Greedy Squirrel Search Algorithm for Large-Scale Traveling Salesman Problems

Chenghao Shi[1], Zhonghua Tang[1,2,3], Yongquan Zhou[2,3](✉), and Qifang Luo[2,3]

[1] College of Electronic Information, Guangxi University for Nationalities, Nanning 530006, China

[2] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China
yongquanzhou@126.com

[3] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China

Abstract. A greedy strategy based on squirrel search algorithm is proposed and used to solve the TSP problem. First, a new path initialization operator is designed for partial path initialization. Secondly, two search operators and a mutation strategy are designed to improve the convergence speed and performance of the algorithm: (1) Greedy crossover operator, which is used to improve the information interaction ability between each individual. (2) RC2opt+R3opt operator, which is used to enhance the local search ability of individuals. (3) Local mutation strategy, which uses seasonal constants to measure population differences and the number of iterations, and performs mutation operations on individuals in the middle and late stages of the algorithm to prevent the algorithm from falling into the local optimum. Finally, a large number of TSP examples in the open dataset TSPLIB are selected to verify the effectiveness of the algorithms and operators. The comparison results with 3 classical algorithms and 7 latest algorithms show that the algorithm has higher precision and better stability.

Keywords: Greedy squirrel search algorithm · Greedy crossover operator · R2Copt+R3opt operator · Traveling salesman problem

1 Introduction

Traveling salesman problem (TSP) [1] is one of the most representative combinatorial optimization problems, and it has a wide range of applications in computer science, operations research and other fields. Suppose there is a traveling merchant who wants to visit n cities, he must choose the path he wants to take, and the restriction is that each city can only be visited once, and he must return to the started city in the end. The path selection goal is to require the path distance to be the minimum value among all paths. Some engineering problems, such as printed circuit board design (PCB design) [2], VRP vehicle routing problem (vehicle routing problem) [3–5], robot path planning [6, 7], telecommunication network [8], can be attributed to in TSP. TSP is a typical

D.-S. Huang et al. (Eds.): ICIC 2022, LNAI 13395, pp. 830–845, 2022.
https://doi.org/10.1007/978-3-031-13832-4_67

NP-hard problem [9, 10]. As the scale of the problem increases, the scale of its feasible solution set increases exponentially, which inevitably leads to the phenomenon of "combinatorial explosion". The classical exact algorithm cannot obtain the optimal solution in polynomial time.

Recent years, Swarm intelligence algorithms are widely used in various fields. Which is a branch of artificial intelligence algorithm (AI). Scholars are inspired by movements from animals, insects and organisms. Over the past few decades, the information interaction behavior of various social insects such as bees, wasps, termites, birds, flies, and fish has been studied, resulting in the creation of various meta-heuristic algorithms [11–13]. The most important factor in their success is their simplicity and versatility. At present, the SI algorithm has been widely used to solve NP-hard problems, in which it is almost impossible for the classical exact algorithm to find the global optimal solution in limited time. In this case, how to find a feasible solution within a reasonable time limit becomes important. Therefore, it is of great significance to use the swarm intelligence optimization algorithm to study and solve the TSP problem.

2 The Basic Squirrel Search Algorithms

The squirrel search algorithm [14] is a swarm intelligence algorithm, which is proposed by Indian scholar Mohit Jain in 2019. It simulates the dynamic foraging strategies of southern flying squirrels and their efficient locomotion, which is gliding. Considered the most aerodynamically complex species, the flying squirrel has a membrane which is simulate to parachute that provides the lifting and dragging forces for squirrel to glide from one tree to other tree. Flying squirrels cannot fly, but rather glide quickly and efficiently across long distances. The reason why flying squirrels glide is to avoid predators, and find the best place to hunt with a minimum cost. There are three conditions that squirrels may experience during dynamic foraging. In each case, Assuming that in the absence of predators, the squirrel would glide through the forest and efficiently search for its preferred food, while the predator's pursuit makes them searching food cautiously and forced them moving in smaller areas. Walk randomly inside to find nearby hidden locations.

2.1 Foraging Behavior

The mathematical model of dynamic foraging behavior is as follows:

Case 1: Acron nut tree → Hickory tree

Flying squirrels which are on acorn nut trees (FS_{at}) may move towards hickory nut tree. In this case, the new location of squirrels can be obtained as follows:

$$FS_{at}^{t+1} = \begin{cases} FS_{at}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{at}^{t}) & R_1 \geq P_{dp} \\ \text{Random Location} & \text{otherwise} \end{cases} \tag{1}$$

where d_g is random gliding distance, R_1 is a random number in the range of [0, 1], $FS_{\mathrm{h}t}$ is the location of flying squirrel that reached hickory nut tree and t denotes the current iteration. The balance between exploration and exploitation is achieved with the

help of gliding constant G_c in the mathematical model. Its value significantly affects the performance of proposed algorithm. In the present work value of G_c is considered as 1.9, which is obtained after rigorous analysis.

