Eng Int Syst (2020) 4: 197–204
© 2020 CRL Publishing Ltd

Engineering
Intelligent Systems

An Improved Algorithm for Single-Cluster TSP Based on ACO

Jianbing Lin[1]* and Zhixiong Chen[2]

[1]*College of Information Engineering, Putian University, Putian 351100, China.*
[2]*College of Mathematics, Putian University, Putian 351100, China.*

Significant distributing characteristics of nodes in a graph may lead to different convergence and affect the performance of the Travelling Salesman Problem (TSP). Regarding this, the concepts of domain and density are defined in view of the dense local distribution of nodes in a graph in TSP, and an improved Ant Colony Optimization (ACO) algorithm named SC-ACO is proposed for TSP using a single cluster feature. In this article, the basic principles and strategies of an SC-ACO algorithm are introduced, and the pheromone and next node selection probability are processed via the final domain to distinguish the different behaviors of ants inside and outside the regions with dense nodes. The specific construction process of the SC-ACO algorithm is then described in detail. Finally, this article creates simulated experiments for TSP using SC-ACO and ACS algorithms, with test results showing that SC-ACO has obvious advantages over the ACS algorithm in solving TSP problems that have a single cluster and a large scale of nodes.

Keywords: cluster; TSP; SC-ACO; domain; density

1. INTRODUCTION

As a classical combinatorial optimization problem, the Travelling Salesman Problem (TSP) has been widely used in path planning, computer networking, circuit design and other fields. Up until now, no complete solution to TSP has been found, which makes TSP a NP-complete problem [1,2]. When solving TSP, with the increase of the number of the nodes the corresponding solution space increases exponentially, and the amount of computation increases dramatically. Due to the excellent performance of some bio-based swarm intelligence meta-heuristic algorithms in solving TSP problems, many scholars have done extensive research in this field in recent years, and have achieved gratifying results [3–5]. The Ant Colony Optimization (ACO) algorithm is one of the representative algorithms, which is inspired by the foraging process of ants. In the process of food searching, ants secrete pheromones to record the path they take, and then other ants choose the shortest paths to find food according to the concentration of pheromones on the path. Early ACO algorithms generally had shortcomings, such as a slow convergence speed and becoming trapped in a local optimum, when solving TSP problems [6,7]. In the framework of such ant colony systems, many scholars have tried different methods in the application of specific TSP problems or combined with other related algorithms, and many new achievements have been reached. Morteza and Hadi used ACO and a dynamic reconfigurable graph to simulate the TSP problem with the same number of nodes, and achieved 11.2% and 6.8% better performance than AS and ACS, respectively [8]. In 2017, Darren M.C. proposed a parallel algorithm based on multi-CPU and the reduction of pheromone matrix dimensions, and achieved a 12-fold time reduction when it was applied to solve the TSP problem with a scale of 200,000 nodes [9]. Some scholars have improved the selective probability of the sub-paths forming potential optimal solutions for the pheromone updating mechanism of the ant colony system to speed up the convergence of the algorithm [10, 11]. To address the problem that the basic ant colony algorithm takes too long to deal with a medium-scale TSP, a dynamic self-adaptive ant colony algorithm based on MapReduce is proposed [12]. This algorithm can dynamically adjust pheromone volatilization coefficient in a pheromone updating strategy, so that the ant colony can self-adaptively find a better path, and the iteration part of the ant

*E-mail: liniss@126.com; Tel.: 0086-594-2825091

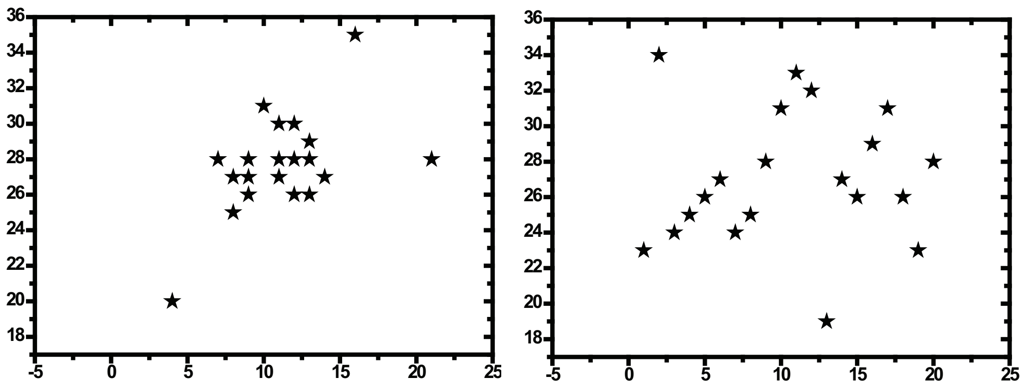

Figure 1 The different distribution characteristics of 20 nodes.

colony algorithm is parallelized by the MapReduce computing model. Finally, this algorithm is deployed and run on the Hadoop platform, and achieves the ideal results that it reduces the average running time to nearly half when compared to the basic ant colony algorithm. In 2017, a multi-objective balanced traveling salesman problem model was proposed for the first time [13], this was used to model the optimization problem with multiple traveling salesman and multi-tasking, and was thereby applied to practical problems with multiple objectives or individuals. It has been validated with the data of TSP problems from small-scale to large-scale, and good results have been obtained.

