Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.022807

ARTICLE

A Scheme Library-Based Ant Colony Optimization with 2-Opt Local Search for Dynamic Traveling Salesman Problem

Chuan Wang[1,*], Ruoyu Zhu[2], Yi Jiang[3], Weili Liu[4], Sang-Woon Jeon[5], Lin Sun[2] and Hua Wang[6]

[1]College of Software, Henan Normal University, Xinxiang, 453007, China

[2]College of Computer and Information Engineering, Henan Normal University, Xinxiang, 453007, China

[3]School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China

[4]School of Computer Science, Guangdong Polytechnic Normal University, Guangzhou, 510665, China

[5]Department of Military Information Engineering, Hanyang University, Ansan, 15588, South Korea

[6]College of Engineering and Science, Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne, VIC 8001, Australia

*Corresponding Author: Chuan Wang. Email: wangch@htu.edu.cn

Received: 27 March 2022 Accepted: 06 June 2022

ABSTRACT

The dynamic traveling salesman problem (DTSP) is significant in logistics distribution in real-world applications in smart cities, but it is uncertain and difficult to solve. This paper proposes a scheme library-based ant colony optimization (ACO) with a two-optimization (2-opt) strategy to solve the DTSP efficiently. The work is novel and contributes to three aspects: problem model, optimization framework, and algorithm design. Firstly, in the problem model, traditional DTSP models often consider the change of travel distance between two nodes over time, while this paper focuses on a special DTSP model in that the node locations change dynamically over time. Secondly, in the optimization framework, the ACO algorithm is carried out in an offline optimization and online application framework to efficiently reuse the historical information to help fast respond to the dynamic environment. The framework of offline optimization and online application is proposed due to the fact that the environmental change in DTSP is caused by the change of node location, and therefore the new environment is somehow similar to certain previous environments. This way, in the offline optimization, the solutions for possible environmental changes are optimized in advance, and are stored in a mode scheme library. In the online application, when an environmental change is detected, the candidate solutions stored in the mode scheme library are reused via ACO to improve search efficiency and reduce computational complexity. Thirdly, in the algorithm design, the ACO cooperates with the 2-opt strategy to enhance search efficiency. To evaluate the performance of ACO with 2-opt, we design two challenging DTSP cases with up to 200 and 1379 nodes and compare them with other ACO and genetic algorithms. The experimental results show that ACO with 2-opt can solve the DTSPs effectively.

KEYWORDS

Dynamic traveling salesman problem (DTSP); offline optimization and online application; ant colony optimization (ACO); two-optimization (2-opt) strategy

1 Introduction

The traveling salesman problem (TSP) is one of the most fundamental and intensely studied NP-complete combinatorial optimization problems [1,2]. TSP is very important in operational research and theoretical computer science. It was initially proposed for transportation, such as vehicle routing problems [3] and ship routing and scheduling problems [4]. The traditional TSP can be described as follows: given a series of nodes and the distance between each pair of nodes, the objective is to find the shortest path to visit every node once and return to the starting node. In recent years, researchers have developed many different versions of TSP, including multi-objective TSP [5,6], multi-salesman TSP [7,8], multi-solution TSP [9,10], and asymmetric TSP [11,12]. However, many real-world optimization problems are dynamic and require optimization methods that can adapt to a dynamically changing environment. Therefore, there has been increasing interest in addressing dynamic versions of the TSP, i.e., the dynamic TSP (DTSP), where the topology of nodes changes [13,14] or the weights between the nodes change [15]. Because DTSP is much closer to real-world situations, these problems have a high practical value.

In short, DTSP can be regarded as a series of different static TSPs over time. Therefore, the methods used to solve TSP can also show their efficiency in solving DTSP. Since the search space of TSP and DTSP is the full arrangement of all vertices (i.e., the nodes or the cities), a combinatorial explosion can occur with the increase in the number of vertices. Some existing studies used deterministic algorithms [16] to solve TSP, including the branch and cut method [17], linear programming method [18], and dynamic programming method [19]. However, with the increase in the scale of the problem, the deterministic optimization algorithms are less satisfying to solve the TSP due to the increase in computational complexity. Therefore, in recent studies, scholars focused on using approximate algorithms [16] or evolutionary computation algorithms [20,21] to solve large-scale TSPs. Evolutionary computation algorithms have been one of the most efficient methods to solve NP-hard optimization problems [22]. In recent decades, various types of evolutionary computation algorithms have been developed, mainly including genetic algorithm (GA) [23,24], ant colony optimization (ACO) [25,26], particle swarm optimization [27–29], and differential evolution [30–32].

Various ACO algorithms have been proposed for NP-hard optimization problems [33,34]. For instance, ACO has shown its efficiency and reliability in many optimization problems, including software project scheduling and staffing [35,36], vehicle routing problems [37,38], capacitated arc routing problems [39], virtual machine placement [40], supply chain management [41], new energy vehicle scheduling [42], and cloud workflow scheduling [43]. More importantly, ACO has shown its efficiency in solving static TSP [44,45]. In recent studies, Zhang et al. [46] proposed an improved ACO which evaluated the population according to the membership degree and updated the pheromone in turn to achieve a good balance in solving speed and quality. Skinderowicz [47] presented a novel ACO variant, namely the Focused ACO, to solve TSP, significantly improving performance. Considering the effectiveness of the ACO algorithm and its improved versions in solving static TSP, ACO is also considered to be promising in solving DTSP.

Therefore, this paper focuses on proposing an efficient scheme library-based ACO algorithm with a two-optimization (2-opt) strategy to solve a novel and special DTSP model. The novelties and contributions of this paper mainly include three aspects in terms of problem model, optimization framework, and algorithm design.

Firstly, this paper proposes a novel dynamic change fashion in the problem model to build the DTSP. In most of the existing studies on DTSP, the dynamic factors mainly include the change of weights or the change of distance between two nodes. To solve this kind of DTSP, researchers focus on developing local search operators to optimize the route after the environmental changes quickly. Unlike these DTSP models, this paper proposes to solve a novel and special DTSP model, which mainly focuses on the environmental change of node positions. In this DTSP model, the environmental change is known in advance, and the new environment can be similar to certain previous environments. The new path should be quickly optimized after the nodes' locations change.

Second, in the optimization framework, this paper proposes an idea of offline optimization and online application framework to efficiently reuse the historical information to help fast respond to the dynamic environment. In offline optimization, several candidate solutions are optimized in advance and stored in a mode scheme library, while in the online application, the candidate solutions stored in the mode scheme library are reused to solve the DTSP in a new similar environment. In DTSP, since the environmental changes can be similar, i.e., after the environmental change, the new environment can be similar to a certain previous environment, the idea of offline optimization and online application that reuses the previous solutions stored in the mode scheme library can help to solve DTSP and reduce the computational complexity effectively. In this paper, a path constructed by each ant is called a solution.

Third, this paper proposes combining ACO and the 2-opt strategy in the algorithm design to find the optimal route after an environmental change efficiently. Many existing studies show that the ACO is efficient in solving the TSP. In ACO, each ant can change the surrounding environment by releasing pheromones, perceiving the changes in the surrounding environment, and communicating indirectly through the environment, which is suitable for solving optimization problems in a dynamic environment. Moreover, this heuristic probability search method does not easily fall into local optimization and is shown to find the optimal global solution of TSP efficiently. Based on these advantages, ACO is adopted as the optimization method for solving the special DTSP. However, the uncertainty of DTSP makes it more complex than traditional TSP. It has been shown that the integration of local search operators can significantly improve the performance of ACO. Therefore, this paper embedded the 2-opt strategy into the ACO to solve this type of DTSP.

The remainder of this paper is organized as follows: Section 2 introduces some existing studies on DTSP and describes the special DTSP model. In Section 3, the overall framework and method for DTSP are elucidated. Subsequently, experiments are conducted, and the results are shown in Section 4. Finally, conclusions are drawn in Section 5.

2 DTSP

This section presents previous scholars' research on TSP and DTSP and elaborates on the proposed DTSP model.

