Research Article

Solving Traveling Salesman Problems Based on Artificial Cooperative Search Algorithm

Guangjun Liu,1 Xiaoping Xu,1 Feng Wang,2 and Yangli Tang1

1School of Sciences, Xi’an University of Technology, Xi’an 710054, China
2School of Mathematics and Statistics, Xi’an Jiaotong University, Xi’an 710049, China

Correspondence should be addressed to Guangjun Liu; liuguangjun202106@126.com

Received 18 February 2022; Accepted 21 March 2022; Published 11 April 2022

Academic Editor: Arpit Bhardwaj

Copyright © 2022 Guangjun Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The traveling salesman problem is a typical NP hard problem and a typical combinatorial optimization problem. Therefore, an improved artificial cooperative search algorithm is proposed to solve the traveling salesman problem. For the basic artificial collaborative search algorithm, firstly, the sigmoid function is used to construct the scale factor to enhance the global search ability of the algorithm; secondly, in the mutation stage, the DE/rand/1 mutation strategy of differential evolution algorithm is added to carry out secondary mutation to the current population, so as to improve the calculation accuracy of the algorithm and the diversity of the population. Then, in the later stage of the algorithm development, the quasi-reverse learning strategy is introduced to further improve the quality of the solution. Finally, several examples of traveling salesman problem library (TSPLIB) are solved using the improved artificial cooperative search algorithm and compared with the related algorithms. The results show that the proposed algorithm is better than the comparison algorithm in solving the travel salesman problem and has good robustness.

1. Introduction

Traveling salesman problem (TSP) is not only a basic circuit problem but also a typical NP hard problem and a typical combinatorial optimization problem. It is one of the most famous problems in the field of mathematics [1, 2]. It was first proposed by Menger in 1959. After it was proposed, it has attracted great attention of scholars and managers in operations research, logistics science, applied mathematics, computer application, circle theory and network analysis, combinatorial mathematics, and other disciplines and has become a research hotspot in the field of operations research and combinatorial optimization [3]. In recent years, many scholars have studied the TSP [4–9], and more scholars have expanded the traveling salesman problem [10–12]. At present, the traveling salesman problem is widely used in various practical problems such as Internet environment, road traffic, and logistics transportation [13]. Therefore, the research on TSP has important theoretical value and practical significance.

In the past 30 years, intelligent optimization algorithms have been favored by scholars because of their few parameters, simple structure, and easy implementation such as genetic algorithm (GA) [14–17], differential evolution algorithm (DE) [18], invasive weed optimization (IWO) [19], and particle swarm optimization (PSO) [20]. Artificial cooperative search algorithm was firstly proposed by Pinar Civicioglu in 2013 to solve numerical optimization problems [21]. The algorithm was proposed to simulate the interaction and cooperation process between two superorganisms with predator-prey relationship in the same natural habitat. In nature, the amount of food that can be found in an area is very sensitive to climate change. Therefore, many species in nature will migrate to find and migrate to higher yield breeding areas. It includes the processes of predator selection, prey selection, mutation, and crossover [22–25]. Firstly, the predator population location is randomly generated, the predator location memory is set, and then the prey population location is randomly generated to reorder the prey location, where biological interactions occur during the variation phase. Finally, enter the crossover stage and update...
the biological interaction position through the active individuals in the predator population. Compared with other optimization algorithms, artificial cooperative search algorithm has the advantages of less control parameters and strong robustness and adopts different mutation and crossover strategies. At present, the algorithm has been used to solve scheduling problems, design problems, and other practical problems [26, 27]. To some extent, these methods do solve some practical problems, but there are still some defects such as slow convergence speed and low accuracy. Therefore, it is necessary to improve artificial cooperative search algorithm to improve the performance of the algorithm [28–30].

Aiming at the disadvantages of slow convergence speed, low accuracy, and easy to fall into local optimization of the basic artificial cooperative search algorithm (ACS), this paper proposes a reverse artificial cooperative search algorithm based on sigmoid function (SQACS), that is, after constructing the scale factor by the sigmoid function, the DE/rand/1 mutation strategy of differential evolution algorithm is added in the mutation stage, and the quasi-reverse learning strategy is introduced in the later development stage of the algorithm. In the numerical simulation, the SQACS is used to solve several examples in TSPLIB. The results show that the presented algorithm is feasible.

