.49	1.33	**1.84**	1.57	1.29	0.82	0.87	0.12	0.71	0.3
	CT	**127.12**	116.92	78.12	70.64	68.49	69.15	36.28	35.99	24.87	24.48	21.85
	EV	244.02	**265.51**	211.03	254.82	225.74	198.46	118.7	122.95	36.81	95.21	51.47
d493	**CRE**	1.39	1.95	1.7	1.59	2.89	2.65	2.74	3.29	2.97	3.4	**3.76**
	CT	218.27	**326.09**	325.49	324.85	85.61	81.9	81.27	80.66	81.94	71.1	71.02
	EV	357.03	**520.68**	495.65	484.27	374.28	347.32	355.39	410.04	378.96	411.03	446.94
p654	**CRE**	0.67	0.82	0.51	1.12	0.6	**1.46**	1.45	1.32	1.06	0.74	0.91
	CT	**78.62**	77.89	76.63	5.71	5.69	6.08	5.84	6.09	6.06	6.07	5.96
	EV	145.46	**159.69**	127.53	117.25	65.27	152.47	150.45	137.99	111.94	80.05	97.13
Hit counts		1	**4**	0	0	0	0	0	0	0	0	0

In order to obtain unbiased comparisons, *CRE* and *CT* are both obtained under the same external environment, e.g., the same parameters of ACO and the identic k-Opt local optimization algorithm. This mechanism assures the environment is coincident and the performance is comparable between with clustering and without clustering.

Both accuracy improvement and runtime reduction must be comprehensively considered in the assessment. The evaluation function is defined as (11).

$$EV = \eta \times CRE + CT. \tag{11}$$

Let η be the weight coefficient of the accuracy. It means how many times runtime would be consumed when improving one percent accuracy. Generally, η is 100.

The relative error and the runtime based on the test are shown in Table 1. We observed that the accuracy was improved and the runtime was reduced for all instances by using the proposed algorithm. The accuracy improvement was the best for ch150 and kroa200 when the evaluation value was the highest, and the third best for rd400. While the runtime reduction was the most obvious for ch150, kroA200, and d493, and the second best for rd400 and p654. In five instances, the candidate number 35 was hit four times, and the number 30 was hit one time. The result shows that the runtime and the relative error of the proposed algorithm are both in the best range when there are at most 35 nodes in each group. The number of clusters increases as the nodes become more, so does the communication and merge time between groups. In theory, the number of nodes is the optimal for the ACO algorithms in the both upper and lower layers when the nodes are clustered 35 groups and each group has at most 35 nodes on average.

That is, the hierarchical hybrid algorithm with 1225 (35×35) nodes has a strong advantage, especially in term of runtime. But when the number of nodes continually increases, the number of groups increases synchronously. So does the time of communication, integration, and optimization in the process of merging local solutions to the global solution. Consequently, the superiority would gradually decline even lose. The proposed algorithm is tested in three groups according to problem size. The first group is small-scale (50 to 350 nodes), the second group is large-scale (350 to 1200 nodes), and the last group is very large-scale (1200 to 3500 nodes). The instances with different distribution styles and different sizes are selected to comprehensively test the performance of the proposed algorithm. We used 10 different symmetric TSP instances from TSPLIB (eil51, berlin52, st70, eil76, rat99, kroa100, eil101, lin105, ch150, and kroA200) for small-scale problems, 11 different instances (rand400, fl417, pr439, pcb442, d493, rat575, p654, d657, u724, rat783, and pcb1173) for large-scale TSP problems, and nine instances (d1291, nrw1379, fl1400, d1655, rl1889, vm1748, u2152, pr2392, and pcb3038) for very large-scale problems. Every instance is repeated 100 times. Each run stops when the test repeats continually 1000 times.

B. EVALUATION OF THE DPC-ACO-KOpt ALGORITHM ON THE SMALL-SCALE TSP PROBLEMS

There are usually less than 10 groups after using DPC algorithm for small-scale TSP problems. The communication time between groups is very short. The number of nodes for ACO are slightly different between with clustering and without clustering. The runtime of the proposed algorithm decreases, but not obviously.

Since the node distribution and the size in every TSP instance may be different, the optimal values of the α and β may be different in the ACO algorithm. In fact, the parameter β strongly depends on the parameter α [47]. The values of the parameters α and β in ACO are discussed in the literature, such as [32], [48], and [49]. Table 2 lists the best values of the parameters α and β for the small-scale TSPLIB instances in [31]. The same parameter values are selected in our study. Additional parameters for the proposed algorithm are shown in the Table 3.

TABLE 2. The exponential parameter values for the small-scale TSP instances.

TSP Instance	α	β
eil51	1.11	1.44
berlin52	0.95	1.05
st70	0.94	1.05
eil76	0.88	1.50
rat99	0.99	1.07
kroA100	1.01	1.10
eil101	1.20	0.75
lin105	1.20	0.65
ch150	0.75	1.20
kroA200	0.75	1.15

TABLE 3. Other parameter setting of DPC-ACO-KOpt for the small-scale TSP instances.

Parameters	ρ	Q	m
Values	0.1	1	0.6 times the number of the nodes

Generally, the number of ants m is in direct proportion to the number of nodes, and the ratio is closely related to the runtime and the solution accuracy. Computational time increases as the ratio increases. The solution may lose the optimal accuracy when the ratio is below or above the optimal region. The ratio was determined as 0.6 according to the average tour length, the shortest tour length and the runtime based on eil51 instance.

