A Novel Sparrow Search Algorithm for the Traveling Salesman Problem

CHANGYOU WU¹, XISONG FU¹, JUNKE PEI¹, AND ZHIGUI DONG²

¹School of Management Science and Engineering, Shandong Institute of Business and Technology, Yantai 264005, China
²Liaoning Institute of Science and Technology, Benxi 117004, China

Corresponding author: Changyou Wu (wuchangyou_81@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 41601593, in part by the National Key Research and Development under Project 2018YFD0300105, and in part by the Startup Foundation for Doctors of Liaoning Institute of Science and Technology College under Grant 1910B04.

ABSTRACT The sparrow search algorithm (SSA) tends to fall into local optima and to have insufficient stagnation when applied to the traveling salesman problem (TSP). To address this issue, we propose a novel greedy genetic sparrow search algorithm based on a sine and cosine search strategy (GGSC-SSA). First, the greedy algorithm is introduced to initialize the population and to increase the diversity of the population. Second, genetic operators are used to update the population, balancing global search and local development capabilities. Finally, the adaptive weight is introduced in the producer update to increase the adaptability of the algorithm and to optimize the quality of the solution, and a sin-cosine search strategy is introduced to update the scroungers. In addition, the GGSC-SSA is compared with the genetic algorithm (GA), simulated annealing (SA), particle swarm optimization (PSO), grey wolf optimization (GWO), ant colony optimization (ACO) and the artificial fish (AF) algorithm on TSP datasets for performance testing. We also compare it with some recently improved algorithms. The results of the simulations are encouraging; the GGSC-SSA significantly enhances the solution precision, optimization speed and robustness.

INDEX TERMS Sparrow search algorithm, traveling salesman problem, greedy algorithm, genetic operators, sin-cosine search strategy, combinatorial optimization.

I. INTRODUCTION

The core idea of swarm intelligence algorithms is to find optimal solutions by simulating the living habits and behavior rules of creatures in nature and by searching for the spatial distribution of solutions in a limited space. Domestic and foreign scholars have proposed a large number of swarm intelligence algorithms through the swarm behavior of various swarms of intelligent creatures such as ants, bees, birds, wolves, fireflies, sailfish, and sparrows, such as Particle Swarm Optimization (PSO) [1], Firefly Algorithm (FA) [2], Ant Colony Optimization (ACO) [3], Grey Wolf Optimization (GWO) [4], Sailfish Algorithm (SFO) [5] and Sparrow Search Algorithm (SSA) [6] and so on [7]–[10]. Among them, the sparrow search algorithm was proposed by Jiankai Xue and Bo Shen in 2020. Compared with other intelligent algorithms, the SSA has the advantages of simple implementation, strong scalability, robustness, and high solution efficiency. Since proposed, it has attracted the attention of lots of scholars [11], [12], [12]–[18].

Swarm intelligence algorithms are widely used in engineering optimization problems such as the knapsack problem [19], path planning [20]–[22], robot control [23], data mining [24], [25] and other issues [26]–[29]. The population characteristics and behavior of the heuristic algorithm are conducive to solving the discretization problem [30], [31]. The traveling salesman problem (TSP) is a classic combinatorial optimization problem [32], [33]. It is one of the standard test problems used in the performance analysis of swarm intelligence algorithms and has NP-hard characteristics. There are other practical problems that can be solved in real life by abstracting and extending the TSP. For this reason, the TSP remains a popular topic in current research on new and different heuristic strategies, and it is significant in both theory and practice. The continuous development of swarm intelligence algorithms has shed new light on NP-hard problems. An increasing number of algorithms have been successfully applied to TSPs, including PSO [34], the GA [35], SA [36] and the SSA.
In basic PSO, there are few parameters that need to be adjusted, and the algorithm is easy to implement. However, the accuracy of solving the high-dimensional test function set is slightly insufficient. The GA has strong parallelism and global search capabilities. However, it is easily falls into local optima. SA has a strong ability to jump out of local optima but has many parameters that need to be adjusted, and the cooling time directly affects the efficiency of the algorithm. The advantages of the SSA mainly include strong robustness, simple implementation and few parameters. However, due to the random generation problem of early producer sparrows, the algorithm falls into local optima. A good SSA for solving the TSP should have the following characteristics: (1) Each improvement strategy should be adjusted according to the size of the TSP. (2) The use of loop sentences should be reduced in the improved SSA to increase the speed of the algorithm. (3) In the entire solution process, a balance between exploration and development should be achieved. (4) For small or large TSP instances, the algorithm should be able to converge to the global optimal solution with high accuracy. In response to the above problems, this article proposes an improved SSA (the greedy genetic sine cosine sparrow search algorithm (GGSC-SSA)).

In this paper, the main effort is to improve the convergence speed and the solution accuracy on TSP instances of different sizes. The main contributions of this work to research on the TSP are as follows:

- We propose an improved SSA. The GGSC-SSA offers three main improvements over the basic SSA:
  1) The greedy algorithm is introduced into the SSA. First, the greedy algorithm is a simpler and faster design technique for finding a higher-quality TSP solution set, and it is used when the SSA initializes the population. Then, the top-down, iterative method is used to make successive greedy choices, and each time a greedy choice is made, the problem is reduced to a smaller subproblem, increasing the ability of the initial SSA to jump out of local optima.
  2) We apply genetic crossover and mutation strategies to update the SSA population.
  3) We introduce dynamic adaptive weights to update the position of the producers.
  4) We introduce the sine and cosine search strategy to expand the search range of the scavenger, effectively preventing the algorithm from prematurely converging.
- The GGSC-SSA and other classic algorithms proposed in the literature are tested on the TSPLIB test set. It is found that the GGSC-SSA is superior to other algorithms in terms of solution time and solution accuracy.
- We compare the proposed GGSC-SSA with other improved algorithms that have recently been presented and show that the proposed GGSC-SSA has great advantages in terms of solution quality.

The remainder of this report is structured in the following manner. In Section II, the latest SSA and TSP studies are presented. In Section III, some basic knowledge is briefly presented. In Section IV, the proposed GGSC-SSA is described in detail, including the greedy strategy initializing the population, genetic variation strategy, and adaptive inertia weight investigated. In Section V, a series of TSP instances are simulated, and the results of the experiments are analyzed. Finally, a summary of the paper with conclusions and directions for future improvement is presented in Section VI.

II. THE RELATED WORK

The TSP is of great significance in the history of operations research. In 1952, Danzig and others successfully solved the TSP examples of 48 cities in different states in the United States and 49 cities in the District of Columbia, introducing the problem to more people for the first time. The significance of combinatorial optimization research has also improved the accuracy in solving discrete problems. With the rapid development of heuristic algorithms, an increasing number of scholars have tried to apply different heuristic algorithms to solve the TSP. The TSP is an NP-hard problem; thus, there are no algorithms that can find the optimal solution in polynomial time, so it is very important to study the swarm intelligence algorithm of the TSP. A large number of new meta-heuristic algorithms are produced. As a result, this natural-inspired algorithm design method has been widely criticized. How to design an improved algorithm for solving practical problems in your own domain is very important [37], [38]. This paper chooses the SSA to solve the TSP, which is a very large challenge, because the SSA has just recently been proposed and applied to the TSP for the first time and has not been widely used.

Most methods that can be used to solve the TSP are usually divided into two categories: (1) heuristic algorithms and (2) exact algorithms. Although there are some accurate methods for solving the TSP with priority constraints in the literature, such as branching and shearing and dynamic programming, exact solutions cannot be obtained as the scale of the TSP continues to grow, and exact methods can solve only a small part of this problem. In recent years, due to the complexity of the TSP, metaheuristic algorithms such as the tabu search algorithm, simulated annealing (SA) algorithm and genetic algorithm (GA) have been proposed in the literature to solve this problem. This article briefly reviews the related literature published in recent years.