The remainder of this paper is organized in the following manner. Section 2 describes the TSP model. In Section 3, the basic and improved ACS algorithms are introduced in detail. Solving TSP by the SQACS is described in Section 4. Section 5 covers simulations that have been conducted, while Section 6 presents our conclusion.

2. TSP Model

In general, TSP specifically refers to a traveling salesman who wants to visit n cities, starting from a city, must pass through all the cities only once, and then return to the departure city, requiring the traveling agent to travel the shortest total distance [13]. It is described in graph theory language as follows. In a weighted completely undirected graph, it is necessary to find a Hamilton cycle with the smallest weight. That is, let $G = (V, E)$, $V = \{1, 2, \cdots, n\}$ represent the set of vertices and $E$ represent the set of edges, and each edge $e = (i, j) \in E$ has a non-negative weight $m(e)$. Now it is necessary to find the Hamilton cycle $C$ of $G$ so that the total weight $M(C) = \sum_{E(C)} m(e)$ of $C$ is the smallest. If $d_{ij}$ is used to represent the distance between city $i$ and city $j$, $d_{ij} \geq 0, i, j \in V$, $x_{ij} = \begin{cases} 1, & \text{edge } (i, j) \text{ is on the optimal path} \\ 0, & \text{otherwise} \end{cases}$, then the mathematical model of TSP is as follows:

$$\min Z = \sum_{i \neq j} d_{ij}x_{ij},$$  \hspace{1cm} (1)

$$\text{s.t.} \sum_{j \neq 1} x_{ij} = 1 \quad i \in V,$$  \hspace{1cm} (2)

$$\sum_{i \in S} x_{ij} = |S| - 1 \quad S \subseteq V,$$  \hspace{1cm} (4)

where $|S|$ represents the number of vertices in the set $S$. The first two constraints shown in (2) and (3) indicate that there is only one inbound and one outbound edge for each vertex, and the third constraint (4) indicates that no sub-loops will be generated.

3. Artificial Cooperative Search Algorithm

3.1. Basic Artificial Cooperative Search Algorithm. Basic artificial cooperative search (ACS) algorithm is a global search algorithm based on two populations, which is used to solve numerical optimization problems [21]. Generally, ACS includes the following population initialization, predator selection, prey selection, mutation, crossover, update selection, and so on.

3.1.1. Population Initialization. ACS contains two superorganisms: $\alpha$ and $\beta$, in which $\alpha$ and $\beta$ contain artificial superorganisms equal to the population size ($N$). In the relevant sub-superorganisms, the number of individuals is equal to the dimension ($D$) of the problem. $\alpha$ and $\beta$ ultrasound organisms are used to detect artificial predators and prey sub-superorganisms. The initial values of the sub-superorganism of $\alpha$ and $\beta$ are defined by the following:

$$\alpha_{i,j} = \text{low}_{i,j} + R(0, 1) \times (\text{up}_{i,j} - \text{low}_{i,j}),$$  \hspace{1cm} (6)

$$\beta_{i,j} = \text{low}_{i,j} + R(0, 1) \times (\text{up}_{i,j} - \text{low}_{i,j}),$$  \hspace{1cm} (7)

where $i = 1, 2, \cdots, N$, $N$ is the population size, $j = 1, 2, \cdots, D$, $D$ is the dimension of the optimization problem, $\alpha_{i,j}$ and $\beta_{i,j}$ are the components of the $i$-th sub-superorganism in the $j$-th dimension, $\text{up}_{i,j}$ and $\text{low}_{i,j}$ are the upper and lower limits of the $j$-th dimension search interval, respectively, and $R(0, 1)$ is a random number uniformly distributed on $[0, 1]$.

3.1.2. Predator Selection. At this stage of ACS, the cooperative relationship between two artificial superorganisms is defined. In each iteration of ACS, according to the “if then else” rule, the artificial predator sub-superorganism is randomly defined from two artificial superorganisms ($\alpha$ and $\beta$), and the artificial predator is selected through (8). At this stage of ACS, in order to help explore the search space of the problem and promote the utilization of high-quality solutions, a memory process is developed. In order to provide this memory process, during coevolution, artificial predators

$$\sum_{i \neq j} x_{ij} = 1 \quad j \in V,$$  \hspace{1cm} (3)

$$\sum_{i, j \in S} x_{ij} = |S| - 1 \quad S \subseteq V,$$  \hspace{1cm} (4)

$$x_{ij} \in \{0, 1\} \quad i, j \in V,$$  \hspace{1cm} (5)
will follow artificial prey for a period of time to explore more fertile eating areas.

\[
predator = \begin{cases} 
\alpha, \text{key} = 1, & r_1 < r_2 \\
\beta, \text{key} = 2, & \text{otherwise}
\end{cases}
\]

where \( r_1 \) and \( r_2 \) are uniformly distributed random numbers on the \([0, 1]\) interval, predator represents the predator, key represents the memory that tracks the origin of the predator in each iteration, and its memory is used to improve the performance of the algorithm.