The k-Opt algorithm is used to escape from the local optimal solution. The k-Opt algorithm has a significant effect on improving solution accuracy. The time cost and solution accuracy improvement of k-Opt algorithm are shown in Table 4 for 10 small-scale TSP instances. The optimization effect is measured in (12). It is the percentage of the average tour length decreased by using k-Opt local optimization algorithm versus the length of BKS. The time cost is calculated in (13), and it is the percentage of the k-Opt runtime versus the total runtime with k-Opt.

$$DL_{KOpt} = (L_{avgwithoutKOpt} - L_{avg})/L_{BKS} \times 100. \quad (12)$$

$$TC_{KOpt} = (T_{KOpt}/T_{\text{with}KOpt}) \times 100. \quad (13)$$

We observed that k-Opt algorithm yielded better tour lengths for all instances. The optimization effect was most obvious for rat99, up to 13.14%. In the worst case, the tour length was decreased 2.23% for eil51. The time consuming for k-Opt was below 0.26% and almost negligible.

As previously mentioned, the characters and features of some related algorithms have been briefly stated in Section 1. To test the performance of the hierarchical hybrid algorithm, 11 other algorithms were selected to compare with the DPC-ACO-KOpt algorithm: PACO-3Opt (2016) [29], PSO-ACO-3Opt (2015) [32], ACE (2015) [27], ASA-GS (2011) [24], SA-ACO-PSO (2011) [28], WFA-2Opt (2013) [18], WFA-3Opt (2013) [18], ACO-Taguchi (2013) [30], IFOA (2017) [22], DSOS (2017) [21], and

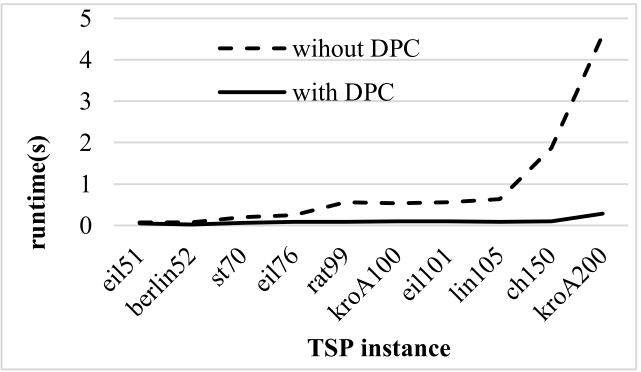


FIGURE 5. Runtime comparison of the proposed algorithm with ACO-KOpt (without DPC) for 10 small-scale TSP instances. DPC-ACO-KOpt outperformed ACO-KOpt without DPC in all test instances. The advantage was more significant when the nodes are more than 105.

C-PSO-ACO-KOpt (2017) [33]. Table 5 demonstrates the comparison of the DPC-ACO-KOpt algorithm with these 11 algorithms for 10 small-scale TSP instances. The quality criteria of the solutions are the average tour length, the standard deviation (SD) and the relative error (RE). The RE is calculated in (14). L_{BKS} is the length of the BKS, L_{avg} denotes the average tour length found by DPC-ACO-KOpt. The better results in the comparison are written in bold and the character '–' means that there is no result in original references.

$$RE = (L_{avg} - L_{BKS})/L_{BKS} \times 100. \quad (14)$$

In terms of the average tour length and RE, DPC-ACO-KOpt algorithm obtained better tours than 11 other algorithms on these instances except kroA100 and lin105. Furthermore, the proposed algorithm generated sub-optimum tours on kroA100 and lin105. WFA-2Opt algorithm yielded better results than the other algorithms on the two instances. The relative error of the proposed algorithm was less than 0.21% for all instances.

For the small-scale TSP instances, the proposed algorithm performs better than the remaining 11 algorithms in 80% of instances ranging from 51 up to 200 nodes (i.e., 8 out of 10 instances). The average relative error was 0.07%, which was obviously less than the 0.14% of C-PSO-ACO-KOpt, 0.21% of ACE, and 0.4% of PACO-3Opt respectively.

The standard deviations of these tours found by the proposed algorithm were minimal for eil51, berlin52, and kroA200. WFA-2Opt had the best standard deviations for kroA100, lin105, and ch150. C-PSO-ACO-KOpt was the most stable for rat99 and eil101, PSO-ACO-3Opt for eil76, and PACO-3Opt for st70. This means the proposed algorithm has more robust solutions than other algorithms.

The proposed algorithm resolves the smaller scale TSP by using ACO after clustering all nodes and decomposing the original TSP problem. The problem size for the ACO is much smaller in the proposed algorithm, and the runtime is shorter in each cluster. Although it spends some runtime on merging the local tours, the total runtime is still shorter. The runtime comparison for 10 small-scale TSP instances is summarized in Fig. 5.

TABLE 4. The time cost and optimization effect of k-Opt algorithm on the 10 TSP instances.

Instance	Without k-Opt			With k-Opt			DL_{kopt}	TC_{kopt}
	Best	Average	Worst	Best	Average	Worst		
eil51	427	435.75	445	**426**	**426.25**	**427**	2.23	0.02
berlin52	7756	8081.5	8612	**7542**	**7542**	**7542**	7.15	0.04
st70	707	735.8	778	**675**	**675.25**	**680**	8.97	0.04
eil76	571	589.5	611	**538**	**538.3**	**541**	9.52	0.03
rat99	1300	1371.05	1433	**1211**	**1211.9**	**1215**	13.14	0.07
kroA100	22412	23481.4	24202	**21282**	**21283.65**	**21305**	10.33	0.07
eil101	659	676.65	715	**629**	**630.35**	**636**	7.36	0.07
lin105	15171	15903.8	16955	**14379**	**14380.1**	**14401**	10.6	0.09
ch150	6882	7146.1	7370	**6528**	**6536.5**	**6573**	9.34	0.26
kroA200	30252	31745.8	33257	**29368**	**29398.4**	**29461**	7.99	0.18

C. EVALUATION OF THE DPC-ACO-KOpt ALGORITHM ON THE LARGE-SCALE TSP PROBLEMS

There are usually no more than 35 groups after clustering by DPC algorithm for large-scale TSP. The communication time between groups is not long. The problem sizes for ACO are clear different between with clustering and without clustering. The superiority of short runtime is significant for these large-scale TSP instances. Eleven cases from TSPLIB with 350 to 1200 nodes were chosen to test. The test environment was the same as that in the small-scale TSP instances. The parameters α and β are the mean values of 10 small-scale instances. The specific values of all parameters are listed in Table 6.