2.1 Related Work

In recent studies on DTSP, Siemiński et al. [48] verified the usefulness of parallel and adaptive ant colony communities for solving the DTSP. Ma et al. [49] considered the dynamic optimization problem as a combination of a series of static optimization problems, and an adaptive ACO is proposed to solve the DTSP. The previous research on ACO for DTSP can be divided into the following four categories, as shown in Fig. 1: 1) the change of node position, including adding or deleting a node [50]; 2) the change in node weights, such as assigning a weight to each node. When a node task must be processed urgently, its weight will change [51]; 3) the change of the edge endpoint. For example, if there is no passage between two nodes, it can be seen that the node's endpoint has changed [52]; and 4) the change of edge weight between nodes. For example, the time of passing through an edge changes due to weather or other reasons [53].

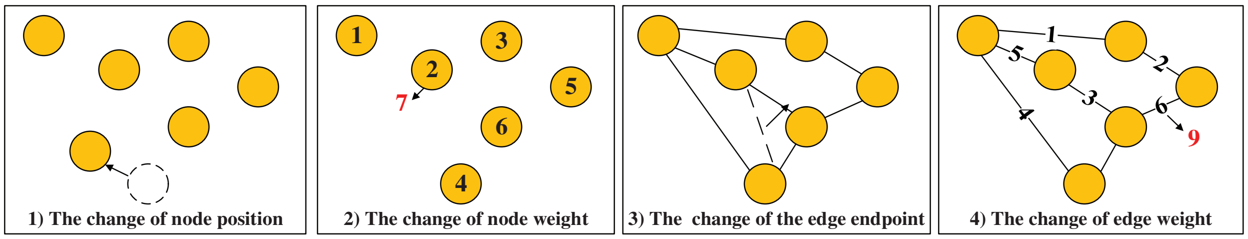

1) The change of node position
1
2
3
5
6
4
7
2) The change of node weight
3) The change of the edge endpoint
1
5
2
3
6
4
9
4) The change of edge weight

Figure 1: An example of DTSP classifications

ACO algorithm is one of the most commonly used methods to solve the TSP. Liu [52] proposed a rank-based ACO to solve the environmental changes caused by traffic jams and road closures between cities, which sometimes forced salespeople to change routes. Eyckelhof et al. [53] considered several ways of adapting the pheromone matrix locally and globally and presented a new ant system approach to dealing with DTSP. Mavrovouniotis et al. [54] considered a novel variation of DTSP where traffic jams occurred in a cyclic pattern and proposed a hybrid method that combined memory and immigrant schemes with ACO to address this type of DTSP. Mavrovouniotis et al. [55] also proposed integrating the unstringing and stringing local search operator with the max-min ant system to address symmetric and asymmetric DTSPs. Kuo et al. [51] proposed an algorithm using an improved fuzzy ant colony system to solve the problem of how to determine the completion order of tasks. Montemanni et al. [56] proposed a solution strategy based on the ACS paradigm to solve the dynamic vehicle routing problem.

In addition to the ACO, researchers also use other evolutionary algorithms and heuristic algorithms to solve the DTSP. Meng et al. [57] proposed a variable neighborhood search algorithm with direct route encoding and random initialization to solve the dynamic colored TSP, in which the weights of edges changed over time. Wang et al. [58] presented an agent-based evolutionary search algorithm that uses the principle of a collaborative endeavor learning mechanism to solve DTSP. Khouadjia et al. [59] presented an adaptive particle swarm for solving the vehicle routing problem with dynamic requests, which may significantly decrease travel distances and is adaptive to a dynamic environment. Strąk et al. [60] presented a discrete particle swarm optimization (DPSO) algorithm with heterogeneous (non-uniform) parameter values for solving the DTSP. Sabar et al. [61] proposed an effective population-based approach that combined a local search algorithm with various evolutionary operators (crossover and mutation) in an adaptive manner to address the dynamic vehicle routing problem. A heuristic method and a predictive technique were combined to solve DTSP in [62], which built a GA that feeds on Newton's motion equation to show how route optimization can be improved when targets are constantly moving.

According to the survey, previous studies on DTSP mainly focused on improving algorithms and developing local search operators to address the difficulties of real-time response to environmental changes and route optimization caused by the addition of dynamic factors. Unlike the previous studies, this paper focuses not only on the improvement of the algorithm but also on the development of the DTSP model and the optimization framework.

2.2 Problem Definition

The TSP is one of the most popular and well-researched NP-hard combinatorial optimization problems. In most real-world applications, the TSP exists in a dynamic environment rather than a static one. The TSP in a dynamic environment, called DTSP, is more challenging and uncertain. As the environment changes over time, the optimal solution of DTSP will change, and the problem's search space (called the landscape) will also change, which brings challenges. To solve DTSP efficiently, the algorithm is required to detect the change of optimization problem promptly, respond to the change quickly, and find a new optimal solution efficiently. However, it is difficult for classic evolutionary algorithms to satisfy these conditions, which leads to challenges in solving DTSP.

This paper designs a new DTSP model, where the environmental change is the change of the node position, and the environment is regularly changed. This DTSP model considers two conditions in real-world applications. First, the position of a node will change due to road or traffic problems. This change is temporal, and only one node will change in each period (a period corresponds to an environmental change). Second, since traffic jams are usually regular, the environmental changes can also be regular. Therefore, some new environments can be similar to certain previous environments.

Mathematically, the model of DTSP in this paper can be formulated as follows: Given a list of nodes, a salesman must traverse each node once and only once to build a Hamilton path. First, suppose the road map is an undirected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \{0, 1, \ldots, N-1\}$ is the set of nodes (denoted by the node indices), and N is the number of nodes. The node with serial number 0 indicates the beginning and ending node of the traveling salesman; other nodes in set $\boldsymbol{V}$ indicate the nodes that the salesman must visit. $\boldsymbol{E} = \{e_{ij} \mid i, j \in \boldsymbol{V}, i \neq j\}$ is the edge set, indicating the road/edge between the nodes i and j. Each edge e_{ij} has a weight value, denoted as d_{ij}, which measures the distance between the two nodes i and j. In the solution of the DTSP, the decision variables x_{ij} and y_i are defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{If edge } e_{ij} \text{ is visited} \\ 0, & \text{Otherwise} \end{cases} \tag{1}$$

$$y_i = \begin{cases} 1, & \text{If the node } i \text{ is visited} \\ 0, & \text{Otherwise} \end{cases} \tag{2}$$

The objective of DTSP is to find the shortest Hamilton path when the location of a node changes over time. The Hamilton path can be formulated as permutation Φ (denoted by the node indices). Then, DTSP finds the shortest length in all feasible permutations. More precisely, this paper formulates it as follows:

$$\min f(\Phi) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} d_{ij} x_{ij} \tag{3}$$

The constraints for DTSP are formulated as follows:

Constraint I: All nodes are strictly visited only once.

$$y_i = 1, i = 0, 1, \ldots, N-1 \tag{4}$$

Constraint II: Each edge of the solution has only one starting node connected to it.

$$\sum_{i=0}^{N-1} x_{ij} = 1 \tag{5}$$

where $i = 0, 1, \ldots, N-1$.

Constraint III: Each edge of the solution has only one destination node connected to it.

$$\sum_{j=0}^{N-1} x_{ij} = 1 \tag{6}$$

where $i = 0, 1, \ldots, N-1$.

Constraint IV: Solutions constituting incomplete routes are eliminated.

$$X = (x_{ij}) \in S \tag{7}$$

where S is the branch elimination constraint. That is, the incomplete route is eliminated.

In this DTSP model, the environmental change is the change of node position. When the location of a node changes in a new environment, the path must be re-optimized. Note that, in this DTSP model, the starting node is fixed, and its position remains unchanged. When a new environment starts, the position of a node will change. The location change rule is as follows:

$$p' = p_0 + \Delta p \tag{8}$$

where p'(x', y') is the position updated after a change, $p_0(x_0, y_0)$ is the position of the original location of the changing node, and Δp is the amplitude of position change, which is set to a random integer change (add or subtract) in the range of [5, 10].