In 2019, Al et al. [22] presented a parallel version of the 2-opt algorithm based on Optical Transpose Interconnection System (OTIS) to solve the TSP. Reference [39] proposed a novel Artificial Bee Colony (ACO) algorithm based on a swap sequence. The experimental results show that the improved ACO has a good performance on the TSPLIB test set, although it has insufficient solution time. In [40], Kim and Moon proposed a traveling salesman problem with a drone stations (TSP-DS) based on the characteristics of UAV system delivery services. Zhukova et al. [41] developed a hybrid and accurate algorithm for solving the asymmetric traveling salesman problem (ATSP). The key technology is to predict the
solution time of the exact solution based on the combination of branch and bound method and approximate algorithm. The experimental results show that the proposed algorithm solves the asymmetric traveling salesman problem more effectively. Zhu et al. [42] proposed a novel ant colony optimization based on pearson correlation coefficient. A large number of simulations in TSPLIB show that the proposed improved ant colony algorithm can obtain a better solution for small, medium and large-scale TSP. Zhong et al. [43] introduced a discrete Pigeon-inspired optimization (DPIO) algorithm which uses the Metropolis acceptance criterion of simulated annealing algorithm to solve the TSP problem. Zhao et al. [44] converted the energy-related mission plan into a dynamic traveling salesman problem, and proposed a hybrid method combining the Gaussian pseudospectral method and the genetic algorithm (GPM-GA). The experimental results show the effectiveness of GPM-GA in terms of energy efficiency, computational efficiency and smoothing of the attitude trajectory.

In 2020, Yang et al. [45] proposed a novel game-based ACO(NACO) that includes two ant colony systems and introduces mean filtering to process pheromone distribution, which effectively solves the problem that basic ACO is easy to fall into local optimum. In [46], ABC and Greedy Algorithm were combined in a novel manner to form an improved ABC, which was successfully applied to multi-objective traveling salesman problem. In [47], a modified version of social group optimization (SGO) has more competitive results when solving TSP, and its convergence speed is better than GA and discrete particle swarm optimization. The efficiency of solving large-scale TSP problems has also been proved. Tu-san et al. [48] introduced a novel variant of the TSP, called the intermittent travelling salesman problem (ITSP), and proposed a branch and bound method to solve the optimality problem of ITSP. Reference [49] proposed a new ACO based on dynamic adaptive method. In addition, the experiment of the variant ant colony algorithm tested on the TSPLIB instance shows that this method has better algorithm performance. Tran et al. [50] designed a UAV trajectory that reduces energy consumption based on the traveling salesman problem, and proposed a new heuristic search and dynamic programming (DP) method. The results show that the DP algorithm is close to exhaustion with significantly reduced complexity. Cinar et al. [86] proposed an improved Tree Seed algorithm to solve TSP. Experimental results show that DTSA is another qualified and competitive solver on discrete optimization. In [51], a novel heuristics mathematical Eq.tion is proposed, which is based ACO to minimize travel costs. In [52], an analog electronic computing system has been successfully applied to the traveling salesman problem. The system spontaneously and dynamically simulates the effective foraging behavior of similar organisms, and realizes the flexibility and flexibility of high problem mapping, and has high application potential. Popescu et al. [53] have successfully developed a novel approach to approximate the Shapley value of Euclidean TSG, which is inspired by the one-dimensional extended case. This method can effectively reduce the computational complexity of the traveling salesman problem. Cavaleri et al. [54] proposed a method of distance balance diagram which effectively solves the traveling salesman problem. Reference [55] introduced a metaheuristic approach, namely (Ib/ub)Alg, which was successfully applied to the close-enough traveling salesman problem. In [56], a strategy to consider the size of the time window is proposed, which effectively improves the efficiency of solving the traveling salesman problem with a time window(TSPTW). Reference [57] proposed an agglomerative greedy brain storm optimization algorithm(AGBSO) for solving the TSP.

In 2021, several scholars have been proposed various methods for TSP such as Pan et al. [58] proposed a novel Ant Colony Optimization based Pheromone refactoring mechanism. This algorithm effectively solves the problem of large-scale TSP falling into local optimality and slower convergence speed. Yang et al. [45] introduced an improved ACO for symmetric TSP problem based on Long Short-Term Memory network and adaptive Tanimoto communication strategy. Zhang et al. [59] proposed an improved whale optimization algorithm based on the adaptive weight, Gaussian disturbance, and variable neighborhood search strategy. Experimental results show that, compared with recent related algorithms, this algorithm has better optimization performance and higher efficiency. Zelinka et al. [60] introduced a gamesourcing approach to replace ACO. The algorithm is in the form of a maze, TSP nodes move within the maze, and then the performance of the algorithm is evaluated and compared with some well-known versions of ACO. Experiments show that this method achieves better results on well-known NP-hard optimization problems such as TSP. Yousefikhoshbakht [61] provided an improved particle swarm algorithm that shifts the particles to the best particles. This method takes into account the concept of randomness and prevents premature convergence of the algorithm. Wu et al. [62] combined k-means, top-layer ACS, and bottom-layer ACS to solve large-scale TSP. The experimental results show that the solution efficiency of the algorithm is effective. Vasquez et al. [63] studied the Traveling Salesman Problem with Drone(TSP-D), which is a variant of the TSP problem, and proposed a mixed-integer programming Eq. and a Benders-type precise algorithm. Finally, the proposed method has been empirically tested in a randomly generated example to prove its effectiveness. Sun et al. [64] studied the generalization ability of the machine learning model, which can effectively solve the problem of the classic traveling salesman problem (TSP). Experiments have proved that the model can find a better solution from the optimization problem. Even if tested on different TSP problem variants, the model can still make useful predictions and improve the solution quality of the TSP problem. Steiber et al. [65] introduced a new type of dynamic model to solve the problem of multiple traveling salesmen(MTSP) with moving targets in an accurate way. Compared with other mathematical models and swarm
intelligence algorithms on randomly generated large-scale examples, the results show that the proposed model has strong solution efficiency and robustness. Silva et al. [66] invented a new technique for parallel computing called Multi Improvement (MI). In addition, three dynamic programming algorithms for solving the Maximum Multi Improvement Problem (MMIP) are developed, and the effective solution of the traveling salesman problem is given. In order to solve the inefficiency of the dynamic programming method in solving the sequential order problem (SOP), Salii et al. [67] proposed a new dynamic programming method with lower bound heuristic parameters. The scheme is tested on an example of TSPLIB, and the effectiveness of the proposed method is proved.

Overall, a large number of scholars have contributed to the solution of the TSP, and it has been proven that the TSP has value in real life. Although scholars have made some progress in combinatorial optimization theory and application, there are still some key issues. The improved algorithms proposed in [57], [59] and [68] can reach an ideal state when solving small-scale symmetric TSPs. The improved algorithms proposed in [34], [41], [45] and [69] have achieved satisfactory solution accuracy when solving symmetric TSPs, but the solution time is very unsatisfactory. References [40], [70] and others have converted practical problems into TSPs well, but the solving efficiency of the algorithm is an obvious disadvantage. Note that the cited in this section are representative of only a small portion of the related work on the TSP. Due to the increasing number of studies on the TSP, it is difficult to summarize all the related work. Therefore, to further understand the solutions related to the TSP and its variants, it is recommended that readers study the work introduced in [71], [72] and [73]. On the other hand, readers who want to know more about the possible applications of the SSA can refer to [11]–[14], [16]–[18].

### III. THE BASIC PROBLEM DESCRIPTION

#### A. TRAVELING SALESMAN PROBLEM (TSP)

The TSP is a classical combinatorial optimization problem. In this problem, a businessman needs to visit several cities and then return to the starting city: each city can be visited only once, and the shortest path needs to be determined. Although the constraints are simple, it is extremely complicated to solve as the number of places increases. Currently, the solutions to this combinatorial optimization problem can be roughly divided into two categories: precise optimization algorithms and metaheuristic algorithms. The existing research shows that precise optimization algorithms can effectively solve small-scale TSPs, while metaheuristic algorithms are more suitable for medium- and large-scale TSPs. Precise optimization algorithms include the gradient descent method, Newton method, dynamic programming, enumeration method and others. Meta-heuristic algorithms include SA, PSO, the SSA, etc. The SSA employs the ideas of evolution and a flexible behavior strategy and provides a novel method for solving the TSP. The mathematical model of the TSP can be expressed as:

\[
\min Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}x_{ij} \\
\text{s.t.} \\
\sum_{j=1}^{n} x_{ij} = 1, \quad i \in V \\
\sum_{i=1}^{n} x_{ij} = 1, \quad j \in V \\
\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \in V \\
x_{ij} \in \{0, 1\}
\]  

(1)

In Eq. 1, \(d_{ij}\) represent the distance between each vertex, \(x_{ij}\) represent the decision variable, \(x_{ij} = 1\) is on the loop, \(x_{ij} = 0\) is not on the loop. Corresponding to the Hamiltonian cycle, \(G = (V, E)\), \(V\) is the vertex set, \(E\) is the edge set, the third set of constraints are sub-tour elimination constraints.