To test the performance of the DPC-ACO-KOpt algorithm in the large-scale TSP problems, eight other algorithms briefly explained in Section 1 were selected to compare with the DPC-ACO-KOpt algorithm: PACO-3Opt (2016) [29], PSO-ACO-3Opt (2015) [32], HMMA (2015) [25], PMSOM (2015) [19], HCACO (2014) [31], HGA (2014) [20], ASA-GS (2011) [24], and DSOS (2017) [21]. The comparison results with eight other algorithms on the 11 large-scale instances are shown in Table 7. DPC-ACO-KOpt algorithm yielded the better solution than eight other algorithms except rat-style instances (rat575 and rat783) in terms of the average tour length and the best tour length. The relative error of the proposed algorithm was less than 0.99% except rat-style instances. It was distinct that ASA-GS was more suitable for rat-style TSP instances than the proposed algorithm. The densities of all nodes in the rat-style instances are almost equal so that the connection between groups could not lead to better tour when merging the local tours, and k-Opt does not remedy the deficiency.

For the large-scale TSP instances, the proposed algorithm outperforms the other algorithms in 81.82% of instances ranging from 400 up to 1173 nodes (i.e., 9 out of 11 instances). The average relative error was 1.45%, which was less than the 1.65 of ASA-GS, 2.4 of PACO-3Opt, and 2.65 of PSO-ACO-3Opt respectively. If excluding the rat-style instances, the average relative error of the proposed algorithm is 0.70 which is much less than 1.45.

D. EVALUATION OF THE DPC-ACO-KOpt ALGORITHM ON THE VERY LARGE-SCALE TSP PROBLEMS

There are usually no more than 100 groups after clustering by using DPC algorithm for very large-scale TSP. The communication time between groups may be long and cannot be neglected. The size of every group after clustering is much smaller than the scale without clustering. Overall, the runtime can be reduced. The test environment and parameters are both the same as them in the large-scale instances.

To test the performance of the DPC-ACO-KOpt algorithm in the very large-scale TSP problems, five other algorithms briefly explained in Section 1 were selected to compare with the DPC-ACO-KOpt algorithm: PMSOM (2015) [19], HMMA (2015) [25], HCACO (2014) [31], ASA-GS (2011) [24], and DSOS (2017) [21]. The comparison results with five other algorithms on the nine very large-scale instances are shown in Table 8.

We observed that DPC-ACO-KOpt algorithm constructed better solution than other remaining algorithms on all very large-scale TSP instances. The relative error of the proposed algorithm was between 0.22% and 1.57%.

For the very large-scale TSP instances, the proposed algorithm surpasses the five other algorithms in 100% of these nine instances in terms of solution accuracy. The average relative error was 0.54%, which was significantly less than the 2.17 of ASA-GS, 6.94 of PMSOM, and 10.96 of HCACO respectively.

Compared with the suboptimum algorithm, the solution accuracy of the proposed algorithm improves 2.0 times for the small-scale problems (i.e., 0.07 versus 0.14), 1.14 times for the large-scale problems (i.e., 1.45 versus 1.65), 4.02 times for the very large-scale problems (i.e., 0.54 versus 2.17) in terms of the average relative error. The proposed algorithm is more applicable for the very large-scale TSP problem than the small-scale and large-scale TSP. This verifies the fact the algorithm has more remarkable advantage for the very large-scale TSP problem.

Overall, the proposed algorithm overcomes other algorithms in 86.67% of the 30 instances ranging from 51 to 3038 nodes (i.e., 26 out of 30 instances). The average relative

TABLE 5. Comparison of the proposed algorithm with other algorithms on the small-scale TSP instances.