The objective of DTSP is to find the optimal Hamilton path of the salesman to minimize the total distance after the environmental change. This type of DTSP widely exists in daily life. For example, in logistics distribution, the target location may change in some real-world scenarios. After the position of a node is changed, the optimal solution in the previous environment may not be the optimal solution in the current environment. Thus, the algorithm is required to quickly find an optimal path after the change of node position. Therefore, research on this type of problem is of great practical significance. The problem will become more complex by changing some variables, such as increasing the number of salesmen or the number of nodes with changed locations.

3 Scheme Library-Based ACO with 2-Opt Algorithm

3.1 Scheme Library-Based Offline Optimization and Online Application Strategy

Different from existing studies, in the proposed DTSP model, the environmental changes can be regular. That is, the new environment can be similar to some previous environments. To enhance the performance according to this property, this paper proposes an interesting and effective idea of offline optimization and online application strategy, which stores the optimized solutions in the scheme library and reuses the solutions after the environmental change. When users encounter similar environmental changes, they can directly call the corresponding scheme stored in the scheme library for online application. If a change in the scheme library requires no optimization scheme, the proposed ACO with 2-opt algorithm is called for optimization and placing the optimized solution into the scheme library. Herein, this paper chooses ACO as the optimization algorithm. Many researchers use ACO because of its flexibility and strong adaptability in solving TSP and DTSP. Furthermore, this paper integrates the 2-opt strategy into ACO, dramatically improving local search ability [63,64].

The offline optimization and online application strategy flowchart is shown in Fig. 2. The left frame shows the process of offline optimization, while the right frame shows the online application. First, a mode scheme library is created to store the already optimized solutions (i.e., mode schemes) in the previous environments. Note that the size of the mode scheme library is unlimited, so all the solutions corresponding to all modes are stored in the mode scheme library. Each mode represents an environment under the influence of different dynamic factors. Second, once the environmental change is detected, the online application process is carried out. That is, after the environmental change is detected if the new environment is similar to a certain previous environment, the previous solution (i.e., mode) that corresponds to a similar environment is directly used in this new environment. If the solution that corresponds to the new environment is found, the process of utilizing mode scheme library is finished, i.e., finish the mode. Otherwise, the offline optimization strategy is carried out, where the mode optimization algorithm (i.e., ACO with 2-opt) is adopted to find the optimal solution for the current environment, and the optimal solution is added to the mode scheme library. The above operations are repeated until the termination conditions are met.

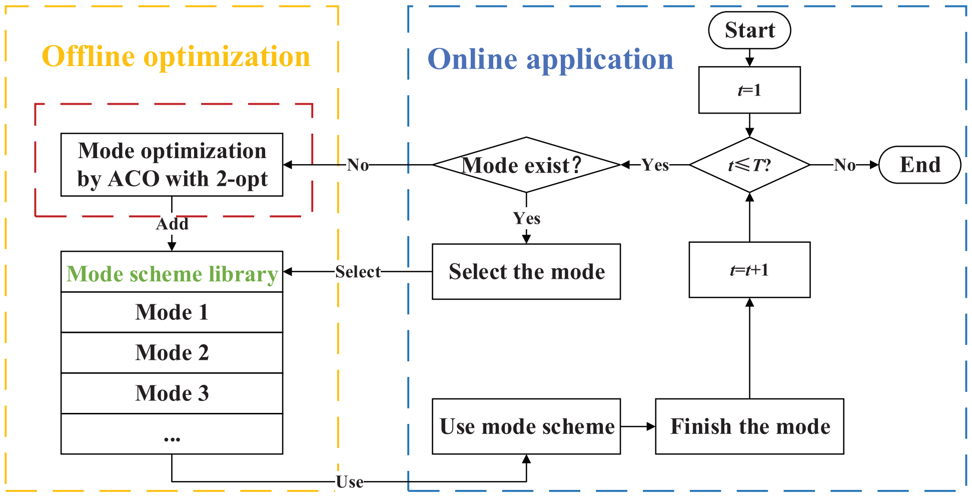


Figure 2: The overall framework for offline optimization and online application strategy

3.2 ACO

ACO [65] is a stochastic search algorithm to simulate the foraging process of real ants in nature. ACO is a swarm intelligence algorithm with the advantages of robustness, global search, parallel distribution calculation, and easy combination with other methods. In ACO, the principle of finding the shortest path to the food source is described as follows: each ant releases pheromones on the way to the food source. Over time, the pheromones on the longer path will not accumulate because of the long passing of time, while those on the shorter path will easily accumulate because of the short passing of time. Therefore, more pheromones can be left per unit of time on the shorter path. Other ants choose the path with the strongest pheromone by judging the concentration of "pheromone", which causes them to focus more on the shorter path. Through this continuous positive feedback, the entire ant colony will eventually choose the shortest path to find food. Due to its good results, ACO is widely used to solve all combinatorial optimization problems, especially TSP [66,67]. This paper adopts the elitist ant system (EAS) to solve DTSP. The two main steps in the process are path construction and pheromone update.

3.2.1 Path Construction

In the beginning, each ant selects a node as its starting node (In this paper, the starting node is set as node 0) and maintains a visited nodes sequence to store the visited nodes. In each step of path construction, the ant selects the next node to visit. Specifically, if the current node of ant k is i, the ant chooses the next node j in its path from the remaining nodes based on the roulette wheel selection, with the selection probability of each node computed as follows:

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^\alpha[\eta(i,j)]^\beta}{\sum\limits_{u\in J_k(i)}[\tau(i,u)]^\alpha[\eta(i,u)]^\beta}, & \text{If } j \in J_k(i) \\ 0, & \text{Otherwise} \end{cases} \tag{9}$$

where $J_k(i)$ represents the set of nodes that can be reached directly from node i, and in order to avoid selecting repetitive nodes, a tabu table is set up in the algorithm to store the nodes visited by ants. Therefore, the set of nodes $J_k(i)$ also should not be in the sequence of the visited nodes. $\eta(i,j)$ is heuristic information calculated by $\eta(i,j) = 1/d_{ij}$. $\tau(i,j)$ represents the amount of pheromone on edge e_{ij}. α and ; are two user-defined parameters. α is the parameter of pheromone importance, β is the expectation factor.

3.2.2 *Pheromone Updating*

In ACO, the pheromone is updated after all ants complete the travel operation. As shown in Eq. (9), pheromones between nodes are the most important factor affecting the probability of p_k. The pheromone update includes three parts. First, after each ant finishes its travel, the pheromones on all paths evaporate, which prevents the pheromone from accumulating with the increase in iteration times. Specifically, this paper multiplies the pheromones on all edges by a constant less than 1 (i.e., $1-\rho$). Second, all ants release pheromones to the edge of their paths according to the path length. Thirdly, the EAS strengthens the pheromone update on the optimal path. That is, after the pheromone update, the ant with the best path R_b adds additional pheromones to each edge of the path R_b. These additional pheromones can make the ants search faster and converge more correctly. Based on these three parts, the update of the pheromone quantity $\tau(i,j)$ on edge between nodes i and j is shown as the following formula:

$$\begin{aligned} &\tau(i,j) = (1-\rho)\cdot\tau(i,j) + \textstyle\sum_{k=1}^{m}\Delta\tau_k(i,j) + w\cdot\Delta\tau_b(i,j) \\ &\Delta\tau_k(i,j) = \begin{cases} (L_k)^{-1}, & \text{If } e_{ij} \in R^k \\ 0, & \text{Otherwise} \end{cases} \\ &\Delta\tau_b(i,j) = \begin{cases} (L_b)^{-1}, & \text{If } e_{ij} \in R_b \\ 0, & \text{Otherwise} \end{cases} \end{aligned} \tag{10}$$

where m is the number of ants. The greater the number of ants is, the more accurate the optimal solution will be, but many repeated solutions will be generated. As the algorithm approaches the optimal value, the positive feedback of information will be reduced, and a considerable amount of repeated work will consume resources and increase the time complexity. ρ is the evaporation rate of pheromone. When ρ is too small, there are too many pheromones left on each path, resulting in an invalid path continuing to be searched, affecting the algorithm's convergence speed. When ρ is too large, although an invalid path can be excluded from the search, it cannot guarantee that the effective path will also be abandoned, affecting the search for the optimal value. $\Delta\tau_k(i,j)$ is the amount of pheromone released by the kth ant on the edge it passes, which is equal to the reciprocal path length of ant k in this round. L_k represents the path length, which is the sum of the lengths of all edges in R^k. Parameter w is the edge weight value, and L_b is the optimal path length since the algorithm's beginning.