#### B. SPARROW SEARCH ALGORITHM (SSA)

The inspiration of the SSA comes from the foraging behavior of sparrows in nature. Sparrows have excellent flying ability and strong vigilance and are resident birds that like to live with humans. According to the different foraging behaviors, sparrows are divided into two types: producers and scroungers. Producers are responsible for finding food and

---

**Algorithm 1 Pseudocode of the Basic SSA**

**Input:**  
- \(N\): the number of sparrows  
- \(PD\): the number of producers  
- \(ST\): the safety value  
- \(SD\): the number of sparrows that perceive danger  
- \(R\): the alarm value

**Output:** \(X_b\): the global optimal individual  
- \(F_b\): the best fitness value

1: Initialize a population of \(N\) sparrows and the parameters.
2: \(g = 1\) (Record the number of iterations).
3: while \(g < MaxIter\) do
4:  
5:  
6: \(R = \text{rand}(1),\)
7: for \(i = 1: PD\) do
8:  
9:  
10: for \(j = (PD + 1): N\) do
11:  
12:  
13: for \(J = 1: SD\) do
14:  
15:  
16: \(g = g + 1\)
17: \(X_b, F_b\) return

---

**In Eq. 1,** \(d_{ij}\) **represent the distance between each vertex,** \(x_{ij}\) **represent the decision variable,** \(x_{ij} = 1\) **is on the loop,** \(x_{ij} = 0\) **is not on the loop. Corresponding to the Hamiltonian cycle,** \(G = (V, E)\), **\(V\) is the vertex set,** \(E\) **is the edge set,** the **third set of constraints are sub-tour elimination constraints.**
have higher energy reserves. Scroungers follow and monitor producers and have low energy reserves; some scroungers compete with producers for food. When predators (natural enemies of sparrows) appear in a foraging area, a sparrow recognizes the danger and immediately enters alert mode. The basic SSA pseudocode is shown in Algorithm 1.

In the SSA, it is necessary to simulate the sparrow foraging process to find the solution of the target problem. The position of the sparrows is denoted as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix}$$

(2)

where \( n \) represents the population size of sparrows and \( d \) is the dimension of the variables to be optimized. The fitness values of all sparrows are expressed by the following matrix:

$$F_X = \begin{bmatrix} f([x_{11}, x_{12}, \ldots, x_{1d}]) \\ f([x_{21}, x_{22}, \ldots, x_{2d}]) \\ \vdots \\ f([x_{n1}, x_{n2}, \ldots, x_{nd}]) \end{bmatrix}$$

(3)

Each value in \( F_X \) represents the value of the individual. The higher the fitness value of a sparrow, the easier it is for it to obtain food during foraging. Additionally, they can act as producers that are responsible for the food search of the whole population and can find food outside the search space. According to Eq. 2 and 3, in each iteration, the producers update the position, and the formula of the position update is as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^t \cdot \exp \left( \frac{x_{i,j}^t - \text{Iter}_{\text{max}}}{\alpha} \right) & \text{if } W < ST \\ X_{i,j}^t + Q \cdot L & \text{if } W \geq ST \end{cases}$$

(4)

where \( t \) represents the current iteration. \( X_{i,j}^t \) represents the value of \( t \) iterations of the \( i \)th sparrow in the \( j \)th dimension. \( \alpha \) is the random number in the interval \([0, 1]\). \( \text{Iter}_{\text{max}} \) represents the maximum number of iterations of the current population. \( W (W \in [0, 1]) \) and \( ST (ST \in [0.5, 1.0]) \) represent the alarm threshold and safety threshold respectively. \( Q \) is a random number that follows the normal distribution. \( L \) shows a \( 1 \times d \) matrix in which each element inside is 1. When \( W < ST \), the sparrow population is in a safe state and continues to forage, while the producers search for food in a large range. If \( W \geq ST \), then predators are present in the sparrow population, and all sparrows need to immediately fly to a safe area.

Except for the producers, all the sparrows in the population are scroungers, and their positions are updated with the following formula:

$$X_{i,j}^{t+1} = \begin{cases} Q \cdot \exp \left( \frac{X_{i,j}^t - X_{\text{OP}}^t}{\beta} \right) & \text{if } i > \frac{n}{2} \\ X_{i,j}^{t+1} \cdot X_{\text{OP}}^t + X_{i,j}^{t+1} \cdot A^T (AA^T)^{-1} \cdot L & \text{otherwise} \end{cases}$$

(5)

where \( X_{\text{OP}} \) is the best location of the producers. \( X_{\text{Worst}} \) denotes the worst position of the scroungers in the current iteration. When \( i > n/2 \), the fitness value of the \( i \)th scrounger is low, and it is unable to access enough food.

When the sparrow population forages at the feeding source, 10%-20% of the sparrows perform early warning work to prevent being attacked by predators. The updated position of sparrows with early warning capability can be shown as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{\text{Best}}^t + \beta \cdot \frac{X_{i,j}^t - X_{\text{Best}}^t}{|f_i - f_b| + \delta} & \text{if } f_i \neq f_b \\ X_{i,j}^t + H \cdot \frac{X_{i,j}^t - X_{\text{Worst}}^t}{|f_i - f_b| + \delta} & \text{if } f_i = f_b \end{cases}$$

(6)

where \( X_{\text{Best}} \) is the current global optimal position. \( f_i \) represents the fitness value of the current sparrows. \( f_b \) and \( f_w \) express the current best and worst fitness values, respectively. \( \beta \) represents standard normally distributed random numbers with an average value of 0 and a variance of 1. \( H \) is a random number in the interval \([0, 1]\), and controls the moving direction of sparrows and the adjustment of the step size. \( \delta \) is the minimum constant, which prevents the situation where the denominator is 0 and the fitness value of the current sparrow is the global worst. When \( f_i \neq f_b \), the sparrows are at the edge of the foraging area and are vulnerable to predators. \( f_i = f_b \) shows that the sparrow at the center of the population is aware of the danger and needs to quickly approach other sparrows to readjust the foraging strategy.

**IV. IMPROVED SSA FOR THE TSP**

A. THE GREEDY ALGORITHM INITIALIZES THE POPULATION

Dynamic programming algorithms usually give solutions with a bottom-up method, while greedy algorithms, in contrast, use the method of constructing the optimal solution stepwise with top-down method and make greedy choices in an iterative fashion. Every time a greedy choice is made, the optimization problem is simplified to a smaller subproblem [74]–[76]. The nature of the greedy algorithm means that the global optimal solution of the problem can be achieved through a series of local optimal choices, that is, greedy choices. Greedy algorithms have been widely used in path planning [77], job-shop scheduling [78] and other issues [79]. In this paper, the greedy algorithm is used to replace the random generation of the population in the original SSA, which not only maintains the diversity of the population, but also improves the efficiency of the algorithm. Note that greedy algorithm has some shortcomings, such as the inability to guarantee that the final solution is the optimal solution.

B. THE GENETIC OPERATORS UPDATE THE POPULATION

1) OX CROSSOVER OPERATOR

To increase the diversity of sparrow population, OX crossover operation in the GA is used after population initialization [80]. Assume that the parent individuals are as follows:
Algorithm 2: Pseudocode of the GGSC-SSA

**Input:** MaxIter: the maximum number of iterations  
N: the number of sparrows  
PD: the number of producers  
ST: the safety value  
SD: the number of sparrows that perceive danger  
R: the alarm value

**Output:** $X_b$: the global optimal individual  
$F_b$: the best fitness value

1: Structure coding method. Initializes the sparrow population $N$ and the parameters with the greedy algorithm.
2: $g = 1$(Record the number of iterations).
3: while $g < $MaxIter do
4: Calculate the fitness value of all individuals.
5: Update the population with the OX crossover operator.
6: $R = \text{rand}(1)$
7: for $i = 1$ to $PD$ do
8: Use Eq.8 to update the producer location.
9: end for
10: for $j = (PD + 1)$ to $N$ do
11: Use Eq.5 to update the producer’s location.
12: Use the sine and cosine search strategy to enhance global ability using Eq.9
13: end for
14: for $J = 1$ to $SD$ do
15: Use Eq.6 to update the producer’s location.
16: end for
17: Get the current new location.
18: Compare the new and old individuals.
19: end while
20: return $X_b$ and $F_b$

| Parent individual 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------------|---|---|---|---|---|---|---|---|
| Parent individual 2 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