Although most of the existing biological swarm intelligent meta-heuristic algorithms including ACO adopt parallel algorithms, they are still unable to solve the computational problems caused by large-scale nodes. Researchers have been searching for more efficient algorithms based on biological swarm intelligence for many years to delve into deeper areas [14]. In practical applications, the distribution of nodes in some graphs shows certain characteristics that have a great influence on the solution space [5, 15]. Clustering is used to describe the distribution characteristics of TSP nodes exactly. If the node distribution of a TSP problem conforms to the cluster characteristics, the corresponding solution space will be compressed, which will improve the efficiency of the optimization process of TSP problem, eliminate some of the unnecessary optimization processes and further optimize the solution of the TSP problem that meets the requirements; thereby enhancing the efficiency of the algorithm. In order to solve TSP problems, the distribution characteristics of nodes in a graph are analyzed and studied in this article, and the concepts of domain and density are proposed to determine whether the graph of a given problem conforms to the characteristics of a single cluster. On this basis, the ACO algorithm of the TSP problem with a single cluster feature is improved, named as single cluster-ACO (SC-ACO), and finally simulation tests and comparative analysis are conducted.

2. DEFINITION OF DOMAIN AND DENSITY

Generally, the distribution of nodes in a TSP problem is random, and the distance and direction between nodes have no specific law. However, some nodes show certain characteristics from the overall distribution. Therefore, it is reasonable to study the distribution characteristics of nodes from a general perspective. In order to describe the overall distribution characteristics of nodes, the concepts of domain and density are proposed below.

Domain refers to a square area of a plane, represented by SD. A specific domain SD_i can be described as Formula (1):

$$SD_i \ \ [O(x, y), r], \ i = 1, 2, \ldots, n \tag{1}$$

where O is the origin of the domain, r is the radius of the domain (half of the side length of the square), and $(x,\ y)$ are the coordinates corresponding to the origin.

Density is an index to describe the distribution of nodes in a domain, represented by ρ. The corresponding density (ρ_i) of a domain SD_i can be expressed by Formula (2):

$$\rho_i = Num_i / Area_i \tag{2}$$

where Num_i is the number of nodes in domain SD_i and $Area_i$ is the area of domain SD_i.

For the convenience of calculations, $Area_i$ is usually further processed as: $Area_i$ = sqrt ($Area_i$), where sqrt is a square root function. According to the above definitions, if a node in the graph is taken as the origin of the domain, the number of nodes in domains with different radius and area will be different. For a limited number of nodes, a suitable origin and radius can be found, and the domain they constitute will be the largest domain $SD_{\max}$, which contains all the nodes in the graph.

For example, two kinds of node distributions with 20 nodes (if not specified, the units of co-ordinate axis in this paper are all integer) are shown in Fig. 1. Fig. 1A shows a graph containing densely distributed nodes, while Fig. 1B shows a graph containing relatively scattered nodes. According to the definition of domain, domain SD_1 [O (10, 28, 5] and SD_2 [O (10, 28, 5] is selected for Fig. 1A and 1B, respectively (Fig. 2).

The density of nodes in Fig. 2A and 2B is calculated following the definition of density. Although $r = 5$ and Area $= 100$ in both domains, domain SD_1 contains 17 nodes ($Num_1 = 17$), while SD_2 only contains 8 nodes ($Num_2 = 8$). Therefore, the corresponding densities of Fig. 2A and 2B are $\rho_1 = Num_1/\ Area_1 = 17$ / sqrt(100) = 1.7 and $\rho_2 = Num_2/\ Area_2 = 8$ / sqrt(100) = 0.8. It can be seen that, the nodes in

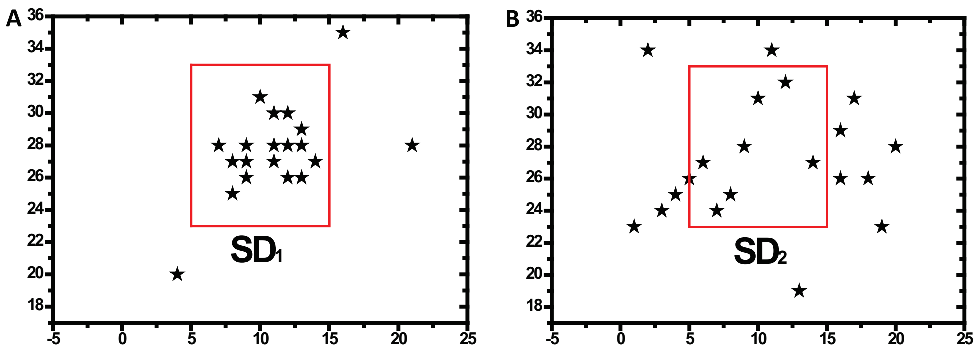

Figure 2 A suitable domain was selected for differently distributed nodes.