### 3.1.3. Prey Selection

Using the same rules as selecting artificial predators, artificial prey is selected through two artificial superorganisms (\(\alpha\) and \(\beta\)). In ACS, the hierarchical sequence of artificial prey is replaced by random transformation function, which is used to simulate the behavior of superorganisms living in nature. The artificial prey is selected by (9), and the selected prey is used to define the search direction of ACS in each iteration.

\[
prey = \begin{cases} 
\alpha, & r_1 < r_2 \\
\beta, & \text{otherwise}
\end{cases}
\]

where \( r_1 \) and \( r_2 \) are uniformly distributed random numbers in the \([0, 1]\) interval and \(prey\) represents prey.

### 3.1.4. Mutation

Using the mutation process defined in equation (10), the biological interaction position between artificial predator and prey sub-superorganism is simulated. The algorithm embeds a walk process (random walk function) in the mutation process to simulate the foraging behavior of natural superorganisms. In order to promote the exploration of the problem search space and the development of more effective solutions, the variation matrix is generated by using some experience obtained by the artificial predator sub-superorganism in the previous iteration.

\[
X_i(\text{iter} + 1) = \text{predator}_i(\text{iter}) + R \times (\text{prey}_i(\text{iter}) - \text{predator}_i(\text{iter})), \\
R = \begin{cases} 
4 \times a \times (b - c), & r_1 < r_2 \\
\Gamma(4 \times \text{rand}, 1), & \text{otherwise}
\end{cases}
\]

where in order to control the scale factor of biological interaction speed, it is calculated from (13). \(\text{iter}\) is the current number of iterations, \(i \in \{1, 2, \ldots, N\}\), \(a, b, c, \text{rand}, r_1\) and \(r_2\) are random numbers uniformly distributed on the \([0, 1]\) interval, and \(\Gamma\) is the gamma distribution with shape parameter \(4 \times \text{rand}\) and scale parameter 1.

### 3.1.5. Crossover

As defined in equation (11), the active individuals in the artificial predator sub-superorganism are determined by a binary integer matrix \(M\). The initial value of \(M\) is a matrix whose elements in row \(N\) and column \(D\) are all 1. In ACS, those individuals who can only find new biological interaction sites and can participate in migration at any time are called active individuals. The degree of cooperation between individuals in the migration process is determined by the control parameter \(P\), which limits the number of active individuals produced by each artificial superorganism. Then, the parameter controls the number of individuals involved in the crossover process, that is, it determines the probability of biological interaction in the crossover process. The crossover operator of ACS is given by

\[
X_{i,j}(\text{iter} + 1) = \begin{cases} 
\text{predator}_{i,j}(\text{iter}), & M_{i,j} > 0 \\
X_{i,j}(\text{iter}), & \text{otherwise}
\end{cases}
\]

\[
\text{Condition}_{i,j} = \begin{cases} 
1, & r_1 > P \times r_2 \\
0, & \text{otherwise}
\end{cases}
\]

\[
M_{i,j} = \begin{cases} 
M_{i,j}, & r_3 > P \times r_4 \\
M_{i,j} \times \text{Condition}_{i,j}, & \text{otherwise}
\end{cases}
\]

### 3.1.6. Update Selection

The memory key set in the predator selection stage updates the \(\alpha\) and \(\beta\) superorganisms, so as to better select predators and prey at the beginning of the next iteration, so as to strengthen the global search performance. The specific operation is shown in (12) and (13).