Algorithm	avg (RE)	Instance	eil51	berlin52	st70	eil76	rat99	kroA100	eil101	lin105	ch150	kroA200
		BKS	426	7542	675	538	1211	21282	629	14379	6528	29368
Proposed		Best	426	7542	675	538	1211	21282	629	14379	6528	29368
algorithm		Avg.	**426.25**	**7542**	**675.25**	**538.30**	**1211.90**	21283.65	**630.35**	14380.10	**6536.5**	**29398.40**
		SD.	**0.44**	**0.00**	1.11	0.80	1.25	5.50	2.01	4.92	14.36	**39.08**
	0.07	RE (%)	**0.06**	**0.00**	**0.04**	**0.06**	**0.07**	0.01	**0.21**	0.01	**0.13**	**0.10**
PACO-3Opt		Avg.	426.35	7542	677.85	539.85	1217.10	21326.80	630.55	14393.00	6601.40	29644.50
(2016) [29]		SD.	0.49	0.00	**0.99**	1.09	4.01	33.72	2.63	19.76	15.01	53.43
	0.4	RE (%)	0.08	0.00	0.42	0.34	0.50	0.21	0.25	0.10	1.12	0.94
PSO-ACO-		Avg.	426.45	7543.20	678.20	**538.30**	1227.40	21445.10	632.70	14379.15	6563.95	29646.05
3Opt (2015)		SD.	0.61	2.37	1.47	**0.47**	1.98	78.24	2.12	0.48	27.58	114.71
[32]	0.49	RE (%)	0.11	0.02	0.47	0.06	1.35	0.77	0.59	0.00	0.55	0.95
ACE (2015)		Avg.	426.82	7543.04	676.42	538.31	1213.29	21298.60	633.62	14385.5	6550.00	–
[27]		SD.	0.58	13.37	2.69	1.14	4.12	42.46	3.90	26.22	13.61	–
	0.21	RE (%)	0.19	0.01	0.21	0.06	0.19	0.08	0.73	0.05	0.34	–
ASA-GS		Avg.	428.872	7544.37	677.11	544.369	1219.49	21285.4	640.515	14383	6539.8	29438.4
(2011) [24]		SD.	–	–	–	–	–	–	–	–	–	–
	0.51	RE (%)	0.67	0.03	0.31	1.18	0.7	0.01	1.83	0.02	0.16	0.23
SA-ACO-		Avg.	427.27	7542	–	540.20	–	21370.30	635.23	14406.37	6563.70	29738.73
PSO (2011)		SD.	0.45	0.00	–	2.94	–	123.36	3.59	37.28	22.45	356.07
[28]	0.52	RE (%)	0.30	0.00	–	0.41	–	0.41	0.99	0.19	0.55	1.27
WFA-2Opt		Avg.	426.65	7542	–	541.22	–	**21282.00**	639.87	**14379**	6572.13	29654.03
(2013) [18]		SD.	0.66	0.00	–	0.66	–	**0.00**	2.88	**0.00**	**13.84**	151.42
	0.52	RE (%)	0.15	0.00	–	0.60	–	**0.00**	1.73	**0.00**	0.68	0.97
WFA-3Opt		Avg.	426.60	7542	–	539.44	–	**21282.80**	633.50	14459.40	6700.10	29646.50
(2013) [18]		SD.	0.52	0.00	–	1.51	–	**0.00**	3.47	1.38	60.82	110.91
	0.66	RE (%)	0.14	0.00	–	0.27	–	**0.00**	0.72	0.56	2.64	0.95
ACO-		Avg.	435.40	7635.40	–	565.50	–	21567.10	655.00	14475.20	–	–
Taguchi		SD.	–	–	–	–	–	–	–	–	–	–
(2013) [30]	2.45	RE (%)	2.21	1.24	–	5.11	–	1.34	4.13	0.67	–	–
IFOA		Avg.	427.53	**7542**	677.26	–	1237.20	21357	642.05	14427.06	6618.20	–
(2017) [22]		SD.	1.26	**0.00**	2.33	–	15.17	43.77	4.77	44.67	31.68	–
	0.88	RE (%)	0.36	**0.00**	0.34	–	2.16	0.35	2.08	0.33	1.38	–
DSOS		Avg.	427.9	7542.60	679.20	547.40	1228.37	21409.50	650.60	–	–	–
(2017) [21]		SD.	1.20	0.00	2.80	3.90	14.32	149.15	4.57	–	–	–
	1.18	RE (%)	0.45	0.01	0.62	1.75	1.43	0.60	3.43	–	–	–
C-PSO-		Avg.	426.29	7543.29	676	538.15	1213.9	21319.5	631.2	14379.29	–	29642
ACO-KOpt		SD.	0.46	3.9	1.73	0.65	**0.99**	47.79	**1.5**	1.3	–	145
(2017) [33]	0.14	RE (%)	0.07	0.01	0.14	0.02	0.07	0.17	0.34	0	–	0.46

TABLE 6. Parameter setting of DPC-ACO-KOpt for the large-scale TSP instances.

Parameters	α	β	ρ	Q	Maximum Iteration
Values	0.98	1.10	0.1	1	1000

error is 0.75 for all 30 instances (0.07 of small-scale TSP, 1.45 of large-scale TSP, and 0.54 of very large-scale TSP). Therefore, this result supports the fact that the proposed algorithm can realize TSP solutions with high-accuracy and compete favorably with the state-of-the-art TSP algorithms, especially for the very large-scale TSP problem.

V. RESULTS AND ANALYSIS

The better solution accuracy of the proposed algorithm has been demonstrated in Table 5, 7, and 8. The results arise from better strategy, parameter tuning, and local optimization. The solution accuracy after grouping and merging does not decline but surpasses other algorithms. This phenomenon is mainly due to two reasons. One is that the ACO algorithm has higher-accuracy for small-scale TSP problems versus large-scale TSP problems. The other is that the DPC algorithm clustering nodes based on the density is effective for the decomposing of TSP problem. It is appropriate that the proposed algorithm solves the small-scale TSP by using ACO algorithm after DPC algorithm decomposes large-scale TSP into small-scale TSP. The performance is further revealed by descriptive statistical analysis and runtime analysis.

A. DESCRIPTIVE STATISTICAL ANALYSIS ON SOLUTION ACCURACY

The two best algorithms for small-scale instances are the proposed algorithm and C-PSO-ACO-KOpt, while for large-scale and very large-scale instances are the proposed algorithm and ASA-GS. We compare them against BKS using MATLAB statistical package to further validate the performance.

In summary, descriptive statistics reveal these algorithms in terms of mean, standard deviation, minimum, maximum and range. The Levene's test validates whether all the algorithms have the same variance. The one way Analysis of

TABLE 7. Comparison of the proposed algorithm with other algorithms on the large-scale TSP instances.