the domain SD_1 are denser than that of SD_2. In the case of an irregular distribution of nodes, density can be used to describe some general characteristics of node distribution in the same conditional domains. If the density of a given domain reaches a specific value, the nodes in a given domain form clusters, and the node distribution characteristics of the corresponding TSP have cluster features. The special value is then called as a threshold. For the TSP problem with a large number of nodes, the node distribution may display multiple clusters. This article is limited to studying the TSP problem with only a single cluster feature, those with multi-cluster feature will be analyzed in another study.

3. BASIC ALGORITHM FRAMEWORK OF TSP WITH A SINGLE CLUSTER

The main goal of solving the TSP problem is to find the optimal or better path for a given graph, which traverses all nodes in the graph. The traditional TSP algorithm does not consider the influence of node distribution on the construction process and result of solution from an overall view. In this paper, it is assumed that the distribution of nodes in the graph, i.e. the set domains and the corresponding densities, has an important influence on the construction process and the final results of TSP solutions. Therefore, the corresponding algorithms need to deal with the influence accordingly. In order to achieve this goal, a weighted undirected graph of a TSP problem with a complete single cluster is represented by $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set with $n = |V|$ nodes, and E is a set of edges that are fully connected to these nodes. Each edge has a weight $d(i, j)$, which represents the distance between node i and node j.

The goal of the improved ACO algorithm for a single cluster TSP problem is to find the shortest loop in graph G, which can traverse every node in graph G once and only once. That is to find out the permutation with the smallest sum of all weights in all the permutations π of all nodes in graph G. The objective function is shown as Formula (3):

$$\min f(\pi) = d(v_{\pi(N)}, v_{\pi(1)}) + \sum_{i=1}^{n-1} d(v_{\pi(1)}, v_{\pi(i+1)}) \qquad (3)$$

In the classical ACO algorithm, with the increase of the number of nodes in the graph, the computation for the problem increases sharply, and the performance of the corresponding algorithm decreases. Following the size and overall distribution of nodes, the characteristics of node distribution by setting domains with different sizes and numbers are described and the classical ACO algorithm is improved, so as to optimize the performance of the algorithm with a single cluster feature.

According to the definitions mentioned above, the shape of the domain SD is square, and the size of the domain is then identified by the radius r (1/2 of the side length). The origin of the domain is identified by the origin co-ordinate O (x, y). When $r = 0$, the specified domain SD is a point, when $r = +\infty$, the domain is the whole plane. Obviously, if r is relatively small, the corresponding domain contains fewer nodes, and the optimal f (π) is easy to achieve. With the increase of r, it will be increasingly difficult to find the optimal f(π) for the corresponding domain. According to the researches of a number of ant colony correlation algorithms [2,4,5], it is known that artificial ants are prone to fall into a local optimum in the process of searching for the optimal solution, so it is almost impossible to find a solution that is optimal in all domains. A possible way is to find the sub-optimal solution and then iterate through multiple cycles, in order to obtain an approximate global optimal solution or a better solution. For the case shown in Fig. 2A, the density of nodes in the defined domain is obviously higher than that shown in Fig. 2B. If an artificial ant visits a node in the defined domain shown in Fig. 2A, it should continue visiting all other points within the domain before visiting the nodes outside the domain, so that the paths remains shorter, and thus it is possible to obtain the global optimal situation or a better solution.

In the next section, whether the distribution of nodes constitutes a single cluster feature is determined according to the distribution of node sets in the graph, and the initial action of artificial ants and some corresponding measures to be taken in the process of path searching are then determined via the domains and densities, so as to construct an improved algorithm for the ACO with a single cluster feature (SC-ACO).