\[
a_i(\text{iter} + 1) = \begin{cases} 
\text{predator}_i(\text{iter}), & \text{key} = 1 \\
a_i(\text{iter}), & \text{otherwise}
\end{cases}
\]

\[
\beta_i(\text{iter} + 1) = \begin{cases} 
\text{predator}_i(\text{iter}), & \text{key} = 2 \\
\beta_i(\text{iter}), & \text{otherwise}
\end{cases}
\]

where \(i \in \{1, 2, \ldots, N\}\), \(j \in \{1, 2, \ldots, D\}\). \(\text{predator}_{i,j}\) represents the component of the \(i\)-th predator in the \(j\)-th dimension, and \(M_{i,j}\) represents the component of the \(i\)-th active individual of the predator in the \(j\)-th dimension. \(r_1, r_2, r_3, r_4\) represent uniformly distributed random numbers in the \([0, 1]\) interval, and \(P\) represents the probability of biological interaction. Different experiments with different \(P\) values in the \([0.05, 0.15]\) interval show that ACS is not sensitive to the initial value of its control parameters.

### 3.2. Improved Artificial Cooperative Search Algorithm

Because ACS is not mature and perfect in theory and practice, aiming at its shortcomings such as slow convergence speed, low accuracy, and easy to fall into local optimization, a reverse artificial cooperative search algorithm
based on sigmoid function (SQACS) is proposed. The specific improvement scheme is as follows.

### 3.2.1. Constructing Scale Factor R with Sigmoid Function.

In ACS, the scale factor $R$ controlling the speed of biological interaction is randomly generated, which often makes the algorithm fall into local optimization, which is not conducive to the global search of the algorithm. In order to solve this problem, the following sigmoid function is introduced:

$$y = \frac{1}{1 + e^{-x}} \quad \text{Equation (14)}$$

The sigmoid function is continuous, derivable, bounded, and strictly monotonic, and it is a kind of excitation function [31]. In ACS, according to the mechanism of the biological interaction position, it is known that at the beginning of the algorithm, it needs to quickly approach the optimal position. When it reaches the optimal position, it is necessary to reduce the search speed of the algorithm. Through the sigmoid function and constructing (15), the scale factor $R$ that randomly controls the speed of biological interaction is transformed into a quantity that changes with the number of iterations and is mapped to the range of $[0, 1]$, so that the scale factor $R$ in $[0, 1]$ gradually decreases in the range, so as to find the optimal solution more accurately. In this way, the scale factor $R$ constructed using the sigmoid function is as in equation (15), and its curve is shown in Figure 1.

$$R(\text{iter}) = \frac{1}{1 + e^{\ln 100 \times \text{iter}/G_{\max} - \ln 100}} \times \frac{G_{\max} - \text{iter} + 1}{G_{\max}} \quad \text{Equation (15)}$$

where $G_{\max}$ is the maximum number of iterations, $\text{iter}$ is the current number of iterations, and $R(\text{iter})$ is the scale factor at the $\text{iter}$-th iteration.

### 3.2.2. Quadratic Mutation Strategy.

The DE/rand/1 mutation strategy of the DE is added to the second mutation of the population generated in the mutation stage of the ACS [18]. Research has found that the Gaussian, random, linear, or chaotic changes of the parameters in the DE can effectively prevent premature convergence. Therefore, after the DE/rand/1 mutation strategy of the DE is added to the ACS, a new mutation population is generated, and the next crossover behavior is performed. Thereby, the algorithm can avoid falling into the local optimum and improve the calculation accuracy. The quadratic mutation formula is

$$X_{i,j}(\text{iter} + 1) = X_{i,j}(\text{iter}) + s f \times (X_{2,j}(\text{iter}) - X_{3,j}(\text{iter})) \quad \text{equation (16)}$$

where $i \in 1, 2, \ldots, N$, $j \in 1, 2, \ldots, D$, $\text{iter}$ is the current iteration number, random integers $r_1, r_2, r_3 \in N$, and $r_1 \neq r_2 \neq r_3 \neq i$. The variation factor $s f$ is a control parameter that scales any two of the three vectors and adds the scaled difference to the third vector. In order to avoid search stagnation, the variation factor $s f$ usually takes a value in the range of $[0.1, 1]$.