Algorithm	avg (RE)	Instance	rd400	fl417	pr439	pcb442	d493	rat575	p654	d657	u724	rat783	pcb1173
		BKS	15281	11861	107217	50778	35002	6773	34643	48912	41910	8806	56892
Proposed		Avg.	**15387.25**	**11890.15**	**107752.15**	**51158.95**	**35347.5**	7135.0	**34979.55**	**49317.65**	**42282.7**	9192.1	**57113.7**
algorithm		Best	**15333**	**11870**	**107412**	**51001**	**35237**	7071	**34693**	**49103**	**42118**	9118	**56901**
	1.45	RE (%)	**0.7**	**0.25**	**0.5**	**0.75**	**0.99**	5.34	**0.97**	**0.83**	**0.89**	4.38	**0.39**
PACO-		Avg.	15613.9	11987.4	108702	52202.4	35841	7012.4	35075	50277.5	43122.5	9127.3	–
3Opt		Best	15578	11972	108482	51962	35735	7003	35045	50206	42764	9111	–
(2016) [29]	2.4	RE (%)	2.18	1.07	1.39	2.81	2.4	3.53	1.25	2.79	2.89	3.65	–
PSO-ACO-		Avg.	15691.3	11980.4	108965.4	52368.1	35973.8	7018.6	35098.2	50475.5	43300.3	9138.1	–
3Opt		Best	15594	11947	108530	52131	35789	6987	35052	50291	43172	9128	–
(2015) [32]	2.65	RE (%)	2.69	1.01	1.63	3.13	2.78	3.63	1.31	3.2	3.32	3.77	–
HMMA		Avg.	16723.68	12897.84	116855.91	55670.17	38410.25	7745.63	37931.4	55554.31	47132.71	10256.23	–
(2015) [25]		Best	16534	12543	114095	54401	37187	7575	37044	55163	46662	10149	–
	11.29	RE (%)	9.44	8.74	8.99	9.63	9.74	14.36	9.49	13.58	12.46	16.47	–
PMSOM		Avg.	–	–	–	53362.6	–	–	36399.4	–	43980.35	–	61386.47
(2015) [19]		Best	–	–	–	52631	–	–	35953	–	43704	–	60938
	5.75	RE (%)	–	–	–	5.09	–	–	5.07	–	4.94	–	7.9
HCACO		Avg.	–	–	–	55326	–	–	–	53234	–	9505	–
(2014) [31]		Best	–	–	–	54838	–	–	–	52665	–	9453	–
	8.58	RE (%)	–	–	–	8.96	–	–	–	8.84	–	7.94	–
HGA		Avg.	15852.74	–	109249.66	52376.26	–	–	–	–	–	–	–
(2014) [20]		Best	–	–	–	–	–	–	–	–	–	–	–
	2.93	RE (%)	3.74	–	1.9	3.15	–	–	–	–	–	–	–
ASA-GS		Avg.	15429.8	12043.8	110226	51269.2	–	**6904.82**	–	–	42470.4	**8982.19**	57820.5
(2011) [24]		Best	15351	11940	110020	51064	–	**6872**	–	–	42275	**8954**	57761
	1.65	RE (%)	0.97	1.54	2.81	0.97	–	**1.95**	–	–	1.34	**2**	1.63
DSOS		Avg.	–	–	–	–	–	7117.32	–	–	–	9102.67	–
(2017) [21]		Best	–	–	–	–	–	7073	–	–	–	9045	–
	4.22	RE (%)	–	–	–	–	–	5.08	–	–	–	3.37	–

TABLE 8. Comparison of the proposed algorithm with other algorithms on the very large-scale TSP instances.

Algorithm	avg (RE)	Instance	d1291	nrw1379	fl1400	d1655	vm1748	rl1889	u2152	pr2392	pcb3038
		BKS	50801	56638	20127	62128	336556	316536	64253	378032	137694
Proposed		Avg.	**50911.5**	**56869.05**	**20430.35**	**63066.3**	**337555.55**	**321097.2**	**64575.9**	**381052.05**	**139855.5**
algorithm		Best	**50828**	**56772**	**20236**	**62733**	**337040**	**319706**	**64321**	**379942**	**139445**
	0.54	RE (%)	**0.22**	**0.41**	**1.51**	**1.51**	**0.3**	**1.44**	**0.5**	**0.8**	**1.57**
PMSOM		Avg.	–	–	–	–	–	–	–	402528.47	148351.52
(2015) [19]		Best	–	–	–	–	–	–	–	401659	148186
	6.94	RE (%)	–	–	–	–	–	–	–	6.25	7.62
HMMA (2015)		Avg.	58161.95	–	23752.24	–	–	–	–	–	–
[25]		Best	57380.17	–	23099	–	–	–	–	–	–
	13.86	RE (%)	12.95	–	14.77	–	–	–	–	–	–
HCACO		Avg.	–	–	–	–	–	–	–	421630	–
(2014) [31]		Best	–	–	–	–	–	–	–	419476	–
	10.96	RE (%)	–	–	–	–	–	–	–	10.96	–
ASA-GS (2011)		Avg.	52252.3	–	20782.2	64155.9	343911	–	–	–	141242
[24]		Best	51751	–	20647	63636	342437	–	–	–	140742
	2.17	RE (%)	1.87	–	2.58	2.43	1.75	–	–	–	2.21
DSOS (2017)		Avg.	–	–	–	–	–	–	–	425431.78	–
[21]		Best	–	–	–	–	–	–	–	419246	–
	12.54	RE (%)	–	–	–	–	–	–	–	12.54	–

Variance (ANOVA) test reveals performance difference among all the solutions and tests whenever the parametric assumption is met.

Table 9 and 10 demonstrates the descriptive statistics about the performance of proposed algorithm, C-PSO-ACO-KOpt, and ASA-GS with BKS. The proposed algorithm is averagely smaller than C-PSO-ACO-KOpt or ASA-GS in terms of mean, standard deviation, maximum and range. This supports the fact that the proposed algorithm is better than C-PSO-ACO-KOpt or ASA-GS. The data about the latter two algorithms have wider range and data spread around its mean value, while proposed algorithm has the smaller range and data dispersion around its mean value.