4. CONSTRUCTION OF SC-ACO

4.1 Construction of Domain

The corresponding node set in graph G is set as $V = \{v_1, v_2, \ldots v_n\}$. For each node $V_i (i = 1, 2, \ldots, n)$, the corresponding position is expressed by $v_i(x_i, y_i)$, where x_i and y_i

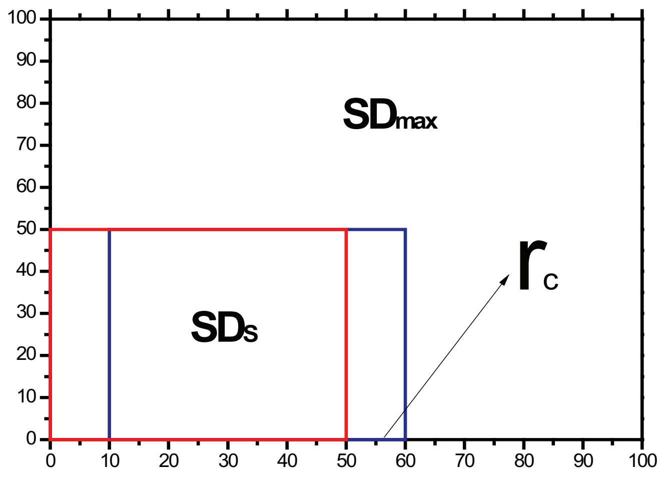

Figure 3 Scanning process of SD$_s$ to SD$_{\max}$.

are corresponding to the abscissa and ordinate on the plane, respectively [16]. Setting $x_{\max} = \max(x_1, \ldots, x_n)$, $x_{\min} = \min(x_1, \ldots, x_n)$, where max () means the maximum function in $x_1, \ldots, x_n$, while min () represents the minimum function in $x_1, \ldots, x_n$. Similarly, setting $y_{\max} = \max(y_1, \ldots, y_n)$ and $y_{min} = \min(y_1, \ldots, y_n)$, and then the largest domain can be expressed as the following formula:

$$SD_{\max}[O(x_{\min} + x_{\max})/2, (y_{\min} + y_{\max}/2), r_{\max}] \tag{4}$$

where r$_{\max}$ = $\max((x_{\max} - x_{\min})/2, \ (y_{\max} - y_{\min})/2)$. The density of the corresponding maximum domain can be calculated as Formula (5):

$$\rho_{\max} = n/sqrt(4r_{\max}^2) = n/2r_{\max} \tag{5}$$

After the SD$_{\max}$ is obtained, an initial scanning field SD$_s$ is needed. Generally, the radius of the initial scanning field r_s is set as half of the radius of the largest domain (r$_{\max}$), that is, $r_s = r_{\max}/2$. Subsequently, the maximum domain is scanned comprehensively by scanning the scanning fields in different positions to obtain different densities. By comparing these densities with the given threshold, the cluster feature can be determined.

4.2 Scanning Strategy

For the scale of graph G, an integer (r_c) is selected as the stepping increment in the scanning field, so that r_c = int($r_{\max}$/C), where C is the stepping coefficient and int () is the bracket function. The stepping co-efficient determines the stepping increment, and it can be adjusted according to the results from the simulation test. There are two ways to scan: one is a transverse scan, and the other is a longitudinal scan. In this study, the main focus is specifically on the transverse scan. For simplicity, the maximum domain of the problem is set as SD$_{\max}$ [O ((0,0)], 100]. As previously mentioned, the radius of the initial scanning field SD$_s$ is half of that of the maximum domain, that is, $r_s = r_{\max}/2 = 100/2 = 50$. Selecting the stepping co-efficient C = 10, the stepping increment rc = int(100/10) = 10. At the beginning of the transverse scan, the apex of the lower left corner of the scanning field is set to overlap that of the maximum domain in order to fix the maximum domain (Fig. 3). After scanning the first scanning field, the scanning field is moved transversely by one incremental unit for the second scanning. When the right side of the scanning field overlaps that of the maximum domain, the first scanning line is completed. Subsequently, the apex of the lower left corner of the scanning field is set to overlap the coordinates of the lower left corner of the second line (0, r_c), i.e. (0, 10), to continue the scan of the second scanning line transversely. In this way, the whole domain is scanned. Thus, the frequency of scans (S$_{num}$) can be deduced as Formula (6):

$$S_{\min} = (\text{int}((2r_{\max} - 2r_s)/r_c) + 1)^2 \tag{6}$$

Shown in Fig. 3 the red box is the location of the first scan in the scanning field, and the blue box is the location of the second scan in the scanning field.

4.3 Determination of Cluster Features

In the process of scanning the maximum domain by the scanning fields, the density of each scanning field is calculated. After the scanning is completed, the maximum of all of the densities will be selected and recorded as $\rho_{\max}$, and the domain containing the maximum density is the final domain, called SD$_{\text{final}}$. The number of nodes in the final domain is NUM$_{\text{final}}$. According to the characteristics of the problem, a density threshold ρ_{th} is set. If $\rho_{\max} \geq \rho_{\text{th}}$, it is assumed that the pattern is clustered, otherwise it is not clustered. For those cases that do not constitute cluster features, the stepping co-efficient can be adjusted and the scanning process can be repeated to determine whether the domains constitute cluster features. If several densities are higher than the threshold value in the scanning process, the area of the scanning domain can be increased and the scanning process can be repeated. The specific increment of the new scanning field is determined according to the new density values. If a new maximum density is greater than or equal to the density threshold and the other density values are less than the density threshold, the increment is the final increment. If there is more than one density greater than the threshold, the area of the scanning field is increased continually and the above process is repeated. The choice of density threshold is determined according to the scale of the problem and the effect of simulation experiment.