3.3 Two-Optimization (2-opt)

The 2-opt operation, first proposed by Croes et al. [68], is a widespread and straightforward local search method that has been widely used to cooperate with evolutionary algorithms to improve solution quality [69]. Herein, this paper combines the 2-opt operation with ACO to enhance efficiency.

In each generation, the 2-opt strategy is applied to the edges in each solution (i.e., each ant) to generate new solutions by reserving the directions of several consecutive edges. First, the 2-opt strategy randomly selects two different edges and deletes these two edges. After the deleting operation, two edge sequences are obtained. Then, the edge sequence between these two edges is reversed, and this reversed edge sequence is re-connected to the other edge sequence to obtain a new solution. Finally, the better one (i.e., the solution with a shorter path) between the new solution and the original solution is reserved. Specifically, the route in Fig. 3 serves as an example. The original solution is "1–2–3–7–6–5–4–8–1". Suppose edges 3–7 and 4–8 are the selected edges in 2–opt. In the original solution, the edge sequence between edge 3–7 and edge 4–8 is "7–6–5–4", and the other edge sequence is "8–1–2–3". After the reverse operation in the 2–opt strategy, the edge sequence between edge 3–7 and edge 4–8 is changed to "4–5–6–7". Re-connecting edge sequences "8–1–2–3" and "4–5–6–7", a new solution "1–2–3–4–5–6–7–8–1" is generated. Obviously, the path length of the new solution is shorter than that of the original solution. Thus the original solution is replaced by the new solution.

Algorithm 1: The pseudocode of offline optimization by ACO with 2-opt

```
Input: Maximum number of iterations $T$, Number of nodes $N$.
    Begin
    1: For each edge Do
    2:      Initialize pheromone $\tau_0$;
    3: End For;
    4: count = 0;
    5: While (count $<$ $T$) Do
    6:      For each ant $k$ Do
    7:          Select the node 0 as the starting node;
    8:          For $i = 1$ to $N - 1$ Do
    9:              Select the next node $j$ with the probability given by (9);
    10:         End For
    11:         Apply 2-opt on the solution of ant $k$;
    12:         Get the Hamilton path constructed by ant $k$;
    13:         Calculate the path length $L_k$;
    14:     End For
    15:     For each edge Do
    16:         Update the pheromone value by (10);
    17:     End For
    18:     count = count + 1;
    19: End While
    20: Get the best solution $\boldsymbol{S}_{best}$;
    End
Output: $S_{best}$
```

Based on the above description of ACO and 2-opt strategy, the pseudocode of the proposed ACO with 2-opt algorithm is given in Algorithm 1. At the beginning of the algorithm, initialize the ants and the pheromone on each edge. The starting node (i.e., the first node of each path) of each ant is set as node 0. Then, the roulette selection method selects the next visiting node for each ant according to the probability calculated by Eq. (9). After that, to help the algorithm efficiently find the optimal solution, the 2-opt strategy is performed on each ant to improve the quality of the constructed path. The fitness evaluation is carried out in line 13, which calculates the path length constructed by each ant. Then, for each edge, the pheromone quantity is updated according to Eq. (10). Finally, if the termination condition of the algorithm is met (i.e., reach the maximum number of iterations 5000), the algorithm is finished, and the best result is retained.

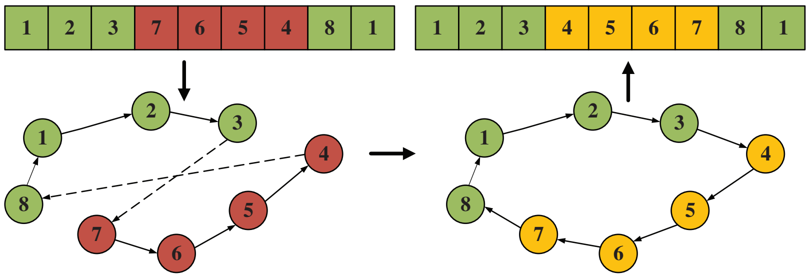


Figure 3: An example of the 2-opt operation

4 Experimental Studies

4.1 Experimental Setting

To evaluate the performance of ACO with 2-opt in the special DTSP, this paper set the number of modes (i.e., number of environmental changes) to eight. The DTSP benchmark instance kroA200 is adopted as the benchmark DTSP from the classic TSP library (TSPLIB) [70]. In this benchmark, there are 200 cities (i.e., nodes). And set the starting node as the node with serial number 0, and the eight modes correspond to the location changes of eight nodes. This paper assumes that the indexes of these changed nodes are 100, 140, 120, 160, 40, 80, 20, and 60. Besides, to further verify the robustness of the ACO with 2-opt, this paper also tests it on a large-scale data set nrw1379 [70]. In this benchmark, there are 1379 cities (i.e., nodes). The parameters of the ACO algorithm are set to typical values, as shown in Table 1. The experimental results are obtained from ten independent execution of each algorithm and select one set of results to draw roadmaps. Finally, it deserves to be noted that all algorithms are implemented in the C++ language and executed on a PC with Intel(R) Core (TM) i5-10210U 1.60 GHz CPUs, 16 GB memory, and a 64-bit Ubuntu 12.04 LTS system.

Table 1: Parameter settings

Parameter	Description	Value
m	Number of ants	30
α	Pheromone weight	1
β	Heuristic information weight	5
ρ	Pheromone evaporation rate	0.1
τ_0	Initial pheromone quantity	1
w	Edge weight of $\Delta_b(i,j)$	200 in kroA200 1379 in nrw1379

4.2 Experimental Results

In the experiments, this paper defines eight types of environmental changes and denotes the eight periods corresponding to eight different environments as t_1–t_8. In each period, the position of a pre-defined node is changed. Specifically, the eight indexes of the changed node corresponding to the eight periods t_1–t_8 are 100, 140, 120, 160, 40, 80, 20, and 60 in the instance kroA200. For example, when the period reaches t_5, the location of node 40 will change. Besides, the eight indexes of the changed node corresponding to the eight periods t_1–t_8 are 900, 150, 450, 300, 1050, 600, 750, and 1200 in the instance nrw1379. These types of environmental changes cyclically occur until the algorithm meets the termination condition.

Firstly, to show the effectiveness and efficiency of the ACO with 2-opt in solving the DTSP proposed in this paper, this paper conduct the comparison between the proposed ACO with 2-opt, ACO, and GA on the DTSP model. The results with respect to the best value and the mean value on kroA200 and nrw1379 are shown in Tables 2 and 3, respectively. In the table, Change-Index represents the index of the node whose position is changed in the corresponding environment. The best value represents the minimum total path length in the 10 independent executions, and the mean value represents the average path length over the 10 executions. In addition, in this table, the best results between the algorithms with respect to the best value are highlighted in boldface, while the best results in terms of the mean value are shaded. As we can see, considering the best value and mean value, the performance of the ACO with 2-opt is superior to that of the compared algorithms in solving DTSP. In most cases with different indexes of changed nodes, the proposed ACO with 2-opt shows the best performance than the compared algorithms. Therefore, our proposed ACO with 2-opt effectively and efficiently solves DTSP.