We randomly select two crossover positions 3 and 6, then move the crossover segment of parent 2 to the front of parent 1, and the crossover segment of parent 1 to the front of parent 2 and delete the duplicate individuals in turn to form two offspring individuals. This is expressed as follows:

| Child individual 1 | 6 | 5 | 4 | 3 | 1 | 2 | 7 | 8 |
|-------------------|---|---|---|---|---|---|---|---|
| Child individual 2 | 3 | 4 | 5 | 6 | 8 | 7 | 2 | 1 |

2) Mutation Strategy

The purpose of the GA mutation operator is twofold: first, to give the GA local random searching abilities [80], [81]. When the GA is close to the optimal solution neighborhood through the crossover operator, the local random search ability of the mutation operator can accelerate the convergence to the optimal solution [82]. In the SSA, the producer approaches the global optimal solution from the beginning iterations, and the search range is restricted, so it is easily trapped in local optima. Therefore, the variation strategy of GA is introduced to update the position of the producer and the exchange operation. The introduction of a mutation strategy improves the search efficiency and global optimization ability of the producers. Dynamic adaptive weights are imported to update the position of the producers [83]. The location of the producers are updated with the weight coefficient [84], and the formula is as follows:

$$
\lambda = \begin{cases} 
\lambda_{\min} - \frac{(\lambda_{\max} - \lambda_{\min})(f_i - f_{\text{avg}})}{f_{\text{avg}} - f_{\text{min}}} & \text{if } f_i < f_{\text{avg}} \\
\lambda_{\max} & \text{otherwise}
\end{cases}
$$

(7)

$$
X_{i,j}^{t+1} = \begin{cases} 
X_{i,j}^t + r_1 \cdot \sin(r_2) \cdot \left| r_3 \cdot P_i^t - X_{i,j}^t \right| & r_4 < 0.5 \\
X_{i,j}^t + r_1 \cdot \cos(r_2) \cdot \left| r_3 \cdot P_i^t - X_{i,j}^t \right| & r_4 \geq 0.5
\end{cases}
$$

(8)

where $\lambda$ is a random number in the interval [0,1]. Two random numbers are generated, and after comparison, $\lambda_{\min}$ and $\lambda_{\max}$ are obtained; $f_{\text{avg}}$ is the average of the current global optimal and worst fitness values. Through the introduction of the dynamic weight coefficient, the adaptation of the algorithm is effectively increased, and the coefficient is adjusted with the number of iterations to better perform a global search.

**C. Sine and Cosine Search Strategy**

In the SSA population update, the scrounger location update is mainly guided by the producers, which results in a rough optimization effect. To further improve the convergence accuracy and optimization effect of the algorithm, and to balance the local development and global search abilities, a sine and cosine search strategy is introduced [85]. After the scroungers are updated according to the location of the producer, a sine and cosine search is carried out to obtain the optimal feasible solution. The mathematical expression of the sine and cosine search strategy is as follows:

$$
X_{i,j}^{t+1} = \begin{cases} 
X_{i,j}^t + r_1 \cdot \sin(r_2) \cdot \left| r_3 \cdot P_i^t - X_{i,j}^t \right| & r_4 < 0.5 \\
X_{i,j}^t + r_1 \cdot \cos(r_2) \cdot \left| r_3 \cdot P_i^t - X_{i,j}^t \right| & r_4 \geq 0.5
\end{cases}
$$

(9)

$$
r_1 = a - a \cdot \frac{t}{\text{MaxIter}}
$$

(10)

where $r_1$ will increase with the increase of iteration times MaxIter, $a$ is a constant, the value in experiment is 2, $r_2$ is a random number the interval [0, 2$\pi$], $r_3$ is a random number between [0,2], and $r_4$ is the uniformly distributed random number on [0,1]. The function of sine and cosine search is to make the algorithm effectively prevent premature convergence and improve the convergence accuracy to a certain extent, so as to improve the efficiency of each iteration.

**D. Construction Coding Mode**

Since the basic sparrow search algorithm cannot directly solve some discrete optimization problems like TSP, it is necessary to reconstruct the search space of the algorithm and redefine the objective function according to the actual problem [86], [87]. Therefore, the solution range of the sparrow search algorithm needs to be transformed into a two-dimensional continuous space. Only by defining the value range of the independent variable and the objective
function expression, the optimal solution and the corresponding independent variable value can be obtained.

After introducing the TSP problem into the SSA, it can be defined as the sparrow population size as $N$ and the number of cities as $D$. In the $D$ dimensional city search space, the position $X_i$ of the $i$-th sparrow is defined as a set of different positive integer sequences, and the $N$ sparrows search for prey in the $D$ dimensional space, that is, the search space domain $K$ is an entity matrix, the formula is as follows:

$$K = \begin{bmatrix} X_1^1 & X_1^2 & \cdots & X_1^D \\ X_2^1 & X_2^2 & \cdots & X_2^D \\ \vdots & \vdots & \ddots & \vdots \\ X_N^1 & X_N^2 & \cdots & X_N^D \end{bmatrix}$$  \quad (11)

The first row of the matrix indicates the position sequence of the first sparrow in the search space, and the last row indicates the position sequence of the $N$th sparrows in the search space.

After constructing the sparrow population search space matrix and the sparrow position sequence expression method, another important problem is to solve the distance matrix $L$ in the TSP problem. For the TSP with the number of cities $D$, the distance matrix $L$ formed by the distance $d_{i,j}$ between the $i$th city and the $j$th city can be expressed as:

$$L = \begin{bmatrix} d_{(1,1)} & \cdots & d_{(1,N)} \\ \vdots & \ddots & \vdots \\ d_{(N,1)} & \cdots & d_{(N,N)} \end{bmatrix}$$  \quad (12)

The distance of $d_{(N,N)}$ is 0. Through the above description, the relationship expression between the search space and the objective function can be constructed as follows:

$$C = \min\ K \left( \sum_{i,j} d_{(i,j)} \right)$$  \quad \text{s.t.}  \quad \begin{cases} (i,j) \leq D \\ (i,j) \in N \\ i \neq j \end{cases}$$ \quad (13)

Among them, $i$ and $j$ represent the city number, $\min K$ represents the optimal population position, and the distance matrix sum corresponding to $\sum d_{(i,j)}$. The function of this function is to read the cumulative sum of the distance between the position sequence in each sparrow position matrix $K$ and the corresponding distance matrix $L$.

Random initialization of different sparrow individuals will generate different solution vectors, and calculate the distance between different solution vectors. The optimization of objective function is determined through function. If the solution obtained is better than the previous one, it will be replaced with a better solution and used as the optimal solution for the current iteration of this sparrow. Otherwise, it remains unchanged, and the next line of judgment is continued until all the sparrow solution vector optimizations are all completed.

According to the above re-encoding settings for the sparrow, the improvement process of the sparrow search algorithm can be abstracted into a combined optimization model on a continuous space, and the sparrow search algorithm can be applied to the TSP problem.

**E. GGSC-SSA FOR THE TSP**

The GGSC-SSA is based on the initial SSA and introduces a greedy algorithm to initialize the population. When the producers and scroungers are updated, the crossover and variation strategies in the GA are embedded to optimize the results of sparrow traversal. Finally, global optimization is carried out according to the position of the early warning sparrow. The steps for solving the TSP with a combined a greedy genetic strategy and sine and cosine SSA are as follows:

1. Initialize the TSP city information and GGSC-SSA parameters, and discretize the algorithm;
2. Initialize the sparrow population with the greedy algorithm;
3. Calculate the fitness values of all sparrows in the population, and find the sparrow with the best fitness value;
4. Select part of sparrows that have the highest fitness value as producers, and update their positions according to Eq. 8. Update the positions of the remaining sparrows as scroungers according to Eq.5;
5. Randomly select early warning sparrows from the population, and update the positions based on Eq.6;
6. Use the sine and cosine search strategy through the updated warning sparrows to prevent the algorithm from falling into local convergence;
7. According to the current state of the sparrow population, update the optimal position and fitness of the entire population, as well as the worst position and fitness;
8. Judge whether all individuals are traversed, if so, proceeding to the next step, otherwise, jumping to step 3;
9. Judging whether the maximum iteration times have been reached, if so, proceed to the next step, otherwise, go to step 2;
10. Introduce the program operation, output the optimal result.

The detailed process of the GGSC-SSA algorithm for solving the TSP is shown in Figure 1. The GGSC-SSA solves TSP pseudocode as shown in Algorithm 2.