4.4 Behavior of Ants in the Domain SD_{final}

After confirming that the graph has cluster characteristics and the final domain is obtained, the behaviors of ants inside and outside the final domain are given below, and the key steps of SC-ACO algorithm are also given.

In the SC-ACO algorithm, artificial ants randomly select a node in the final domain as the starting point. Assuming that the starting point is node i, the probability of ant k choosing j as the next node is given as Formula (7):

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{i\in N_i^k}[\tau_{ij}]^\alpha [\eta_{ij}]^\beta} \tag{7}$$

where $\eta_{ij} = 1/\mathrm{d}_{ij}$ is pre-given heuristic information and d_{ij} is the distance between the node i and the node j. As opposed to the classical ACO algorithm, N in Formula (7) is the final domain, that is, the next node selected by ants can only be in the final domain.

Ants release pheromones when moving in the final domain. Assuming τ_{ij} is the pheromone between node i and node j, the update of pheromone τ_{ij} is given by Formula (8):

$$\tau_{ij}(t+1) = (1-\lambda)\tau_{ij}(t) + \Delta\tau_{ij} \tag{8}$$

where λ is the volatilization co-efficient of pheromone, and $0 < \lambda < 1$. $\Delta\tau_{ij}$ is defined by Formula (9).

$$\Delta\tau_{ij} = \begin{cases} I/C, & \text{edge}(i,j) \text{ in path } T \\ 0, & \text{elese} \end{cases} \tag{9}$$

where C is a constant, whose value can be the path length obtained by a certain algorithm for TSP. After traversing all the nodes in the final domain, the ants randomly select a node outside the final domain and continue to access other nodes outside the final domain according to the classical ACO algorithm. When all the nodes out of the final domain have been visited, it returns to the starting point in the final domain, thus the ants complete a path optimization process. In the next section, specific data are used to simulate the SC-ACO algorithm, and the results are compared with that of ACS algorithm, one of the classical ACO algorithms.

5. SIMULATION TEST AND RESULT ANALYSIS OF SC-ACO

In this section, based on the above-mentioned cluster characteristics of the TSP problem, the definition of domain and density, combined with the specific construction process of SC-ACO algorithm strategy, is applied to carry out the experimental simulation and result analysis of the algorithm. In this process, an ACS simulation test is also carried out for comparative analysis. All simulation tests are conducted on a personal computer (PC) with 8GB RAM and a 3.40 GHZ quad core Intel CPU. The instantiation process of the SC-ACO algorithm is first described, and the path optimization process are then introduced in detail, with some key indicators and their operation methods also explained. Finally, the different results of the indicators are analyzed and compared.

5.1 Initializing Process of SC-ACO

In the initialization stage, according to the scale of the problem, the co-ordinate distribution of nodes is used to generate the basic distribution of nodes on the plane map by the corresponding software. Based on the distribution of nodes, the radius (r_s) and stepping coefficient (C) of the scanning field are determined. Usually, the selection of r_s and C should make the scanning field have a larger density, that is, to make the scanning field contain as many nodes as possible. The final choice of C is determined by the density of the experimental results. Table 1 shows the corresponding co-ordinate data of 50 nodes. According to the above description, the maximum domain of the problem (SD_{max} [O ((99, 51)), 98]) can be obtained by calculation, and the radius of the initial scanning domain is $r_s = r_{max}/2 = 98/2 = 49$. Assuming that the stepping coefficient $C = 10$, the stepping increment $r_c = \mathrm{int}(98/10) = 9$. The total number of scans required for the problem is $S_{num} = (\mathrm{int}(98/9) + 1)^2 = 121$. Before scanning, the scan field is SD_s [O ((1, 4)), 49]. In the process of scanning, 121 density values need to be calculated, and the maximum density value is $\rho_{max} = 0.40$. The density threshold set in this problem is 0.36, so the graph of the TSP problem constitutes a cluster feature, and the final domain SD_{final} [O ((55, 4)), 49] can be obtained.