Table 2: Comparison results between GA, ACO, and ACO with 2-opt on kroA200

Envir.	Change-index	GA		ACO		ACO with 2-opt	
		Best	Mean	Best	Mean	Best	Mean
t_1	100	34808.78	38037.03	30918.18	31738.51	**29717.62**	30179.38
t_2	140	35578.34	39702.48	30732.48	31812.35	**29595.61**	30168.90
t_3	120	34129.78	38267.25	30713.68	31207.09	**29590.49**	30055.31
t_4	160	35845.55	39683.07	30567.65	31481.35	**29642.39**	30275.27
t_5	40	35958.55	37671.29	30210.45	31418.82	**29672.13**	30359.95
t_6	80	35134.52	37700.04	30450.60	31284.21	**29575.66**	30132.09
t_7	20	36694.34	40444.58	30108.26	31205.17	**29614.43**	30118.28
t_8	60	35274.57	38741.69	30888.61	31561.85	**29721.47**	30339.05

Table 3: Comparison results between GA, ACO, and ACO with 2-opt on nrw1379

Envir.	Change-index	GA		ACO		ACO with 2-opt	
		Best	Mean	Best	Mean	Best	Mean
t_1	900	315134.52	318019.91	76678.01	78461.06	**75641.32**	77166.05
t_2	150	310808.78	312326.39	76937.54	79562.62	**75576.78**	77304.37
t_3	450	306226.08	308267.25	76516.83	78956.03	**74163.23**	77207.03
t_4	300	305003.86	311108.47	**75410.39**	78818.21	75562.46	77183.09
t_5	1050	301207.02	311016.88	76531.58	78942.07	**75212.04**	77245.65
t_6	600	305845.55	313601.06	**75499.88**	78327.40	76137.45	77588.45
t_7	750	300334.99	307671.29	**74914.56**	78346.01	75107.18	77575.74
t_8	1200	305274.57	315016.15	76465.90	78523.15	**74701.50**	76748.80

Secondly, to show the effect of the local search operation 2-opt, this paper conducts a comparison between the proposed ACO with 2-opt and the ACO without 2-opt, and the results of the comparison are also shown in Tables 2 and 3. From Tables 2 and 3, the following conclusions can be drawn: 1) Concerning the results with respect to the best value in the 10 independent runs, the ACO with 2-opt achieves the best performance on all the 8 modes in kroA200 and 5 modes in nrw1379, while the ACO without 2-opt achieves the best performance on no modes in kroA200 and 3 modes in nrw1379. 2) From the mean value over the 10 independent runs, ACO with 2-opt achieves the best results on all the 8 modes in both kroA200 and nrw1379, while ACO without 2-opt does not obtain the best results on any modes. 3) With the different indexes of changed nodes, regardless of the perspective of the best value or the perspective of the mean value, the superiority of ACO with 2-opt over ACO without 2-opt becomes increasingly notable. Therefore, the 2-opt is necessary for ACO with 2-opt algorithm.

The route optimization diagrams of the eight modes on kroA200 and nrw1379 are shown in Figs. 4 and 5, respectively. The pentagram red dot indicates the starting node, which is labeled node 0. The round green dot indicates the node whose location has changed. (a) shows the original distribution of nodes in the kroA200 and nrw1379 test sets. The figures from (b) to (i) provide 8 modes in the scheme library corresponding to the change in node location in each period, which is the result of 5000 iterations. Figs. 4 and 5 show the roadmap for online applications.

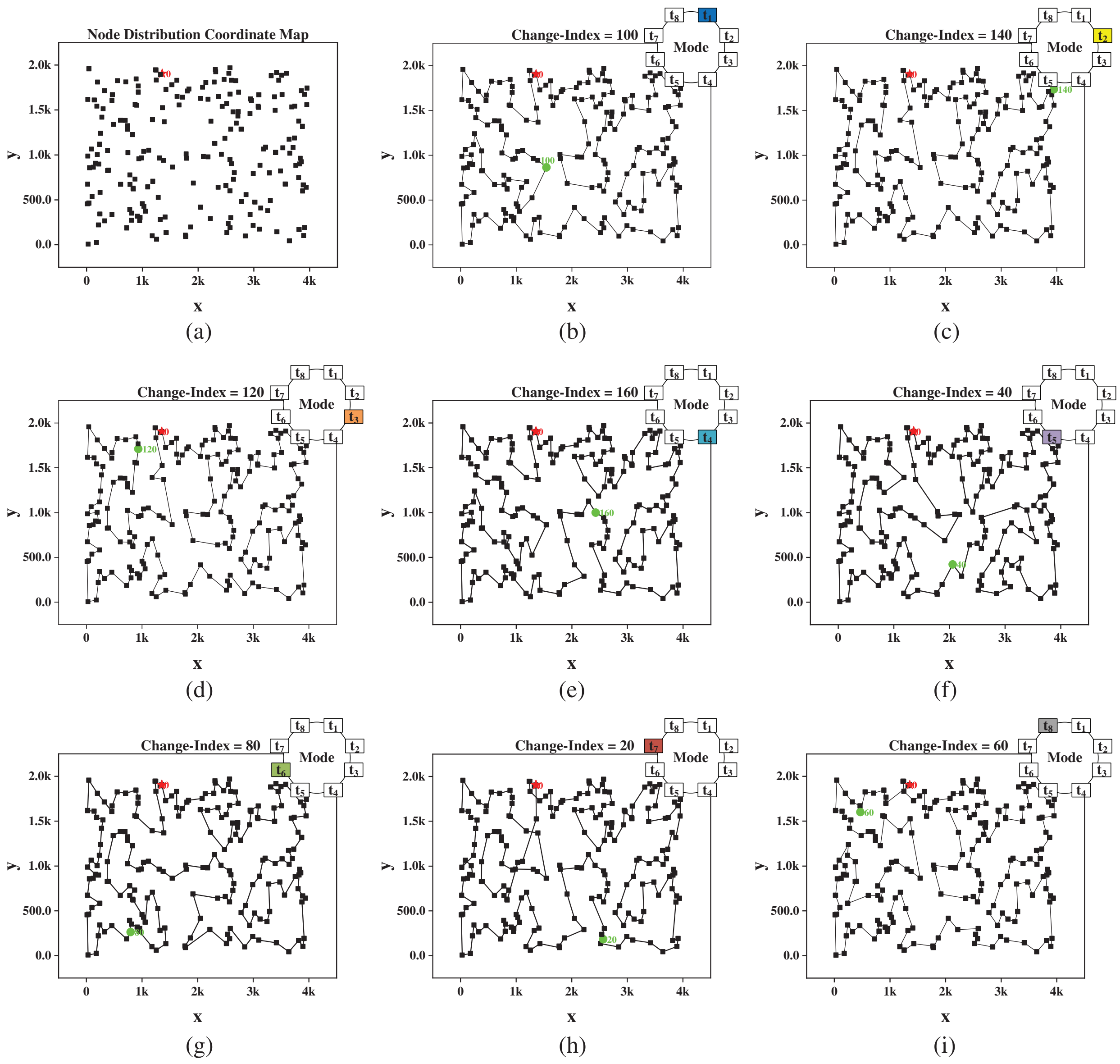
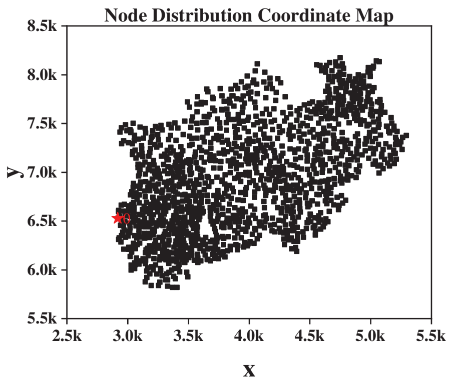
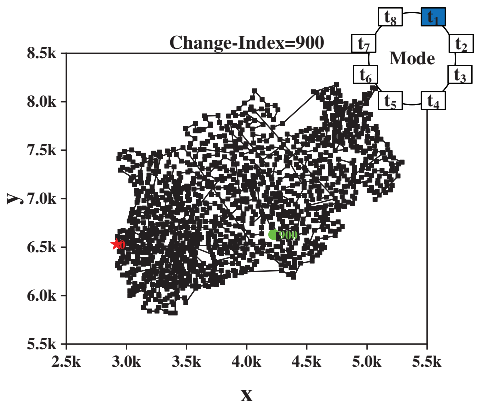