**V. EXPERIMENTAL STUDIES**

In this section, the experiments conducted on the TSP to test the improved SSA are introduced in detail. First, we analyze the parameters related to the algorithm and the related components of the improved SSA. For the TSP dataset, the 36 examples used in this article are from the TSPLIB benchmark. In these 36 instances, the number of city nodes ranges from 22 to 1291. For each instance, the algorithm described in this article is run 50 times. To evaluate the performance of the GGSC-SSA, we compare it with the following intelligent optimization methods: (1) traditional intelligent
algorithms, such as the GA, SA, PSO, GWO, ACO, and AF [88] and (2) other improved intelligent algorithms [6], [45], [57], [61], [89]–[93]. The simulation experiment is run on a computer equipped with an Intel Core i7-10600 processor, the program is created in a Windows 10 environment, and the programming software used is MATLAB2018a.

A. COMPONENT TESTING AND PARAMETER SETTINGS
For many intelligent optimization algorithms, parameter tuning is the key to algorithm optimization performance. After analysis of the existing literature related to the SSA [6], [11], [12], [18], [94], [95], the parameters of the GGSC-SSA are determined through repeated and in-depth revisions, as shown in Table 1. Note that the parameters in the sine and cosine search strategy are set according to the literature [85].

To test the performance of each component of the GGSC-SSA and to analyze the impact on the basic SSA, the three components are added to the SSA and renamed. Additionally, comparative experiments are carried out on the Ulysses22, Eil51, Berlin52, Rat99 and Ch130 datasets. The results are shown in Table 2. The algorithm variants for different components are defined as follows:

1) The SSA introduced with the greedy algorithm is represented by SSA1.
2) The SSA introduced with the genetic operator is represented by SSA2.
3) The SSA that introduces the sine and cosine search strategy is represented by SSA3.

Additionally, we obtain the characteristics of each component through experiments, and the results are as follows: 1) The time spent by SSA1 on the tested dataset is significantly shorter than that of the original SSA, but the solution accuracy is not sufficient, and the global search capability is poor. When SSA1 is used to solve the TSP examples ulysses22, eil51, berlin52, rat99 and ch130, the solution times (in seconds) are 1.32, 3.69, 10.24, 12.16 and 15.73, respectively. However, the solution accuracy is the worst, at 85.63, 460.97, 7956.23, 1398.3 and 7026.34. The optimized value is lower than that of the original SSA.

2) The solution accuracy of SSA2 is better than that of the original SSA and weaker than that of SSA3, and the solution time is significantly longer than those of the original SSA.
### TABLE 3. Results of the proposed SSA algorithm and the basic SSA algorithm.

| Instance | Optima | Best | Dev(%) | Time(s) | Best | Dev(%) | Time(s) |
|----------|--------|------|--------|---------|------|--------|---------|
| ulysses22 | 75.67  | **75.24** | 0.00 | 1.95 | 75.31 | 0.00 | 8.32 |
| att48    | 33522  | 34920.15 | 4.17 | 26.04 | 35338.34 | 5.42 | 46.37 |
| eil51    | 426    | **426**  | 0.00 | 3.87 | 441.81 | 3.71 | 9.19 |
| berlin52 | 7542   | **7542** | 0.00 | 9.48 | 7713.03 | 2.27 | 24.72 |
| st70     | 675    | 676.96  | 0.29 | 8.26 | 878.35 | 30.13 | 9.82 |
| pr76     | 108159 | 109170.3 | 0.94 | 15.13 | 137077.98 | 26.74 | 25.68 |
| eil76    | 538    | 539.79  | 0.33 | 9.32 | 584.15 | 8.58 | 23.96 |
| rat99    | 1211   | **1211** | 0.00 | 11.4 | 1298.3 | 7.21 | 15.6 |
| kroA100  | 21282  | **20989.04** | 0.00 | 19.88 | 22413.24 | 5.32 | 31.23 |
| kroB100  | 22140  | 22152  | 0.05 | 11.5 | 22629.86 | 2.21 | 40.42 |
| kroC100  | 20749  | 21346.53 | 2.88 | 11.92 | 22310.2 | 7.52 | 17.86 |
| kroD100  | 21294  | 22138.53 | 3.97 | 16.48 | 23622.62 | 10.94 | 36.24 |
| rd100    | 7910   | 7951.28 | 0.52 | 15.34 | 8721.06 | 10.25 | 41.69 |
| eil101   | 629    | 638    | 1.43 | 11.75 | 702.93 | 11.75 | 18.29 |
| lin105   | 14379  | 14543.56 | 1.14 | 15.95 | 15821.21 | 10.03 | 43.32 |
| pr124    | 59030  | 59607.74 | 0.98 | 17.68 | 64491.5 | 9.25 | 50.93 |
| ch130    | 6110.86 | 6115.25 | 0.07 | 15.59 | 6841.39 | 11.95 | 36.09 |
| pr144    | 58537  | 58862.09 | 0.56 | 20.05 | 60800.67 | 3.87 | 58.77 |
| ch150    | 6528   | 6533   | 0.08 | 13.02 | 41861.01 | 541.25 | 20.35 |
| d198     | 15780  | 15951.29 | 1.09 | 74.85 | 16339.32 | 3.54 | 106.16 |
| kroA200  | 29368  | 29507.35 | 0.47 | 49.31 | 30651.85 | 4.37 | 58.02 |
| kroB200  | 29437  | 29678.92 | 0.82 | 64.37 | 30358.6 | 3.13 | 110.66 |
| tsp225   | 3916   | **3900.93** | 0.00 | 31.23 | 4599.76 | 17.46 | 91.15 |
| pr226    | 80369  | 81361.17 | 1.23 | 35.63 | 89208.17 | 11.00 | 86.34 |
| gil262   | 2378   | 2394.98 | 0.71 | 42.76 | 2801.18 | 17.80 | 122.77 |
| a280     | 2579   | 2583.12 | 0.16 | 83.82 | 2642.12 | 2.45 | 216.72 |
| lin318   | 42090  | 43023.73 | 2.22 | 184.73 | 45687 | 8.55 | 245.31 |
| fl417    | 11861  | 11936.25 | 0.63 | 257.88 | 12821.92 | 8.10 | 350.37 |
| pr439    | 107217 | 113074.47 | 5.46 | 86.09 | 114774.9 | 7.05 | 102.31 |
| pcb442   | 50778  | 52375.32 | 3.15 | 80.04 | 53268 | 4.90 | 96.58 |
| d493     | 35002  | 36470.63 | 4.20 | 214.13 | 37568.59 | 7.33 | 437.01 |
| rat575   | 6773   | 6929.25 | 2.31 | 373.28 | 7066.02 | 4.33 | 613.31 |
| d657     | 48912  | 49204.01 | 0.60 | 445.99 | 51870.36 | 6.05 | 609.57 |
| rat783   | 8806   | 9093.27 | 3.26 | 376.62 | 10972 | 24.60 | 647.32 |
| pr1002   | 259045 | 271894.56 | 4.96 | 987.53 | 300900.41 | 16.16 | 1325.34 |
| d1291    | 50801  | 51983.28 | 2.33 | 1321.59 | 53642.59 | 5.59 | 1768.56 |

Note: Bold highlighting is only used to make the review more readable and make it easier for researchers to obtain the optimal data.

and SSA1. SSA2 has a good local search ability and easily falls into local optima. The optimal values of SSA2 on the datasets eil51 and ch130 are 428.36 and 6416.38, and the times are 12.23 and 49.63. Compared with the optimal values of the initial SSA, the values are 13.45 and 425.01 higher, and the solution times are 6.04 and 13.54 seconds longer, respectively.

3) SSA3 performs relatively well in terms of solution accuracy, significantly better than the basic SSA, SSA1 and SSA2, but it takes more time to reach a solution. SSA3 has excellent global optimization capabilities that effectively prevent the algorithm from falling into local optima. The times spent on test sets berlin52 and rat99 are 39.56 and 67.63 seconds, respectively, which are 14.84 and 12.93 seconds longer than the initial SSA solution times. In terms of solution accuracy, the deviation rate (calculated as shown in Eq.14) of SSA3 are 0.08% and 0.68%, which are significantly higher than the 2.27% and 7.21% of the original SSA.

$$Dev = \frac{BV - KV}{KV} \times 100\% \quad (14)$$

where $Dev$ represents the deviation rate, $BV$ represents the best solution value of the algorithm, and $KV$ represents the best known value.
B. COMPARISON OF THE ORIGINAL AND IMPROVED SSA

In this subsection, a comprehensive comparison between the GGSC-SSA and SSA is made to prove that the performance of the improved SSA is better than that of the original SSA. The experimental results of these two algorithms are listed in Table 3, where “optimal” represents the known optimal solution of the instance, “best” represents the average value after running the instance 50 times, and “time” represents the average time after running the instance 50 times. A total of 36 TSP instances are tested.