After confirming that the graph has a cluster feature and final domain, the original data needs to be stored. The node data in the final domain and outside the final domain are stored in different parts of the newly generated data file. The node data in the final domain is located in the front part of the newly generated data file, while the node data outside the final domain is located in the back part of the file. The co-ordinates of all nodes in the data file are stored in a new co-ordinate array, in order for the corresponding distance matrix between the nodes to be calculated. After calculating the distance matrix, the SC-ACO algorithm needs to initialize pheromones between nodes. There are different strategies for the initialization of the pheromone between nodes located inside and outside the final domain. In general, higher initial pheromones are allocated between nodes inside the final domain, while the initial values of pheromones between nodes outside the final domain are calculated following the classical ACO approach. The initial value of a pheromone between nodes inside the final domain is selected according to the density value, which is usually a function that is positively correlated with the density of the domain. The function can be adjusted according to the simulation results, but its initial value must be higher than that of a pheromone outside the final domain. The final stage of initialization is to determine the initial position of ants. Ants can be placed at any node in the final domain initially, and the number of ants depends on the size of the problem set. A reasonable number of ants for

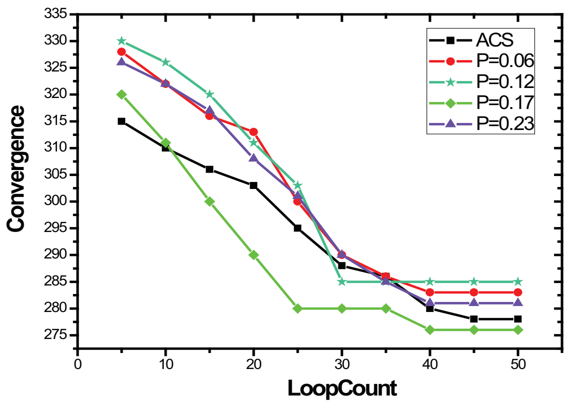

Figure 4 Simulation result of SC-ACO with different probability increments.

various classical ACO algorithms is given in detail by Dorigo M [6] *et al.* and Stutzle T *et al.* [7]. Unless specifically stated, the parameters of relevant simulation experiments are set according to the rules described previously [6,7].

5.2 Path Searching Stage

SC-ACO enters the path searching phase after initializing the parameters required by the algorithm. In this stage, each ant randomly selects a node from the nodes inside the final domain as the starting node, and then calculates the probability of each path from the starting node to the optional node according to Formula (7), and then chooses the next node according to roulette. In the process of traveling to the next node, ants release a certain amount of pheromones. In order to compare with the ACS algorithm, the release rule of pheromones is the same as that of the ACS system. In the process of selecting the next node, ants will determine whether the node is located in the final domain. If the node is located in the final domain, the ants will increase the probability calculated according to Formula (7) by a probability increment to indicate that the nodes in the final domain have priority in the next path selection, and the probability increment value is positively correlated with the value of density. If the starting node and the optional node are not located in the final domain at the same time, the probability of the ant choosing the next node is still calculated according to Formula (7). The ant will repeat the above actions until all the nodes in the graph are selected, and finally the ant will return to the starting node to complete the cycle. After that, the ant will re-initialize and continue the path searching process, compare the results with the previous search results, and save the better results. The above process will circulate all the time and the number of cycles can be controlled when initializing.

5.3 Simulation Results and Analysis

In the SC-ACO algorithm, C $= 10$ and $\rho_{\text{th}} = 0.36$ are set to simulate the TSP problem with 50 nodes. The co-ordinate data of the nodes are shown in Table 1. When the initial value of the pheromone between nodes in the final domain is 0.3, the initial value of the pheromone between nodes in the other domain is 0.1, the number of ants is 50, $\alpha = 1$, $\beta = 3$, and the number of cycles is 50, the simulation results of SC-ACO with different given probability increments are illustrated in Fig. 4.

The abscissa represents the number of cycles and the ordinate indicates the final convergence value. The black line represents the results of the ACS algorithm, the red line represents the results of SC-ACO for the probability increment of $p = 0.06$, the dark green line represents the results of SC-ACO for the probability increment of $p = 0.12$, the bright green line represents the results of SC-ACO for the probability increment of $p = 0.17$ and the blue line represents the results of SC-ACO for the probability increment of $p = 0.23$.

For several different probability increments, SC-ACO can all converge to a lower value. With the increase in the number of cycles, SC-ACO can converge to a quite satisfactory value even if it encounters a relatively small probability increment (e.g. $p = 0.06$). Further simulation tests show that the probability increments in the interval [0.06, 0.23] are more suitable. However, compared with the ACS algorithm with the same node data and parameter settings, the SC-ACO algorithm has no advantage either in calculating time or in convergence value. This can be explained by analyzing the specific operation process of the SC-ACO and ACS algorithms. For the data shown in Table 1, the final domain of the SC-ACO algorithm contains 39 nodes, and the remaining nodes are randomly distributed outside the final domain. Due to the relatively small number of nodes, ants and cycles, the differences in the pheromones of the paths are obvious after several searching cycles, which lead to the ants of the SC-ACO and ACS algorithms having the same behavior in the final domain. Thus, the simulation results of the SC-ACO and ACS algorithms have no obvious difference. However, with the increase of the number of cycles, further simulation results show that the time consumed by SC-ACO is much less than that of the ACS system. The reason is, after many cycles, the pheromone concentration of each path in the final domain increases with the effect of the probability increments, which reduces the computational amount of roulette probability selection by ants through pheromones [16]. As the ACS algorithm cannot give additional probability increment to the nodes in the final domain in the calculation, it is impossible to reduce the whole operation time by reducing the amount