### 3.2.3. Quasi-Reverse Learning Strategy.

In the later development stage of the algorithm, a better biological interaction position should be found between the populations. Because the position is changing and this change is random, it often prevents it from searching for the optimal solution in a small local area. In order to overcome the above shortcomings, a pseudo-reverse learning strategy is introduced to generate pseudo-reverse populations to increase the diversity of the populations, so that organisms can conduct detailed search for interaction positions in neighboring communities to avoid skipping the optimal solution, and then greedy selection from the current population and quasi-reverse population can effectively find the optimal solution [32–35]. The detailed process is given below:

(i) Assuming that $X = (x_1, x_2, \ldots, x_n)$ is a $n$-dimensional solution, $x_1, x_2, \ldots, x_n \in R$ and $x_i \in [l_i, u_i]$, $i \in [1, 2, \ldots, n]$. Then, the reverse solution $O_X = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ can be defined as

$$\bar{x}_i = l_i + u_i - x_i \quad \text{equation (17)}$$

(ii) On the basis of the reverse solution, the quasi-reverse solution $Q_X = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ can be defined as

$$\bar{x}_i = \text{rand}\left(\frac{l_i + u_i}{2}, \bar{x}_i\right) \quad \text{equation (18)}$$

In this way, the choice of the quasi-inverse solution and the current solution is

$$X = \begin{cases} 
\bar{x}_i, & f(\bar{x}_i) < f(x_i) \\
x_i, & \text{otherwise}
\end{cases} \quad \text{equation (19)}$$

To sum up, the flowchart of the proposed SQACS is shown in Figure 2.
4. Solving TSP by SQACS

Taking the shortest path (i.e., equation (1)) as the objective function, the SQACS is used to solve the TSP. In order to better solve the TSP and realize the transformation between biological organism and TSP solution space, each biological interaction location $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is defined as a sequence of traversing and accessing each city number in this paper. For example, one of the interaction positions $X_i = [1, 3, 2, 4, 5, 6]$ means that the TSP is that the traveler first visits the city with number 1, then successively visits the cities with numbers 3, 2, 4, 5 and 6, and finally returns to the departure city, that is, the city with number 1, and the corresponding objective function is equivalent to the path length of TSP. For the TSP, the shorter the individual’s visit path is, the greater the fitness value is, so the fitness function $f(x_i) = 1/Z_{X_i}$ is selected, where $i = 1, 2, \ldots, n$, $n$ is the number of cities to visit, and the lower and upper limits of variables are 1 and $n$, respectively. The specific steps of SQACS to solve TSP are as follows:

**Step 1.** Population initialization: use the city number to encode the TSP path and randomly generate the arrangement order of $n$ cities.

**Step 2.** Calculate the fitness value of each individual in the population.

**Step 3.** Randomly select the predator and prey population, and then randomly rearrange the position of prey population.

**Step 4.** Calculate the scale factor $R$ of biological interaction velocity.

**Step 5.** Determination of active individuals $M$ in predator population by binary integer mapping.

**Step 6.** Mutation: calculate the location $X$ of biological interaction, i.e., the visit route of the traveler.

**Step 7.** Crossover: if the active individual mapping is greater than 0, update the path to the predator location; otherwise, keep the original location unchanged.

**Step 8.** Reselection of predator and prey populations.

**Step 9.** Judge whether the termination conditions are met. If so, stop the algorithm update and output the optimal position and optimal function value, that is, the shortest route and shortest path value of TSP. Otherwise, return to Step 2.

5. Numerical Simulation

In order to verify the performance of the proposed SQACS, SQACS is tested with GA [14], DE [18], IWO [19], PSO [20], ACS [21], IACS1 [27], and IACS2 [36] to solve four TSPs of different scales in TSPLIB standard database: Oliver 30, Att48, Eil51, and Eil76. In the simulation, in order to compare the results under the same conditions as much as possible, the maximum evaluation times of each algorithm is 2000 and the initial population size is 20. Other parameter settings of the algorithm involved are shown in the corresponding references.

After 30 times of solution by SQACS and the above-mentioned algorithms, the optimal value, average value, and average calculation time of the results are shown in Table 1. By comparing the optimal value and average value of each algorithm to solve the TSP, it can be seen that SQACS can obtain the optimal results in the minimum value and average value, and the difference between the optimal value and average value is the smallest, indicating that SQACS has the strongest stability. From the comparison of index values of calculation time, SQACS performs better than other comparison algorithms on the four datasets. It can be seen that SQACS has good feasibility and robustness in solving TSP.