Based on the results shown in Table 3, an obvious conclusion can be drawn. For 36 TSP instances, the improved SSA is significantly better than the basic SSA. We conduct further analysis to prove the validity of the conclusion. Of all the TSP instances, six instances of the GGSC-SSA reach the known optimal solution, namely, on ulysses22, eil51, berlin52, rat99, kroA100 and tsp225. Note that the optimal values of ulysses22, kroA100 and tsp225 are lower than the known optimal solutions, which are 75.24, 20989.04 and 3900.93, respectively. Supporting the authenticity of the results obtained, many studies related to TSPs report...
solutions better than the known optimal solution for TSP examples. For example, in [96], using the improved ant colony algorithm to solve ulysses22, the optimal value of 75.31 found is lower than the known optimal value of 75.67; in [97], the metaheuristic hybrid algorithm is used on the examples ulysses22, att488 and berlin52, and the optimal values are 56.52, 13908.4 and 5970.83, which are all lower than the known best values; in [57], when using the improved brainstorming algorithm to solve ulysses22 and kroA100, the best values are 75.24 and 21070.09, which are both lower than the known best values. In 29 experimental results, the standard deviation is less than 3%, accounting for 75% of all examples. When the number of city nodes is less than or equal to 100, only the standard deviation of att48 is greater than 1%. When the TSP scale continues to increase, the solution performance of the GGSC-SSA is much better than that of the SSA. For the TSP instance tsp225, the best value obtained by the GGSC-SSA is 3900.93, which is 698.83 lower than the best value obtained by the SSA, and the solution time is 59.92 seconds shorter than the 91.15 seconds of the SSA. For TSP instances rat783 and pr1002, the standard deviation rates of the GGSC-SSA are 3.26% and 4.96%, respectively, which are significantly better than the SSA values of 24.60% and 16.16%.

Through the above detailed analysis, it is clear that the GGSC-SSA is superior to the original SSA in terms of solution accuracy, solution time, and stability. The six examples of ulysses22, eil51, berlin52, rat99, kroA100 and tsp225 are...
C. EXPERIMENTATION WITH THE GGSC-SSA AND CLASSICAL INTELLIGENT ALGORITHM

To prove the efficiency of the GGSC-SSA, we compare the improved algorithm with six classic heuristic algorithms, GA, SA, PSO, GWO, ACO and AF, on the ulysses22, eil51 and berlin52 datasets. To ensure the validity and fairness of the experiment, for the same TSP instance, all algorithms are tested under the same hardware environment. Figure 4 shows the optimization process of the eight algorithms used to test the TSP instances ulysses22, eil51 and berlin52.

The information in Figure 4 clearly shows that for the tested TSP examples, the solution accuracy of the GGSC-SSA and the optimal number of iterations are better than those of the other seven algorithms. Note that GGSC-SSA can find a better value in the first iteration. The greedy algorithm is used for the population initialization of the SSA, which greatly enhances the optimization ability of the SSA. To further illustrate the advantages of the GGSC-SSA in terms of the optimization efficiency and solution time, the optimal value found by each algorithm and the specific time spent are introduced in detail. The experimental results are shown in Table 4. In this table, Dev, MeanV and MeanT represent the standard deviation rate, average running time and average running time, respectively. Taking the eil51 dataset as an example, the GGSC-SSA reaches the known optimal value, exhibiting great advantages compared with other algorithms. In particular, compared with GA, PSO and GWO, the deviation rate is 97.16%, 78.53% and 94.55% lower, respectively. Note that the SA algorithm performs better in terms of solution time, but as the TSP scale increases, the solution accuracy continues to decrease. Figure 5 shows the detailed solution time of the eight algorithms used to solve the three TSP instances. When solving small-scale TSPs, GGSC-SSA is better than the other algorithms in terms of optimizing speed, deviation rate and stability. Comparative experiments with traditional heuristic algorithms further show that the improved algorithm has better optimization capabilities and greater robustness.

D. COMPARISON WITH OTHER IMPROVED ALGORITHMS

To further comprehensively verify the efficiency of the GGSC-SSA, it is compared with a series of recently improved intelligent optimization methods. The nine improved algorithms participating in the comparison are as follows: (1) the novel discrete water cycle algorithm (DWCA) [90]; (2) the improved ant colony optimization (IACO) [98]; (3) the agglomerative greedy brain storm optimization (AG-BSO) [57]; (4) the parallel ant colony optimization and 3-opt (PACO-3OPT) algorithm [99]; (5) the genetic ant

### TABLE 4. Experimental results of the SSA algorithm and other classic heuristic algorithms.

| Algorithm | Index | ulysses22 | eil51 | berlin52 |
|-----------|-------|----------|-------|---------|
| GGSC-SSA  | MeanV | 75.24    | 426   | 7542    |
|           | MeanT(s) | 1.95   | 3.87  | 9.48    |
|           | Dev(%)  | 0       | 0     | 0       |
| SSA       | MeanV | 75.31    | 441.81| 7713.03 |
|           | MeanT(s) | 8.32   | 9.19  | 24.72   |
|           | Dev(%)  | 0       | 3.71  | 2.27    |
| GA        | MeanV | 86.61    | 839.9 | 14876.25 |
|           | MeanT(s) | 2.38   | 5.37  | 6.99    |
|           | Dev(%)  | 14.46   | 97.16 | 97.25   |
| SA        | MeanV | 75.31    | 439.74| 7750.17 |
|           | MeanT(s) | 0.31   | 0.5   | 0.71    |
|           | Dev(%)  | 0       | 3.23  | 2.76    |
| PSO       | MeanV | 76.61    | 759.75| 12349.3 |
|           | MeanT(s) | 5.59   | 12.59 | 13.61   |
|           | Dev(%)  | 1.24    | 78.35 | 63.74   |
| GWO       | MeanV | 81.17    | 828.8 | 15281.43 |
|           | MeanT(s) | 3.07   | 6.46  | 6.95    |
|           | Dev(%)  | 7.27    | 94.55 | 102.62  |
| ACO       | MeanV | 75.98    | 446.89| 7663.59 |
|           | MeanT(s) | 13.38  | 37.23 | 37.91   |
|           | Dev(%)  | 0.41    | 4.9   | 6.61    |
| AF        | MeanV | 76.11    | 594.4 | 10307.82 |
|           | MeanT(s) | 16.21  | 18.84 | 19.37   |
|           | Dev(%)  | 0.58    | 39.53 | 36.67   |
TABLE 5. Statistical results of GGSC-SSA and nine other algorithms used to solve TSP instances.

| Algorithm | Instance | Optimal | eil51 | berlin52 | st70 | eil76 | kroA100 | eil101 | ch130 | pr226 | pr439 |
|-----------|----------|---------|-------|----------|------|-------|---------|-------|-------|-------|-------|
| GGSC-SSA  | Best     | 426     | 7542  | 676.96   | 539.79 | 20989.04 | 638     | 6115.25 | 80461.17 | 13074.5 |
|           | Dev(%)   | 0       | 0     | 0.29     | 0.33   | 0.143 | 0.07    | 1.23   | 5.46   |
| DWCA      | Best     | 426     | 7542  | 678.6    | 543   | 21282  | 639     | 6217.21 | -      | -     |
|           | Dev(%)   | 0       | 0     | 0.53     | 0.93   | 0     | 1.59    | 1.74   | -      | -     |
| IACO      | Best     | 426     | 7542  | 676     | 538   | 21308  | 631     | -       | -      | -     |
|           | Dev(%)   | 0       | 0     | 0.15     | 0      | 0.12   | 0.32    | -      | -      | -     |
| AG-BSO    | Best     | 428.58  | 7542  | 678     | 540.69 | 21070.09 | 633     | 6125.25 | 80961.17 | -     |
|           | Dev(%)   | 0.69    | 0     | 0.44     | 0.5    | 0      | 0.64    | 0.23   | 0.74   | -     |
| PACO-3OPT | Best     | 426.85  | 7542.75 | 676.8      | 538.45 | -     | 632.95  | -      | -      | -     |
|           | Dev(%)   | 0.2     | 0.01  | 0.27     | 0.08   | -      | 0.63    | -      | -      | -     |
| GACO      | Best     | 429.36  | 8076.23 | 723.25    | 568    | 21482.21 | -       | -      | -      | -     |
|           | Dev(%)   | 0.79    | 7.08   | 7.15     | 5.58   | 0.95   | -       | -      | -      | -     |
| ABSO      | Best     | 436.26  | -      | 694.28   | -      | 22023  | -       | -      | 80369  | -     |
|           | Dev(%)   | 0.24    | -      | 2.8      | -      | 3.4    | -       | -      | 6.9    | -     |
| PCCACO    | Best     | 426     | 7542  | -       | 538    | 21651  | 637     | 6129   | -      | -     |
|           | Dev(%)   | 0       | 0     | -       | 0      | 1.73   | 1.27    | 0.3    | -      | -     |
| NACO      | Best     | 426     | -      | -       | -      | 21282  | -       | -      | 80453  | 117878 |
|           | Dev(%)   | 0       | -     | -       | -      | 0      | -       | -      | 0.1    | 9.94  |
| MDBSO     | Best     | 436.4   | -      | 685.5    | -      | 21339  | -       | -      | 81606  | -     |
|           | Dev(%)   | 2.44    | -     | 1.5      | -      | 0.26   | -       | -      | 1.53   | -     |