Table 1 Coordinates of fifty nodes.

x	y	x	y	x	y	x	y	x	y
120	45	82	36	114	20	75	65	185	29
116	64	70	22	96	18	103	53	197	90
53	26	74	32	128	43	132	55	118	21
28	71	65	51	86	31	90	53	93	56
27	62	67	48	89	55	104	69	72	59
78	14	79	13	133	19	121	7	142	98
97	51	119	14	73	4	122	8	157	70
105	45	110	20	100	23	127	24	170	66
1	68	108	57	131	49	125	55	123	47
21	23	130	24	95	48	80	41	71	22

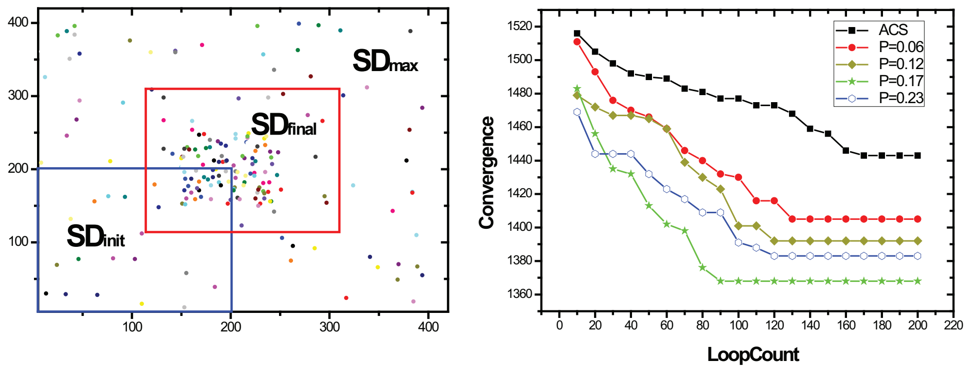

Figure 5 Simulation results of SC-ACO and ACS with 200 nodes.

of computation in the related processes. According to the literature, with the increase of the number of nodes, ants and cycles in graph, the ACO algorithm is likely to fall into a local optimum or search stagnation, causing the related performance to decline sharply [17,18]. To this end, the operation of the SC-ACO algorithm was further tested by increasing the number of nodes and ant data and adjusting the corresponding parameters.

The distribution of nodes in a graph with a single cluster and containing two hundred nodes is shown in Fig. 5A. From the co-ordinate data of nodes, the radius can be calculated as follows: $r_s = r_{\max}/2 = 196/2 = 98$, $r_c = 19$, and the final domain ($\mathrm{SD_{final}}$) is [O ((214, 216)], 98]. Fig. 5A also shows the scanning results of SC-ACO after initialization. When $C = 10$, $\rho_{th} = 0.36$, $\alpha = 1$, $\beta = 3$, the initial pheromone value between nodes inside the final domain is 0.3 and the initial pheromone value of nodes outside the final domain is 0.1, the number of ants is 200 and the number of loops is 200, SC-ACO algorithms with different given probability increments give significantly different simulation results from the ACS algorithm (Fig. 5B).

Fig. 5A shows the graph of 200 nodes and domains, with the black box representing the maximum domain $\mathrm{SD_{max}}$, the blue box being the initial scanning field $\mathrm{SD_{init}}$, and where the red box is the final domain $\mathrm{SD_{final}}$; Fig. 5B shows the simulation results of the SC-ACO and ACS algorithms.

In Fig. 5A, the nodes are densely distributed and cluster-like in the field between 100 and 300 in both the X-axis and Y-axis, so the final domain of the SC-ACO algorithm includes these dense nodes after initialization, and processes these nodes according to the pheromone strategy and probability increment strategy prescribed by the SC-ACO algorithm. They are then tested following the pre-set parameters and compared with the results of the ACS algorithm. Fig. 5B shows that the convergence results of SC-ACO algorithms with different probability increments are better than that of ACS and have advantages in calculating time. Such results can be further improved as the number of cycles increase. When the current node is located inside the final domain, although the ants in ACS system can select the next node from the final domain by pheromone and path heuristic information at a more probable rate, it is easy for the ACS algorithm to fall into a local optimum and search stagnation with the increase of the number of cycles as the volatilization coefficient weakens the interaction of pheromone concentration and path heuristic information [19, 20]. However, the SC-ACO algorithm has an additional probability increment to offset such effects, which leads to the significant difference in the simulation results between the two algorithms.