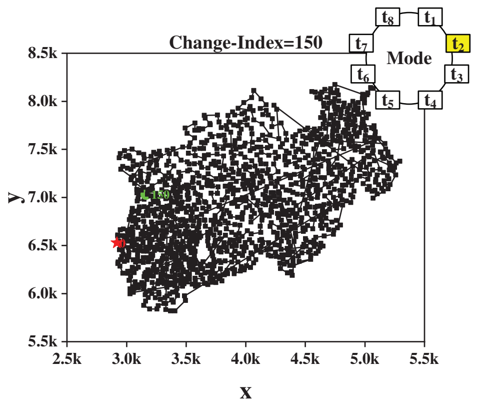
Figure 4: Node distribution position map and route scheme maps of instance kroA200. (a) Node distribution position map. (b) Route scheme map in environment t_1. (c) t_2. (d) t_3. (e) t_4. (f) t_5. (g) t_6. (h) t_7. (i) t_8



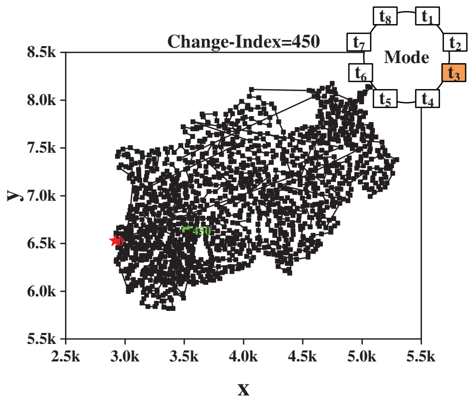
(a)



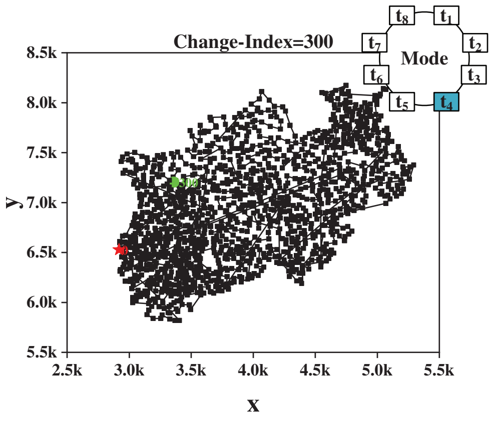
(b)



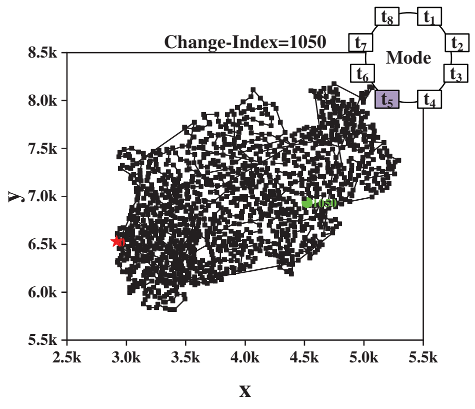
(c)



(d)



(e)



(f)

Figure 5: (Continued)

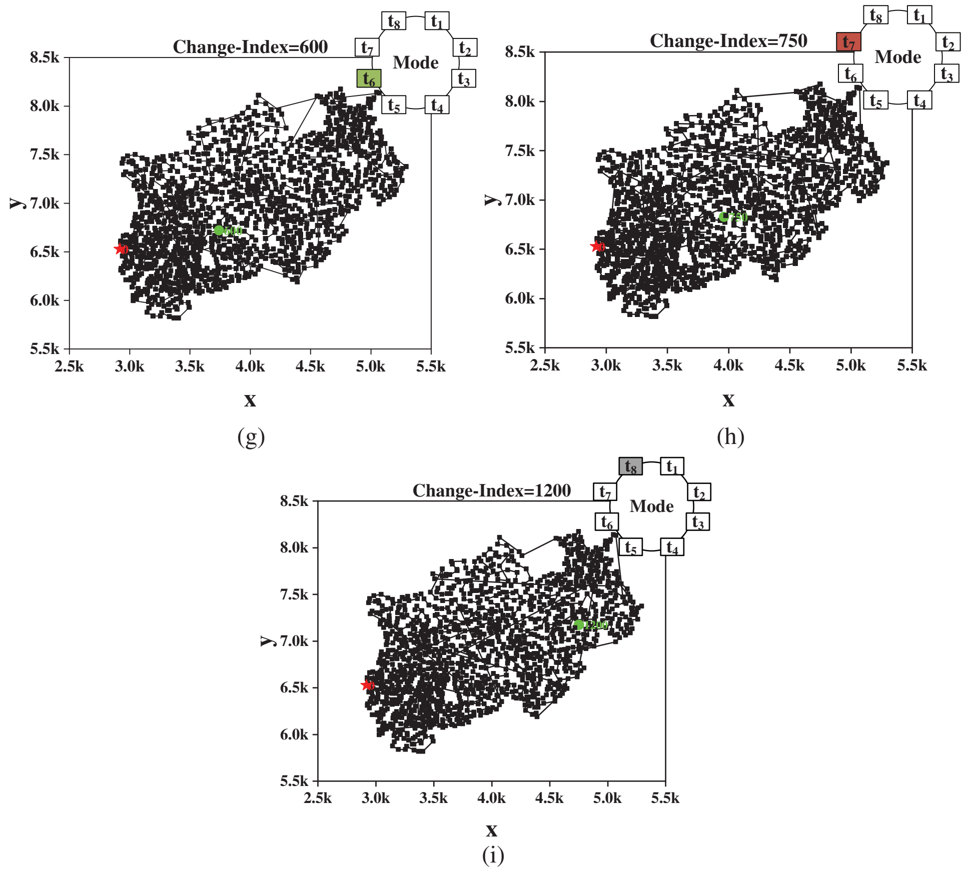


Figure 5: Node distribution position map and route scheme maps of instance nrw1379. (a) Node distribution position map. (b) Route scheme map in environment t_1. (c) t_2. (d) t_3. (e) t_4. (f) t_5. (g) t_6. (h) t_7. (i) t_8

5 Conclusion

In this paper, a new DTSP model is proposed and solved, focusing on the change in node positions over time. In this type of DTSP, a node will regularly change its location due to traffic or other factors during a certain period. This change occurs randomly. Second, this paper proposes an idea of offline optimization and online application. Several solutions are optimized in advance and stored in the scheme library in offline optimization. In the online application, when the environment has changed, and some roads are not feasible, this paper uses the solutions in the scheme library to provide optimal routes to the new similar environment. Third, to effectively solve DTSP via the development of the algorithm, this paper proposes an ACO with a 2-opt strategy. Because ACO is shown to be one of the most effective algorithms in solving TSP, ACO is adopted as the basic optimization method in the special DTSP model and carried out experiments to show its effectiveness of ACO. Our goal is that when the node positions change during this period, the algorithm can provide a new route in advance and minimize the total path length. The experimental results on the DTSP show that this strategy greatly improves the efficiency of the ACO.

Based on this paper, additional versions of DTSP can be extended for future work, and the solution will be more complex. Determining methods to balance the algorithm's effectiveness and efficiency is a worthy research topic.