The analysis of equal variance among the three algorithms is based on the Levene's test because it is robust even with departure from data normality. In Table 9, we conclude that

TABLE 9. Descriptive statistics of proposed algorithm and C-PSO-ACO-KOpt compared with BKS on the small-scale instances.

algorithm	mean	SD	Min	Max	range
BKS	8257.8	10233.56	426	29368	28942
Proposed algorithm	8262.27	10240.43	426.25	29398.4	28972.15
C-PSO-ACO-KOpt	8485.24	10905.86	426.29	29640	29213.71

TABLE 10. Descriptive statistics of proposed algorithm and ASA-GS compared with BKS on the large-scale and very large-scale instances.

algorithm	mean	SD	Min	Max	range
BKS	69755.69	89155.21	6773	336556	329783
Proposed algorithm	70287.02	89469.60	7135	337555.55	330420.55
ASA-GS	71345.39	91180.61	6904.82	343911	337006.18

TABLE 11. Equal variance test of suboptimal algorithms and proposed algorithm against BKS based on Levene's test statistic.

Statistic	Levene's	p-value
Wsmall [a]	0.02282	0.97746
Wlarge [b]	0.00288	0.99712
Wvery [c]	0.00121	0.99879

[a] Wsmall is the Levene's statistic and p-value on the small-scale instances.
[b] Wlarge is the Levene's statistic and p-value on the large-scale instances.
[c] Wvery is the Levene's statistic and p-value on the very large-scale instances.

TABLE 12. One way ANOVA test of the difference between proposed algorithm compared and BKS.

Source of variation	SS [a]	df [b]	MS [c]	F	p-value
Between groups	4.3604e+06	1	4.3604e+06	4.2914e-04	0.9835
Within group	5.8932e+11	58	1.0161e+10		
Total	5.8933e+11	59			

[a] SS is the sum of squares
[b] df is the degree of freedom.
[c] MS is the mean sum of squares.

TABLE 13. Runtime comparison of proposed algorithm with other algorithms.

Instance	proposed algorithm	ASA-GS [24]	ACO-Taguchi [30]	HMMA [25]	PACO-3Opt [29]	C-PSO-ACO-KOpt [33]
eil51	3.03	3.91	3.32	7.22	**2.39**	19.91
berlin52	**1.48**	3.83	3.15	8.19	2.1	20.28
st70	**3.89**	5.15	5.38	13.07	6.97	100
eil76	**4.69**	5.5	6.12	15.25	8.18	150
rat99	**5.6**	7.34	7.9	25.36	19.79	200
kroA100	**6.34**	7.14	11.52	25.81	21.1	305.01
eil101	**6.55**	7.42	20.75	26	20.79	200.9
lin105	**5.65**	7.68	21.45	27.92	14.57	320.1
ch150	**5.95**	10.91	33.28	55.75	79.35	334.32
kroA200	**9.46**	14.26	48.48	98.48	213.12	350.12
Average	**5.26**	7.31	16.14	30.31	38.84	200.06

the test results on small-scale, large-scale, and very large-scale instances are found to be statistically insignificant. This suggests the null hypothesis of equal variance cannot be rejected. In other words, proposed algorithm and the suboptimal algorithms have the same equal variance against BKS.

One way ANOVA is conducted to assess the difference between the proposed algorithm and BKS. In Table 12, the test result indicates that the majority of variation is within group and not between groups. The test that explains the difference between the two solutions reveals that the residual is minimal in the variance. Therefore, the ANOVA test is found to be statistically insignificant in the light of the high p-value and low F-statistic. In other words, there is no statistically significant difference between the BKS and the proposed algorithm.

B. RUNTIME ANALYSIS

The runtime is an important performance for optimization algorithm. Let N_c be the number of ACO iteration, while N_c is a constant in the proposed algorithm. Let n denote the problem size. Let m be the number of ants, and m is usually constant or proportional to n. Computational complexity of ACO algorithm without hierarchical is calculated in (15).

$$T(n) = O(N_c \times n^2 \times m) = O(n^3). \quad (15)$$

Let C_m be the maximum number of nodes in each cluster. Generally, C_m is constant and given as 35 in the study. Computational complexity of the proposed algorithm is calculated in (16).

$$T^{HHA}(n) = O(n + N_c \times C_m^3 \times \lceil n/C_m \rceil + n) = O(n). \quad (16)$$

In hierarchical hybrid algorithm, the runtime of grouping nodes by using DPC and merging local solutions are both $O(n)$. All nodes in TSP problem are decomposed into $\lceil n/C_m \rceil$ clusters, and it would take linear time. There are C_m nodes at most in each cluster. Computational complexity of each cluster is $O(N_c \times C_m^3)$. Therefore, the total time complexity of proposed algorithm is $O(n)$.

It is obvious that computational complexity of the proposed algorithm is less than the traditional ACO algorithm. The advantage of short runtime becomes more remarkable as problem size increases. The algorithms with short runtime or high accuracy in Section 1 were selected: ASA-GS (2011) [24], ACO-Taguchi (2013) [30], HMMA (2015) [25], PACO-3Opt (2016) [29], and C-PSO-ACO-KOpt (33) [2017]. We compared the proposed algorithm with these five methods. Table 13 demonstrates runtime comparison as the nodes become more. Obviously, PACO-3Opt was more suitable for eil51 due to parallelization and without clustering. AS problem size increases, the proposed method has more significant advantage in terms of runtime and overcomes other algorithms on all instances except eil51. The average runtime is 5.26 seconds for proposed algorithm, 1.39 times for ASA-GS (i.e., 5.26 versus 7.31), 3.07 times

for ACO-Taguchi (i.e., 5.26 versus 16.14), and 38.03 times for C-PSO-ACO-KOpt (i.e., 5.26 versus 200.06).