Figure 3 shows the optimal path diagram of four groups of examples in solving TSP by SQACS. As can be seen from Figure 3, except the path intersection in Att48 dataset, the other three figures are a completely closed loop, and the paths do not cross, so the obtained paths are feasible. Also, the solutions of the optimal route obtained are as follows:
Table 1: Comparison of experimental results of eight algorithms.

| Example | Evaluation criterion | GA | DE | IWO | PSO | ACS | IACS1 | IACS2 | SQACS |
|---------|----------------------|----|----|-----|-----|-----|-------|-------|-------|
| Oliver30| Optimal value        | 427.37 | 423.11 | 431.34 | 428.01 | 452.27 | 452.12 | 434.67 | 420.00 |
|         | Average value        | 432.33 | 435.94 | 449.98 | 437.83 | 475.23 | 468.21 | 440.62 | 421.31 |
|         | Average time/s       | 28.36 | 28.28 | 29.67 | 27.26 | 27.41 | 27.85 | 28.26 | 26.17 |
| Att48   | Optimal value        | 33942.47 | 33793.06 | 33596.81 | 33642.34 | 34663.41 | 34527.23 | 33742.84 | 33516.02 |
|         | Average value        | 403.52 | 405.18 | 419.98 | 407.83 | 430.23 | 423.21 | 405.62 | 397.31 |
|         | Average time/s       | 49.36 | 49.42 | 50.21 | 49.23 | 50.26 | 50.85 | 51.97 | 49.02 |
| Eil51   | Optimal value        | 584.01 | 583.11 | 587.34 | 588.01 | 593.27 | 592.12 | 584.67 | 580.00 |
|         | Average value        | 481.52 | 485.10 | 499.98 | 487.83 | 510.23 | 503.21 | 485.62 | 477.31 |
|         | Average time/s       | 55.54 | 55.35 | 56.36 | 54.65 | 57.25 | 57.78 | 58.02 | 55.27 |
| Eil76   | Optimal value        | 784.01 | 783.11 | 787.34 | 788.01 | 793.27 | 792.12 | 784.67 | 780.00 |
|         | Average value        | 584.01 | 583.11 | 587.34 | 588.01 | 593.27 | 592.12 | 584.67 | 580.00 |
|         | Average time/s       | 85.49 | 85.28 | 86.23 | 85.54 | 86.23 | 86.54 | 87.15 | 85.16 |

Figure 3: The optimal path diagrams of four examples obtained by the SQACS. (a) Oliver30. (b) Att48. (c) Eil51. (d) Eil76.

6→5→30→23→22→16→17→12→13→4
3→9→11→7→8→25→26→29→28→
27→24→15→14→10→21→20→19→18→2
1→6; 2→29→34→41→16→22→3→40
9→1→8→38→31→44→18→7→28→36
30→6→37→19→27→17→43→20→33

46→15→12→11→23→14→25→13→47
24→39→32→48→5→42→10→24→45→
35→26→4→2; 40→19→42→44→37→15→
45→33→39→10→30→34→50→9→49→38
11→5→46→51→27→32→1→22→2
16→21→29→20→35→36→3→28→31→26
Data Availability

The data used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was partly supported by the Innovation Capability Support Program of Shaanxi Province of China (grant no. 2020PT-023), Key Research and Development Program of Shaanxi Province of China (grant no. 2020ZDLGY04-04), and Innovative Training Item for College Student of Shaanxi Province of China.

References

[1] K. Wang, L. Huang, C. Zhou, and W. Pang, “Particle swarm optimization for traveling salesman problem,” in Proceedings of the 2nd International Conference on Machine Learning and Cybernetics, pp. 1583–1585, IEEE Press, Xi’an, China, November 2003.

[2] J. Z. Wang and H. Tang, “A fast algorithm for TSP,” Microelectronics and Computer, vol. 28, no. 1, pp. 7–10, 2011.

[3] G. B. Dantzing and R. H. Ramser, “The truck dispatching problem,” Management Science, vol. 25, no. 6, pp. 37–39, 1959.

[4] H. M. Ge, Application and Research of Improved Genetic Algorithm for TSP Problem, Jiangxi University of Technology, Jiangxi, 2016.

[5] Q. He, L. Y. Wu, and T. W. Xu, “Application of improved genetic simulated annealing algorithm in TSP optimization,” Control and decision making, vol. 33, no. 2, pp. 219–225, 2018.

[6] Y. H. Qi, Y. G. Cai, and G. W. Huang, “Discrete fireworks algorithm with fixed radius nearest-neighbor search 3-opt for traveling salesman problem,” Computer Application Research, vol. 38, no. 6, pp. 36–43, 2020.