6. CONCLUSIONS

In this paper, the representation of domain and density for the TSP problem with a single cluster feature was introduced. On this basis, the ACO algorithm was improved and a SC-ACO algorithm was proposed. The framework strategy and specific construction process of the SC-ACO algorithm were described in detail in this paper, and the SC-ACO algorithm was then tested with simulation data and compared with the ACS algorithm. The results show that the SC-ACO algorithm took less time calculating and had a better convergence performance than ACS on TSP problems with a large scale set and a graph with obvious single cluster characteristics. In the future, the way SC-ACO can be applied to the TSP problems with multi-cluster characteristics will be studied, and other additional factors that restrict the convergence speed and value of the SC-ACO algorithm can be found; as well as the sensitivity issues of data, so that the performance of the SC-ACO algorithm can be further improved and have a larger scope of application.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (61772292), the Natural Science Foundation of Fujian Province, China (2019J01814), and the Science and Technology Project of Putian, China (2018GP2004).

REFERENCES

1. Yang J. Y., Ding R. F., Zhang Y., *et al.* An Improved Ant Colony Optimization (I-ACO) Method for the Quasi Travelling Salesman Problem (Quasi-TSP). International Journal of Geographical Information Science, 2015, 29(9), 1534–1551.
2. Zhang S., Wang Y., Li K. Fast TSP Algorithm based on Topological Perception. Journal of Southeast University (Natural Science Edition), 2014, 44(3), 522–525.
3. Liao T. J., Stiitzle T., de Oca M., *et a1*. A Unified Ant Colony Optimization Algorithm for Continuous Optimization. European Journal of Operational Research, 2014, 234(3), 597–609.
4. Wang L., Li M., Liu Z. H. Application of an Ant Colony Optimization based on Attractive Field in TSP. Journal of Jiangsu University, 2015, 36(5), 573–582.
5. Bullnheimer B., Hartl R. F., Strauss C. A New Rank Based Version of the Ant System – A Computational Study. Central European Journal for Operations Research and Economics, 1999, 7(1), 25–38.
6. Stutzle T., Hoos H. MAX-MIN Ant System. Future Generation Computer Systems, 2000, 16(8), 889–914.
7. Karaboga D. An Idea based on Honey Bee Swarm for Numerical Optimization. Kayseri: Erciyes University, 2005, pp. 133–151.
8. Morteza M., Hadi S. S. An Efficient ACO-based Algorithm for Scheduling Tasks onto Dynamically Reconfigurable Hardware using TSP-likened Construction Graph. Appl Intell, 2016, 45(3), 695–712.
9. Chitty D. M. Applying ACO to Large Scale TSP Instances. UK Workshop on Computational Intelligence, 2017, pp. 104–118.
10. Lin X. J., Ye D. Y. An Improved Artificial Bee Colony Algorithm with Guided Normative Knowledge. Pattern Recognition and Artificial Intelligence, 2013, 26(3), 307–314.
11. Duan Y. M., Xiao H. H. Improved Fruit Fly Algorithm for TSP Problem. Computer Engineering and Applications, 2016, 52(6), 144–149.
12. Yang J., Zheng Y., Ma L. Improved Eat Swarm Optimization for Solving Traveling Salesman Problem. Application Research of Computers, 2017, 34(12), 3607–3610.
13. Chen J. R., Chen J. H. Discrete Fishing Strategy Optimization Algorithm. Computer Science, 2017, 44(6), 141–144.
14. Ardalan Z., Karimi S., Poursabzi O., Naderi B. A Novel Imperialist Competitive Algorithm for Generalized Traveling Salesman Problems. Appl. Soft Comput. 2015, 26, 546–555.
15. Helsgaun K. Solving the Bottleneck Traveling Salesman Problem using the Lin-Kernigan-Helsgaun Algorithm. Technical Report, Computer Science, Roskilde University, 2014, 59–71.
16. Lin J. B., Chen Z. X., Yao G. X. An Improved AS Algorithm and Result Analyzing based on Domain and its Density. Engineering Journal of Wuhan University, 2016, 49(4), 627–634.
17. Li Y., Zhou Z. H., Zhao W. J. A Hierarchical Path Finding Algorithm based on Map Distribution Inform. Journal of Chinese Computer System, 2013, 34(11), 65–76.
18. Mollajafari M., Shahhoseini H. S. An Efficient ACO-based Algorithm for Scheduling Tasks onto Dynamically Reconfigurable Hardware using TSP-likened Construction Graph. Appl Intell 2016, 45, 695–712.
19. Guo Z. H., Jin L., Zheng C. Y. Study on Improved Method of Neural Network to Solve TSP. Computer Simulation, 2014, 31(4), 355–358.
20. De Santis R., Montanari R., Vignali G., *et al.* An Adapted Ant Colony Optimization Algorithm for the Minimization of the Travel Distance of Pickers in Manual Warehouses. European Journal of Operational Research, 2018, 267(1), 120–137.