VI. CONCLUSION

TSP is a NP-hard problem to traverse all nodes once and only once in the shortest tour. The traditional exact algorithms and heuristic intelligent algorithms have some weaknesses in solving large-scale TSP, such as slow convergence speed in the intelligent algorithms, and long runtime that unacceptable in the exact algorithms. It becomes more noticeable when there are more than 200 nodes. A novel hierarchical hybrid algorithm is proposed in this article, which takes full advantage of the superiority that ACO algorithm in solving small-scale TSP problem and DPC algorithm in clustering large-scale nodes. The nodes are clustered by using DPC algorithm, and the large-scale TSP problem is decomposed into a few subproblems of TSP with small-scale nodes. Then the TSP tour in each group and the tour among all cluster center nodes are resolved by using ACO algorithm. The initial global tour is constructed by merging the local tours at the nearest nodes between two adjacent clusters. Finally, k-Opt optimizes the initial tour to generate the final global solution. The proposed algorithm is validated on 30 TSP instances in three groups. Experimental results show that the proposed algorithm has a significant effect on reducing runtime and has the superiority of higher accuracy and robustness. The advantage of the hierarchical hybrid algorithm would be more significant as the problem size increases. The next research content is exploring the relationship between iteration times of ACO in each group and the local or global optimal tour. Another research direction is studying how to merge more efficiently and effectively the local suboptimal tours to construct the global optimal solution.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for valuable suggestions and comments which helped us improve the paper.

REFERENCES

[1] J. Yang, C. Wu, H. P. Lee, and Y. Liang, "Solving traveling salesman problems using generalized chromosome genetic algorithm," *Prog. Natural Sci.*, vol. 18, pp. 887–892, Jul. 2008.

[2] J. Yan, T. Weise, J. Lässig, R. Chiong, and R. Athauda, "Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the TSP," in *Proc. CIPLS*, Orlando, FL, USA, Dec. 2014, pp. 148–155.

[3] H. Hernández-Pérez and J.-J. Salazar-González, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," *Discrete Appl. Math.*, vol. 145, no. 1, pp. 126–139, Dec. 2004.

[4] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Oper. Res.*, vol. 46, no. 3, pp. 316–329, Jun. 1998.

[5] Ö. Ergun and J. B. Orlin, "A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem," *Discrete Optim.*, vol. 3, no. 1, pp. 78–85, Mar. 2006.

[6] H. Zhou and M. Song, "An improvement of partheno-genetic algorithm to solve multiple travelling salesmen problem," in *Proc. ICIS*, Okayama, Japan, 2016, pp. 331–336.

[7] R. M. F. Alves and C. R. Lopes, "Using genetic algorithms to minimize the distance and balance the routes for the multiple traveling salesman problem," in *Proc. CEC*, Sendai, Japan, May 2015, pp. 3171–3178.

[8] A. E.-S. Ezugwu, A. O. Adewumi, and M. E. Frîncu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Syst. Appl.*, vol. 77, pp. 189–210, Jul. 2017.

[9] Y. Lin, Z. Bian, and X. Liu, "Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing—Tabu search algorithm to solve the symmetrical traveling salesman problem," *Appl. Soft Comput.*, vol. 49, pp. 937–952, Dec. 2016.

[10] M. S. Kiran, H. İşcan, and M. Gündüz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem," *Neural Comput. Appl.*, vol. 23, no. 1, pp. 9–21, Jul. 2013.

[11] C.-B. Cheng and C.-P. Mao, "A modified ant colony system for solving the travelling salesman problem with time windows," *Math. Comput. Model.*, vol. 46, nos. 9–10, pp. 1225–1235, Nov. 2007.

[12] Y. Marinakis and M. Marinaki, "A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 432–442, Mar. 2010.

[13] H. Ghaziri and I. H. Osman, "A neural network algorithm for the traveling salesman problem with backhauls," *Comput. Ind. Eng.*, vol. 44, no. 2, pp. 267–281, Feb. 2003.

[14] K.-S. Leung, H.-D. Jin, and Z.-B. Xu, "An expanding self-organizing neural network for the traveling salesman problem," *Neurocomputing*, vol. 62, nos. 1–2, pp. 267–292, Dec. 2004.

[15] Y. He, Y. Qiu, G. Liu, and K. Lei, "A parallel adaptive tabu search approach for traveling salesman problems," in *Proc. IEEE NLP-KE*, Wuhan, China, Oct./Nov. 2005, pp. 796–801.

[16] X.-H. Luo, Y. Yang, and X. Li, "Solving TSP with shuffled frog-leaping algorithm," in *Proc. ISDA*, Kaohsiung, Taiwan, Nov. 2008, pp. 228–232.

[17] Z. Xiang *et al.*, "Solving large-scale TSP using a fast wedging insertion partitioning approach," *Math. Probl. Eng.*, vol. 2015, Jun. 2015, Art. no. 854218.

[18] Z. A. Othman, A. I. Srour, A. R. Hamdan, and Y. L. Pan, "Performance water flow-like algorithm for TSP by improving its local search," *Int. J. Adv. Comput. Technol.*, vol. 5, no. 14, pp. 126–137, Oct. 2013.

[19] B. Avşar and D. E. Aliabadi, "Parallelized neural network system for solving Euclidean traveling salesman problem," *Appl. Soft Comput.*, vol. 34, pp. 862–873, Sep. 2015.