Funding Statement: This work was supported in part by the National Research Foundation of Korea (NRF-2021H1D3A2A01082705).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Yao, K., Sun, W., Cui, Y., He, L., Shao, Y. (2021). A genetic algorithm with projection operator for the traveling salesman problem. *2021 IEEE International Conference on Artificial Intelligence and Industrial Design (AIID)*, pp. 194–197. Guangzhou, China. DOI 10.1109/AIID51893.2021.9456487.
2. Xu, X., Li, J., Zhou, M. (2020). Delaunay-triangulation-based variable neighborhood search to solve large-scale general colored traveling salesman problems. *IEEE Transactions on Intelligent Transportation Systems, 22(3),* 1583–1593. DOI 10.1109/TITS.2020.2972389.
3. Dorling, K., Heinrichs, J., Messier, G. G., Magierowski, S. (2016). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(1),* 70–85. DOI 10.1109/TSMC.2016.2582745.
4. Arnesen, M. J., Gjestvang, M., Wang, X., Fagerholt, K., Thun, K. et al. (2017). A traveling salesman problem with pickups and deliveries, time windows and draft limits: Case study from chemical shipping. *Computers & Operations Research, 77,* 20–31. DOI 10.1016/j.cor.2016.07.017.
5. Sampaio, S. M., Dantas, A., Camilo-Junior, C. G. (2019). Routing sales territory by solving a multi-objective TSP variant with evolutionary algorithms. *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 109–116. Portland, OR, USA. DOI 10.1109/ICTAI.2019.00024.
6. Lust, T., Teghem, J. (2010). The multiobjective traveling salesman problem: A survey and a new approach. In: *Advances in multi-objective nature inspired computing*, pp. 119–141. Berlin, Heidelberg, Springer. DOI 10.1007/978-3-642-11218-8_6.
7. Qu, H., Yi, Z., Tang, H. (2007). A columnar competitive model for solving multi-traveling salesman problem. *Chaos, Solitons & Fractals, 31(4),* 1009–1019. DOI 10.1016/j.chaos.2005.10.059.
8. Khan, A., Yanmaz, E., Rinner, B. (2015). Information exchange and decision making in micro aerial vehicle networks for cooperative search. *IEEE Transactions on Control of Network Systems, 2(4),* 335–347. DOI 10.1109/TCNS.2015.2426771.
9. Huang, T., Gong, Y., Kwong, S., Wang, H., Zhang, J. (2019). A niching memetic algorithm for multi-solution traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 24(3),* 508–522. DOI 10.1109/TEVC.2019.2936440.
10. Huang, T., Gong, Y., Zhang, J. (2018). Seeking multiple solutions of combinatorial optimization problems: A proof of principle study. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1212–1218. Bangalore, India. DOI 10.1109/SSCI.2018.8628856.
11. Stutzle, T., Hoos, H. (1997). MAX-MIN ant system and local search for the traveling salesman problem. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pp. 309–314. Indianapolis, IN, USA. DOI 10.1109/ICEC.1997.592327.
12. Kanellakis, P. C., Papadimitriou, C. H. (1980). Local search for the asymmetric traveling salesman problem. *Operations Research, 28(5),* 1086–1099. DOI 10.1287/opre.28.5.1086.

13. Guntsch, M., Middendorf, M. (2001). Pheromone modification strategies for ant algorithms applied to dynamic TSP. *Workshops on Applications of Evolutionary Computation*, pp. 213–222. Berlin, Heidelberg, Springer. DOI 10.1007/3-540-45365-2_22.
14. Kang, L., Zhou, A., McKay, B., Li, Y., Kang, Z. (2004). Benchmarking algorithms for dynamic travelling salesman problems. *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, pp. 1286–1292. Portland, OR, USA. DOI 10.1109/CEC.2004.1331045.
15. Mavrovouniotis, M., Yang, S. (2013). Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *Applied Soft Computing, 13(10),* 4023–4037. DOI 10.1016/j.asoc.2013.05.022.
16. Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59(2),* 231–247. DOI 10.1016/0377-2217(92)90138-Y.
17. Cordeau, J. F., Iori, M., Laporte, G., Salazar González, J. J. (2010). A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. *Networks, 55(1),* 46–59. DOI 10.1002/net.20312.
18. Palacio, J. D., Rivera, J. C. (2019). Mixed-integer linear programming models for one-commoditypickup and delivery traveling salesman problems. *Workshop on Engineering Applications*, pp. 735–751. Cham, Springer. DOI 10.1007/978-3-030-31019-6_62.
19. Thanh, P. D., Binh, H. T. T., Lam, B. T. (2013). A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem. *2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR)*, pp. 66–71. Hanoi, Vietnam. DOI 10.1109/SOCPAR.2013.7054102.
20. Lin, S., Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research, 21(2),* 498–516. DOI 10.1287/opre.21.2.498.
21. Wei, F., Chen, W., Hu, X., Zhang, J. (2019). An empirical study on evolutionary algorithms for traveling salesman problem. *2019 9th International Conference on Information Science and Technology (ICIST)*, pp. 273–280. Hulunbuir, China. DOI 10.1109/ICIST.2019.8836906.
22. Zhan, Z. H., Shi, L., Tan, K. C., Zhang, J. (2022). A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review, 55(1),* 59–110. DOI 10.1007/s10462-021-10042-y.
23. Wu, S., Zhan, Z. H., Zhang, J. (2021). SAFE: Scale-adaptive fitness evaluation method for expensive optimization problems. *IEEE Transactions on Evolutionary Computation, 25(3),* 478–491. DOI 10.1109/TEVC.2021.3051608.
24. Li, J., Zhan, Z. H., Wang, C., Jin, H., Zhang, J. (2020). Boosting data-driven evolutionary algorithm with localized data generation. *IEEE Transactions on Evolutionary Computation, 24(5),* 923–937. DOI 10.1109/TEVC.2020.2979740.
25. López-Ibáñez, M., Blum, C. (2010). Beam-ACO for the travelling salesman problem with time windows. *Computers & Operations Research, 37(9),* 1570–1583. DOI 10.1016/j.cor.2009.11.015.
26. Yu, W., Hu, X., Zhang, J., Huang, R. (2009). Self-adaptive ant colony system for the traveling salesman problem. *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1399–1404. San Antonio, TX, USA. DOI 10.1109/ICSMC.2009.5346279.
27. Wang, Z., Zhan, Z. H., Kwong, S., Jin, H., Zhang, J. (2020). Adaptive granularity learning distributed particle swarm optimization for large-scale optimization. *IEEE Transactions on Cybernetics, 51(3),* 1175–1188. DOI 10.1109/TCYB.2020.2977956.
28. Jian, J., Chen, Z., Zhan, Z. H., Zhang, J. (2021). Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization. *IEEE Transactions on Evolutionary Computation, 25(4),* 779–793. DOI 10.1109/TEVC.2021.3065659.
29. Hou, G., Xu, Z., Liu, X., Jin, C. (2019). Improved particle swarm optimization for selection of shield tunneling parameter values. *Computer Modeling in Engineering & Sciences, 118(2),* 317–337. DOI 10.31614/cmes.2019.04693.