[7] Y. L. Tang and Q. J. Yang, “Parameter design of ant colony optimization algorithm for traveling salesman problem,” Journal of Dongguan Institute of Technology, vol. 27, no. 3, pp. 48–54, 2020.

[8] K. S. Cheng, S. D. Xian, and P. Guo, “Adaptive temperature rising simulated annealing algorithm for traveling salesman problem,” Control Theory & Applications, vol. 38, no. 2, pp. 245–254, 2021.

[9] T. B. Tang, Q. Jiang, and Y. Yan, “An artificial bee colony algorithm based on quantum optimization to solve the traveling salesman problem,” Popular Technology, vol. 22, no. 256, pp. 7–10, 2020.

[10] P. Zumel, C. Ferandea, and A. Castro, “Efficiency improvement in multiphase converter by changing dynamically the number of phases,” in Proceedings of the 2006 37th IEEE Power Electronics Specialists Conference, pp. 1–6, Jeju, Korea (South), June, 2006.

[11] H. Pérez and J. S. González, “The one-commodity pickup-and-delivery traveling salesman problem,” Combinatorial Optimization, vol. 2570, pp. 89–104, 2003.
[12] M. Niendorf, P. T. Kabamba, and A. R. Girard, "Stability of solutions to classes of traveling salesman problems," Cybernetics IEEE Transactions on, vol. 1, pp. 34–42, 2015.
[13] X. X. Zhuo, H. X. Wan, and C. L. Zhu, "The application of ant colony and genetic hybrid algorithm on TSP," Value Engineering, vol. 2, no. 6, pp. 188–193, 2020.
[14] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[15] T. Saigal, A. K. Vaish, and N. V. M. Rao, "Gender and class distinction in travel behavior: evidence from India," Ecofeminism and Climate Change, vol. 2, no. 1, pp. 42–48, 2021.
[16] A. Gita and H. Pri, "Pre-Trip decision on value co-creation of peer-to-peer accommodation services," Acta Informatica Malaysia, vol. 3, no. 2, pp. 19–21, 2019.
[17] X. Liu, J. Zhao, J. Li, B. Cao, and Z. Lv, "Federated neural architecture search for medical data security," IEEE Transactions on Industrial Informatics, p. 1, 2022.
[18] R. Storn and K. Price, "Differential evolution: a simple evolution strategy for fast optimization," Dr. Dobb's Journal, vol. 22, pp. 18–24, 1997.
[19] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," Ecological Informatics, vol. 1, pp. 355–366, 2006.
[20] R. Eberhart and J. Kennedy, "Particle swarm optimization," in Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948, Perth, WA, Australia, November 1995.
[21] C. Pinar, "Artificial cooperative search algorithm for numerical optimization problems," Information Sciences, vol. 229, pp. 58–76, 2013.
[22] B. Cao, J. Zhao, Z. Lv, and P. Yang, "Diversified personalized recommendation optimization based on mobile data," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 4, pp. 2133–2139, 2021.
[23] Z. Lv, Y. Li, H. Feng, and H. Lv, "Deep Learning for security in digital twins of cooperative intelligent transportation systems," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 10, pp. 1–10, 2021.
[24] Y. Zhang, F. Liu, Z. Fang, B. Yuan, G. Zhang, and J. Lu, "Learning from a complementary-label source domain: theory and algorithms," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 6, pp. 1–15, 2021.
[25] Y. He, L. Dai, and H. Zhang, "Multi-Branch deep residual learning for clustering and beamforming in user-centric network," IEEE Communications Letters, vol. 24, no. 10, pp. 2221–2225, 2020.
[26] M. S. Turgut and O. E. Turgut, "Hybrid artificial cooperative search–crow search algorithm for optimization of a counter flow wet cooling tower," Advanced Technology and Science, vol. 5, no. 3, pp. 105–116, 2017.
[27] O. E. Turgut, "Improved artificial cooperative search algorithm for solving nonconvex economic dispatch problems with valve-point effects," International Journal of Intelligent Systems and Applications, vol. 6, no. 3, pp. 228–241, 2018.
[28] X. Wu, W. Zheng, X. Xia, and D. Lo, "Data Quality Matters: a case study on data label correctness for security bug report prediction," IEEE Transactions on Software Engineering, p. 1, 2021.