Note: ‘-’ means that the algorithm is not running on the relevant TSP data set.

colony optimization (GACO) [91]; (6) the adaptive brain storm optimization (ABSO) [100]; (7) the Pearson correlation coefficient ant colony optimization (PCCACO) [91]; (8) the novel ant colony optimization (NACO) [45]; (9) the multi-strategy discrete brain storm optimization (MDBSO) [101].

The statistical results of the GGSC-SSA and other improved algorithms are shown in Table 5, and ‘-’ indicates that the method is not tested in its article. The conclusions drawn from Table 5 are similar to the previous conclusions. In the small TSP examples eil51, berlin52, st70, eil76 and kroA100, good results are obtained. In addition, for the large-scale TSP instance pr439, the performance of the GGSC-SSA proposed in this paper is significantly better than that of the other improved algorithms. In contrast, the GGSC-SSA maintains good adaptability to all TSP instances. In summary, the experimental data show that this method has strong competitiveness. As the complexity of the problem increases, the GGSC-SSA can jump faster out of local optimal solutions, thereby improving the global optimization capability. This effect is mainly derived from the greedy algorithm to obtain a solution close to the global optimal value. The sine-cosine search strategy enhances the global optimization capability, thereby improving the convergence of the algorithm. The GGSC-SSA proposed in this paper can obtain a better and more stable solution when solving TSPs, which is more obvious in large-scale examples.

VI. CONCLUSION

To address the issues of the SSA having insufficient convergence ability and efficiency in solving the TSP, an improved SSA named the GGSC-SSA is introduced in this paper. The three key improvements of the GGSC-SSA are as follows:

1) The greedy algorithm is introduced into the SSA to initialize the population to enhance the solving efficiency of the algorithm.
2) The crossover operation of the GA is used to update the population to enhance the global search ability. The mutation operation is used to update producers and to enhance the local search ability of the algorithm.
3) Sine and cosine search strategies are used through early warning sparrows to prevent premature convergence and to enhance the global optimization ability of the algorithm.

On the basis of a thorough and comprehensive theoretical study of the original SSA and the TSP, a novel SSA (GGSC-SSA) is introduced for the first time to solve the TSP in this research. To demonstrate that the proposed GGSC-SSA is an effective algorithm for solving the TSP, we compare its performance with the basic SSA on 36 TSP instances. Furthermore, we analyze the GGSC-SSA in detail through comparative experiments with six classical algorithms and eight existing improved algorithms. The simulation results validate the effectiveness of the proposed algorithm. The GGSC-SSA shows excellent performance in solving TSP cases on large and small scales and is better than other improved algorithms in most cases. In future work, other intelligent algorithms will be introduced into the SSA to explore the new intergroup communication model and to improve the robustness and adaptability of the SSA.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and Academic Editor for their valuable and constructive comments, which greatly improved the quality and integrity of this manuscript.
REFERENCES

[1] M. Clerc and I. Kennedy, “The particle swarm—Explosion, stability, and convergence in a multidimensional complex space,” IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 58–73, Aug. 2002.

[2] X. Yang, “Firefly algorithms for multimodal optimization,” in Proc. Int. Conf. Stochastic Algorithms Found. Appl., 2009, pp. 169–178.

[3] M. Dorigo, V. Maniezzo, and A. Colomi, “Ant system: Optimization by a colony of cooperating agents,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 29–41, Feb. 1996.

[4] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. D. Socliho, “Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization,” Expert Syst. Appl., vol. 47, pp. 106–119, Apr. 2016.

[5] S. Shadravan, H. R. Naji, and V. K. Bardsiri, “The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems,” Eng. Appl. Artif. Intell., vol. 80, pp. 20–34, Apr. 2019.

[6] J. Xue and B. Shen, “A novel swarm intelligence optimization approach: Sparrow search algorithm,” Syst. Sci. Control Eng., vol. 8, no. 1, pp. 22–34, Jan. 2020.

[7] Y. Shi, “Brain storm optimization algorithm,” in Proc. Int. Conf. Swarm Intel., 2011, pp. 303–309.

[8] S. Mirjalili and A. Lewis, “The whale optimization algorithm,” Adv. Eng. Softw., vol. 95, pp. 51–67, May 2016.

[9] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, “Hybrid clustering analysis using improved krill herd algorithm,” Appl. Intell., vol. 48, no. 11, pp. 4047–4071, 2018.

[10] J. Zhang, K. Xia, Z. He, Z. Yin, and S. Wang, “Semi-supervised ensemble learning for multi-class texture classification,” Expert Syst. Appl., vol. 49, no. 1, pp. 2201–2217, 2019.

[11] S. Kumaravel and V. Ponnusamy, “An efficient hybrid technique for power flow management in smart grid with renewable energy resources,” Energy Sour. A, Recovery, Utilization, Environ. Effects, vol. 9, pp. 16623–16629, 2021.

[12] B. Liu and D. Rodriguez, “A comparative study of improved GA and PSO in solving multiple traveling salesmen problem,” Soft Comput., vol. 25, no. 5, pp. 3775–3794, Mar. 2021.

[13] Y. Saji and M. Barkatou, “A discrete bat algorithm based on Lévy flights for Euclidean traveling salesman problem,” Soft Comput., vol. 95, pp. 51–67, May 2016.

[14] J. Lin, J. K. Teer, P. Cliften, E. Tycksen, and Y. Samuels, “Mining exomic sequencing data to identify mutated antigens recognized by adoptively transferred tumor-reactive T cells,” Nature Med., vol. 19, no. 6, p. 747, 2013.

[15] J. Chen and Z. Chen, “Discrete pigeon-inspired optimization algorithm with metropolis acceptance criterion for large-scale traveling salesman problem,” Swarm Evol. Comput., vol. 48, pp. 134–144, Aug. 2019.

[16] J. Zhao, S. Wang, Y. Hao, and Y. Yang, “Energy-dependent mission planning for agile earth observation satellite,” J. Aerosp. Eng., vol. 32, no. 1, Jan. 2019, Art. no. 04018118.

[17] P. F. Robbins, Y. C. Lu, M. El-Gamal, Y. F. Li, C. Gross, J. Gartner, J. C. Lin, J. K. Teer, P. Cliften, E. Tycksen, and Y. Samuels, “Mining exomic sequencing data to identify mutated antigens recognized by adoptively transferred tumor-reactive T cells,” Nature Med., vol. 19, no. 6, p. 747, 2013.
S. A. Vasquez, G. Angulo, and M. A. Klapp, “An exact solution method for the travelling salesman problem,” Int. J. Comput. Math., vol. 50, no. 11, pp. 3942–3960, Nov. 2020.

S. Verma, J. J. Jena, S. C. Satapathy, and M. Rout, “Solving travelling salesman problem using discrete social group organization,” J. Sci. Ind. Res., vol. 79, no. 10, pp. 928–930, 2020.

T. S. Pham, P. Leyman, and P. De Causmaecker, “The intermittent travelling salesman problem,” Int. Trans. Oper. Res., vol. 27, no. 1, pp. 525–548, Jan. 2020.