30. Li, J., Zhan, Z. H., Tan, K. C., Zhang, J. (2021). A meta-knowledge transfer-based differential evolution for multitask optimization. *IEEE Transactions on Evolutionary Computation*. DOI 10.1109/TEVC.2021.3131236.
31. Zhan, Z. H., Wang, Z., Jin, H., Zhang, J. (2019). Adaptive distributed differential evolution. *IEEE Transactions on Cybernetics, 50(11),* 4633–4647. DOI 10.1109/TCYB.2019.2944873.
32. Jiang, Y., Zhan, Z. H., Tan, K. C., Zhang, J. (2021). Optimizing niche center for multimodal optimization problems. *IEEE Transactions on Cybernetics*. DOI 10.1109/TCYB.2021.3125362.
33. Liu, Y., Gao, C., Zhang, Z., Lu, Y., Chen, S. et al. (2015). Solving NP-hard problems with physarum-based ant colony system. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14(1),* 108–120. DOI 10.1109/TCBB.2015.2462349.
34. Ke, L., Zhang, Q., Battiti, R. (2013). MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE Transactions on Cybernetics, 43(6),* 1845–1859. DOI 10.1109/TSMCB.2012.2231860.
35. Wang, Q., Liu, W., Yu, H., Zheng, S., Gao, S. et al. (2019). CPAC: Energy-efficient algorithm for IoT sensor networks based on enhanced hybrid intelligent swarm. *Computer Modeling in Engineering & Sciences, 121(1),* 83–103. DOI 10.32604/cmes.2019.06897.
36. Chen, W., Zhang, J. (2012). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering, 39(1),* 1–17. DOI 10.1109/TSE.2012.17.
37. Wang, X., Choi, T. M., Liu, H., Yue, X. (2016). Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems. *IEEE Transactions on Intelligent Transportation Systems, 17(11),* 3132–3141. DOI 10.1109/TITS.2016.2542264.
38. Ma, Y., Gong, Y., Xiao, C., Gao, Y., Zhang, J. (2018). Path planning for autonomous underwater vehicles: An ant colony algorithm incorporating alarm pheromone. *IEEE Transactions on Vehicular Technology, 68(1),* 141–154. DOI 10.1109/TVT.2018.2882130.
39. Liu, H., Gao, L., Pan, Q. (2011). A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. *Expert Systems with Applications, 38(4),* 4348–4360. DOI 10.1016/j.eswa.2010.09.104.
40. Liu, X., Zhan, Z. H., Deng, J. D., Li, Y., Gu, T. et al. (2016). An energy efficient ant colony system for virtual machine placement in cloud computing. *IEEE Transactions on Evolutionary Computation, 22(1),* 113–128. DOI 10.1109/TEVC.2016.2623803.
41. Zhang, X., Zhan, Z. H., Fang, W., Qian, P., Zhang, J. (2021). Multi population ant colony system with knowledge based local searches for multiobjective supply chain configuration. *IEEE Transactions on Evolutionary Computation, 26(3)*, 512–526. DOI 10.1109/TEVC.2021.3097339.
42. Shi, L., Zhan, Z. H., Liang, D., Zhang, J. (2022). Memory-based ant colony system approach for multi-source data associated dynamic electric vehicle dispatch optimization. *IEEE Transactions on Intelligent Transportation Systems*. DOI 10.1109/TITS.2022.3150471.
43. Chen, Z., Zhan, Z. H., Lin, Y., Gong, Y., Gu, T. et al. (2018). Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach. *IEEE Transactions on Cybernetics, 49(8),* 2912–2926. DOI 10.1109/TCYB.2018.2832640.
44. Gan, R., Guo, Q., Chang, H., Yi, Y. (2010). Improved ant colony optimization algorithm for the traveling salesman problems. *Journal of Systems Engineering and Electronics, 21(2),* 329–333. DOI 10.3969/j.issn.1004-4132.2010.02.025.
45. Hlaing, Z. C. S. S., Khine, M. A. (2011). Solving traveling salesman problem by using improved ant colony optimization algorithm. *International Journal of Information and Education Technology, 1(5),* 404–409. DOI 10.7763/IJIET.2011.V1.67.
46. Zhang, H., Gao, Y. (2021). Solving TSP based on an improved ant colony optimization algorithm. *Journal of Physics: Conference Series, 1982,* 012061. DOI 10.1088/1742-6596/1982/1/012061.

47. Skinderowicz, R. (2022). Improving ant colony optimization efficiency for solving large TSP instances. *Applied Soft Computing, 120,* 108653. DOI 10.1016/j.asoc.2022.108653.

48. Siemiński, A., Kopel, M. (2019). Solving dynamic TSP by parallel and adaptive ant colony communities. *Journal of Intelligent & Fuzzy Systems, 37(6),* 7607–7618. DOI 10.3233/JIFS-179366.

49. Ma, A., Zhang, X., Zhang, C., Zhang, B., Gao, Y. (2019). An adaptive ant colony algorithm for dynamic traveling salesman problem. *Journal of Information Science & Engineering, 35(6),* 1263–1277. DOI 10162364-201911-201911250001-201911250001-1263-1277.

50. Mavrovouniotis, M., Müller, F. M., Yang, S. (2015). An ant colony optimization based memetic algorithm for the dynamic traveling salesman problem. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 49–56. Madrid, Spain. DOI 10.1145/2739480.2754651.

51. Kuo, R., Wibowo, B., Zulvia, F. (2016). Application of a fuzzy ant colony system to solve the dynamic vehicle routing problem with uncertain service time. *Applied Mathematical Modelling, 40(23–24),* 9990–10001. DOI 10.1016/j.apm.2016.06.025.

52. Liu, J. L. (2005). Rank-based ant colony optimization applied to dynamic traveling salesman problems. *Engineering Optimization, 37(8),* 831–847. DOI 10.1080/03052150500340504.

53. Eyckelhof, C. J., Snoek, M. (2002). Ant systems for a dynamic TSP. *International Workshop on Ant Algorithms*, pp. 88–99. Berlin, Heidelberg, Springer. DOI 10.1007/3-540-45724-0_8.

54. Mavrovouniotis, M., Yang, S. (2011). Memory-based immigrants for ant colony optimization in changing environments. *European Conference on the Applications of Evolutionary Computation*, pp. 324–333. Berlin, Heidelberg, Springer. DOI 10.1007/978-3-642-20525-5_33.

55. Mavrovouniotis, M., Müller, F. M., Yang, S. (2016). Ant colony optimization with local search for dynamic traveling salesman problems. *IEEE Transactions on Cybernetics, 47(7),* 1743–1756. DOI 10.1109/TCYB.2016.2556742.

56. Montemanni, R., Gambardella, L. M., Rizzoli, A. E., Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization, 10(4),* 327–343. DOI 10.1007/s10878-005-4922-6.

57. Meng, X., Li, J., Dai, X. (2017). A dynamic colored traveling salesman problem and its solution. *2017 36th Chinese Control Conference (CCC)*, pp. 2996–3001. Dalian, China. DOI 10.23919/ChiCC.2017.8027819.

58. Wang, D., Liu, S. (2010). An agent-based evolutionary search for dynamic traveling salesman problem. *2010 WASE International Conference on Information Engineering*, pp. 111–114. Beidai, China. DOI 10.1109/ICIE.2010.34.

59. Khouadjia, M. R., Jourdan, L., Talbi, E. G. (2010). Adaptive particle swarm for solving the dynamic vehicle routing problem. *ACS/IEEE International Conference on Computer Systems and Applications-AICCSA 2010*, pp. 1–8. Hammamet, Tunisia. DOI 10.1109/AICCSA.2010.5586976.

60. Strąk, Ł, Skinderowicz, R., Boryczka, U., Nowakowski, A. (2019). A self-adaptive discrete PSO algorithm with heterogeneous parameter values for dynamic TSP. *Entropy, 21(8),* 738. DOI 10.3390/e21080738.

61. Sabar, N. R., Goh, S. L., Turky, A., Kendall, G. (2021). Population-based iterated local search approach for dynamic vehicle routing problems. *IEEE Transactions on Automation Science and Engineering* (Early Access). DOI 10.1109/TASE.2021.3097778.

62. Groba, C., Sartal, A., Vázquez, X. H. (2015). Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices. *Computers & Operations Research, 56(C),* 22–32. DOI 10.1016/j.cor.2014.10.012.

63. Agarwal, A., Lim, M. H., Er, M. J., Chew, C. Y. (2005). ACO for a new TSP in region coverage. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1717–1722. Edmonton, AB, Canada. DOI 10.1109/IROS.2005.1545460.

64. Kefi, S., Rokbani, N., Krömer, P., Alimi, A. M. (2016). Ant supervised by PSO and 2-Opt algorithm, AS-PSO-2Opt, applied to traveling salesman problem. *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 004866–004871. Budapest, Hungary. DOI 10.1109/SMC.2016.7844999.

65. Dorigo, M., Birattari, M., Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1(4),* 28–39. DOI 10.1109/MCI.2006.329691.

66. Dorigo, M., Maniezzo, V., Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1),* 29–41. DOI 10.1109/3477.484436.

67. Sui, L., Tang, J., Pan, Z., Liu, S. (2008). Ant Colony Optimization Algorithm to Solve Split Delivery Vehicle Routing Problem. *2008 Chinese Control and Decision Conference*, pp. 997–1001. Yantai, China. DOI 10.1109/CCDC.2008.4597462.

68. Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research, 6(6),* 791–812. DOI 10.1287/opre.6.6.791.

69. Engels, C., Manthey, B. (2009). Average-case approximation ratio of the 2-opt algorithm for the TSP. *Operations Research Letters, 37(2),* 83–84. DOI 10.1016/j.orl.2008.12.002.

70. Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing, 3(4),* 376–384. DOI 10.1287/ijoc.3.4.376.