[20] Y. Wang, "The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem," *Comput. Ind. Eng.*, vol. 70, pp. 124–133, Apr. 2014.

[21] A. E.-S. Ezugwu and A. O. Adewumi, "Discrete symbiotic organisms search algorithm for travelling salesman problem," *Expert Syst. Appl.*, vol. 87, no. 1, pp. 70–78, Nov. 2017.

[22] L. Huang, G.-C. Wang, T. Bai, and Z. Wang, "An improved fruit fly optimization algorithm for solving traveling salesman problem," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 10, pp. 1525–1533, Oct. 2017.

[23] Y. Lu, U. Benlic, and Q. Wu, "A hybrid dynamic programming and memetic algorithm to the traveling salesman problem with hotel selection," *Comput. Oper. Res.*, vol. 90, no. 1, pp. 193–207, Feb. 2018.

[24] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Appl. Soft. Comput.*, vol. 11, no. 4, pp. 3680–3689, Jun. 2011.

[25] W. Yong, "Hybrid max–min ant system with four vertices and three lines inequality for traveling salesman problem," *Soft Comput.*, vol. 19, no. 3, pp. 585–596, Mar. 2015.

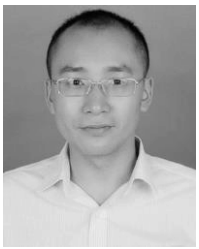
[26] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[27] J. B. Escario, J. F. Jimenez, and J. M. Giron-Sierra, "Ant colony extended: Experiments on the travelling salesman problem," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 390–410, Jan. 2015.

[28] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 14439–14450, Nov./Dec. 2011.

[29] Ş. Gülcü, M. Mahi, Ö. K. Baykan, and H. Kodaz, "A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem," *Soft Comput.*, vol. 22, no. 5, pp. 1669–1685, 2018.

[30] M. Peker, B. Şen, and P. Y. Kumru, "An efficient solving of the traveling salesman problem: The ant colony system having parameters optimized by the Taguchi method," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 21, pp. 2015–2036, Jan. 2013.
[31] J. Jiang, J. Gao, G. Li, C. Wu, and Z. Pei, "Hierarchical solving method for large scale TSP problems," in *Proc. ISNN*, Hong Kong, Nov./Dec. 2014, pp. 252–261.
[32] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem," *Appl. Soft Comput.*, vol. 30, pp. 484–490, May 2015.
[33] I. Khan, M. K. Maiti, and M. Maiti, "Coordinating particle swarm optimization, ant colony optimization and K-Opt algorithm for traveling salesman problem," in *Proc. ICMC*, Haldia, India, 2017, pp. 103–119.
[34] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.
[35] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.
[36] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.
[37] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. ECAL*, Paris, France, 1991, pp. 134–142.
[38] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
[39] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. CEC*, Washington, DC, USA, Jul. 1999, pp. 1470–1477.
[40] A. Vlachos and A. Moue, "Ant colony optimization (ACO) meta-heuristic solving the vehicle scheduling problem (VSP)," in *Proc. WSEAS*, Athens, Greece, Jul. 2006, pp. 812–817.
[41] P. Moradi and M. Rostami, "Integration of graph clustering with ant colony optimization for feature selection," *Knowl.-Based Syst.*, vol. 84, pp. 144–161, Aug. 2015.
[42] O. Dridi, S. Krichen, and A. Guitouni, "A multiobjective hybrid ant colony optimization approach applied to the assignment and scheduling problem," *Int. Trans. Oper. Res.*, vol. 21, no. 6, pp. 935–953, Nov. 2014.
[43] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Comput.*, vol. 21, no. 19, pp. 5829–5839, 2016.
[44] L. Muyldermans, P. Beullens, D. Cattrysse, and D. Van Oudheusden, "Exploring variants of 2-Opt and 3-Opt for the general routing problem," *Oper. Res.*, vol. 53, no. 6, pp. 982–995, Nov. 2005.
[45] A. Levin and U. Yovel, "Nonoblivious 2-Opt heuristics for the traveling salesman problem," *Networks*, vol. 62, no. 3, pp. 201–219, Oct. 2013.
[46] G. Reinelt, Berlin, Germany. *Discrete and Combinatorial Optimization*. Accessed: May 23, 2017. [Online]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
[47] H. Duan, G. Ma, and S. Liu, "Experimental study of the adjustable parameters in basic ant colony optimization algorithm," in *Proc. CEC*, Singapore, Sep. 2007, pp. 149–156.
[48] Z.-F. Hao, R.-C. Cai, and H. Huang, "An adaptive parameter control strategy for ACO," in *Proc. ICMLC*, Dalian, China, Aug. 2006, pp. 203–206.
[49] T. Stützle *et al.*, "Parameter adaptation in ant colony optimization," *Auton. Search*, vol. 6, no. 1, pp. 191–215, Oct. 2011.



ERCHONG LIAO (M'17) received the B.S. degree in computer science and technology from the North China Electric Power University, Baoding, China, in 2005, and the M.S. degree in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 2007. He is currently pursuing the Ph.D. degree in intelligent robot with the North China Electric Power University, Beijing, China.

Since 2007, he has been a Lecturer with the School of Control and Computer Engineering, North China Electric Power University, Baoding. His main research interests include robot path planning and robotic vision.

CHANGAN LIU received the B.S. degree from Northeast Agricultural University in 1995, and the M.S. and Ph.D. degrees from the Harbin Institute of Technology in 1997 and 2001, respectively. He is currently a Professor and the Director of the Intelligence Robot Institute, North China Electric Power University. His research interests focus on the technology of intelligent robot and the theory of artificial intelligence.