A. F. Tuani, E. Keedwell, and M. Collett, “Heterogeneous adaptive ant colony optimization with 3-opt local search for the travelling salesman problem,” Appl. Soft Comput., vol. 97, Dec. 2020, Art. no. 106720.

D.-H. Tran, T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, “Coarse trajectory design for energy minimization in UAV-enabled,” IEEE Trans. Veh. Technol., vol. 69, no. 9, pp. 9483–9486, Sep. 2020.

B. C. H. Silva, I. F. C. Fernandes, M. C. Goldsarb, and E. F. G. Goldsarb, “Quota travelling salesman problem with passengers, incomplete ride and collection time optimization by ant-based algorithms,” Comput. Oper. Res., vol. 120, Aug. 2020, Art. no. 104950.

K. Saito, M. Aono, and S. Kasai, “Amoeba-inspired analog electronic computing system integrating resistance crossbar for solving the travelling salesman problem,” Sci. Rep., vol. 10, no. 1, pp. 1–9, Dec. 2020.

D. C. Popescu and P. Kilby, “Approximation of the Shapley value for the Euclidean travelling salesman game,” Ann. Oper. Res., vol. 289, no. 2, pp. 341–362, Jun. 2020.

M. Cavaleri and A. Donno, “Distance-balanced graphs and travelling salesman problems,” Ars Mathematica Contemporanea, vol. 19, no. 2, pp. 311–324, Nov. 2020.

F. Carrabs, C. Cerone, R. Cerulli, and B. Golden, “An adaptive heuristic approach to compute upper and lower bounds for the close-enough travelling salesman problem,” Inform. J. Comput., vol. 32, no. 4, pp. 1030–1048, Sep. 2020.

V. Cacciani, C. Conterras-Bolton, and P. Toth, “Models and algorithms for the travelling salesman problem with time-dependent service times,” Eur. J. Oper. Res., vol. 283, no. 3, pp. 825–843, Jun. 2020.

C. Wu and X. Fu, “An agglomerative greedy brain storm optimization algorithm for solving the TSP,” IEEE Access, vol. 8, pp. 201606–201621, 2020.

M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and V. Cacchiani, C. Contreras-Bolton, and P. Toth, “Models and algorithms for the travelling salesman problem with time-dependent service times,” Eur. J. Oper. Res., vol. 283, no. 3, pp. 825–843, Jun. 2020.

C. Wu and X. Fu, “An agglomerative greedy brain storm optimization algorithm for solving the TSP,” IEEE Access, vol. 8, pp. 201606–201621, 2020.

M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and M. J. Koushyar, “An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems,” J. Cloud Comput., vol. 9, no. 1, pp. 1–14, Dec. 2020.

J. Zhang, L. Hong, and Q. Liu, “An improved whale optimization algorithm for the travelling salesman problem,” Symmetry, vol. 13, no. 1, p. 48, Dec. 2021.

I. Želinka and S. Das, “Gamesourcing: An unconventional tool to assist the solution of the travelling salesman problem,” Natural Comput., vol. 2, pp. 1–11, Nov. 2020.

M. Yousefikhoshbandkh, “Solving the travelling salesman problem: A modified metaheuristic algorithm,” Complexity, vol. 2021, pp. 1–13, Feb. 2021.

Z. Wu, J. Wu, M. Zhao, L. Feng, and K. Liu, “Two-layered ant colony system to improve engraving robot’s efficiency based on a large-scale TSP model,” Neural Comput. Appl., vol. 2020, pp. 1–11, Jun. 2021.

H. Liu, P. Zhang, B. Hu, and P. Moore, “A novel approach to task assignment in a cooperative multi-agent design system,” Appl. Intell., vol. 43, no. 1, pp. 162–175, 2015.

M. C. de Oliveira, M. R. Delgado, and A. Britto, “A hybrid greedy indicator-and Pareto-based many-objective evolutionary algorithm,” Appl. Intell., vol. 1–23, Jan. 2021.

M. S. Kıran, H. İşcan, and M. Gündüz, “The analysis of discrete artificial bee colony algorithm with 3-opt local search for the travelling salesman problem,” Comput. Oper. Res., vol. 127, pp. 105–127, Mar. 2021.
[91] H. Zhu, X. You, and S. Liu, “Multiple ant colony optimization based on Pearson correlation coefficient.” *IEEE Access*, vol. 7, pp. 61628–61638, 2019.

[92] T.-T. Nguyen, Y. Qiao, J.-S. Pan, S.-C. Chu, K.-C. Chang, X. Xue, and T.-K. Dao, “A hybridized parallel bats algorithm for combinatorial problem of traveling salesman,” *J. Intell. Fuzzy Syst.*, vol. 38, no. 5, pp. 5811–5820, May 2020.

[93] Y. Saji and M. Barkatou, “A discrete bat algorithm based on Lévy flights for Euclidean traveling salesman problem,” *Expert Syst. Appl.*, vol. 172, Jun. 2021, Art. no. 114639.

[94] P. Wang, Y. Zhang, and H. Yang, “Research on economic optimization of microgrid cluster based on chaos sparrow search algorithm,” *Comput. Intell. Neurosci.*, vol. 2021, no. 3, pp. 1–18, Mar. 2021.

[95] Y. Wu, Z. Zhang, R. Xiao, P. Jiang, Z. Dong, and J. Deng, “Operation state identification method for converter transformers based on vibration detection technology and deep belief network optimization algorithm,” *Actuators*, vol. 10, no. 3, p. 56, Mar. 2021.

[96] D. Pengzhen, T. Zhenmin, and S. Yan, “An object-oriented multi-role ant colony optimization algorithm for solving TSP problem,” *Control Decis.*, vol. 29, no. 10, pp. 1729–1736, 2014.

[97] U. Ashraf, J. Liang, and A. Akhtar, “Meta-heuristic hybrid algorithmic approach for solving combinatorial optimization problem,” in *Proc. Int. Conf. Bio-Inspired Comput.*, 2020, pp. 645–656.

[98] D. Qiao, P. Jie, L. Hao, and W. Xiaoyu, “Research on improving ant colony algorithm to solve TSP problem,” *Machinery Design Manuf.*, no. 10, pp. 144–149, 2019.

[99] G. Şaban, M. Mostafa, B. O. Kaan, and H. Kodaz, “A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem,” *Soft Comput.*, vol. 22, no. 5, pp. 1669–1685, Mar. 2018.

[100] Y. Xu, Y. Wu, Y. Fu, X. Wang, and A. Lu, “Discrete brain storm optimization algorithm based on prior knowledge for traveling salesman problems,” in *Proc. 13rd IEEE Ind. Electron. Appl. (ICIEA)*, May 2018, pp. 2740–2745.

[101] Y. Wu, X. Wang, J. Qi, and L. Huang, “An adaptive brain storm optimization algorithm based on heuristic operators for TSP,” in *Proc. Int. Conf. Bio-Inspired Comput.*, Singapore: Springer, 2019, pp. 662–672.

**CHANGYOU WU** received the B.E. and M.E. degrees in industrial engineering and management science engineering and the Ph.D. degree in agricultural system engineering and management from Northeast Agricultural University, China, in 2004, 2008, and 2018, respectively.

He is currently an Associate Professor with the Liaoning Institute of Science and Technology. He published over 20 articles in domestic and international academic journals and conference proceedings. His current research interests include genetic algorithm theory and its application, neural network theory and its application, parallel computing, and water resources management.

**XISONG FU** received the B.E. degree in information management and information system from the LiaoCheng University, China, in 2020. He is currently pursuing the master’s degree with the Engineering College, Shandong Institute of Business and Technology.

Her current research interest includes intelligent optimization algorithm and its application.

**JUNKE PEI** received the B.E. degree in information management and information system from LiaoCheng University, China, in 2020. She is currently pursuing the master’s degree with the Engineering College, Shandong Institute of Business and Technology.

Her current research interest includes intelligent optimization algorithm and its application.

**ZHIGUI DONG** received the B.E. and M.E. degrees in industrial engineering and management science engineering and the Ph.D. degree in agricultural system engineering and management engineering from Northeast Agricultural University, China, in 2005, 2008, and 2018, respectively.

He is currently an Associate Professor with the Liaoning Institute of Science and Technology. He published over 20 articles in domestic and international academic journals and conference proceedings. His current research interests include neural network theory and its application, optimization algorithm theory and its application, big data technology and application, and agricultural big data.

* * *