Case 2: Normal tree → Acron nut tree

Flying squirrels on normal trees (FS_{nt}) may move towards acorn nut trees to fulfill their daily energy needs. In this case, new location of squirrels can be obtained as follows:

$$FS_{nt}^{t+1} = \begin{cases} FS_{nt}^{t} + d_g \times G_c \times (FS_{at}^{t} - FS_{nt}^{t}) & R_2 \geq P_{dp} \\ \text{Random Location} & \text{otherwise} \end{cases} \tag{2}$$

where R_2 is a random number in the range [0, 1].

Case3: Normal tree → Hickory tree

Some squirrels which are on normal trees and already consumed acorn nuts may move towards hickory nut tree in order to store hickory nuts which can be consumed at the time of food scarcity. In this case, new location of squirrels can be obtained as follows:

$$FS_{nt}^{t+1} = \begin{cases} FS_{nt}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{nt}^{t}) & R_3 \geq P_{dp} \\ \text{Random Location} & \text{otherwise} \end{cases} \tag{3}$$

where R_3 is a random number in the range [0, 1]. Predator presence probability P_{dp} is considered to be 0.1 in all cases for the present work.

3 Greedy Squirrels Search Algorithm for TSP

This section will introduce how to solve the TSP problem with a greedy squirrels search algorithm (GSSA), which includes one route initialization operator, two route update operators and one individual mutation strategy.

3.1 Description of TSP

Traveling Salesman Problem can be described: There are n cities, in the case of only one salesman, how does the salesman visit all the cities and return to the departure city. And each city can be visited only once, the goal of the TSP problem is to find a shortest route back to the departure city. According to the graph theory, the Traveling Salesman Problem can be described as a graph G = (V, E), where V is the set of vertices, E is the set of edges, and the optimization goal is to find the shortest Hamiltonian loop. Each vertex represents a city. Each edge represents a path between two cities. The mathematical expression of the TSP model is:

Define: $xij, \forall i, j \in \{1, 2, ..., n\}$ & $j \neq i$

$$\mathrm{Z} = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} d_{ij} x_{ij} \tag{4}$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1, i \in \{1, 2, \ldots, n\}. \tag{5}$$

$$\sum_{i=1,i\neq j}^{n} x_{ij} = 1, j \in \{1, 2, \ldots, n\}. \tag{6}$$

$$\sum_{i\in S} \sum_{j\in S, j\neq i} xij \leq |S| - 1, S \subset V. \tag{7}$$

When the salesman travels from the *No.i* city to the *No.j* city $x_{ij} = 1$, otherwise $x_{ij} = 0$. Equation (4) is the route length, and d_{ij} is the distance from the *No.i* city to the *No.j* city. Equation (5) and Eq. (6) ensure that each city can be visited only once. Equation (7) guarantees that there is only one closed loop in each solution, excluding the case of two or more closed loops in a solution, which is not feasible although the constraints of the equation are satisfied. In the squirrel search algorithm, the natural number coding mechanism is adopted. $X = \{x_1, x_2, \ldots, x_n\}$ is the sequence of cities which salesman passes through, where $x_i \in V$. x_i are unique to each others, and the Traveling Salesman Problem can be described as:

Define the best individual: X_{best}:

$$X_{best} = \{x_1, x_2, \ldots, x_n\}, x_i \in V \tag{8}$$

The minimum distance: Z_{min}:

$$Z_{min} = \sum_{k=1}^{n-1} d_{x_k x_{k+1}} \tag{9}$$

3.2 The Framework of GSSA

The framework of GSSA for TSP can be described as Algorithm 1. It can be seen from the flow chart that compared with SSA, GSSA is different from four parts: (1) Elite Individual Greedy Initialization (2) Greedy Crossover operator (3) 2opt&3opt operator (4) Local mutation strategy. The positions of added four parts in SSA are shown in Fig. 1. The variable names and their corresponding explanations are described in Table 1.

First, define the population size as $\boldsymbol{N}$. Setting the maximum number of iterations $\boldsymbol{Max_{iter}}$, and determine the number of elite individuals $\boldsymbol{N_E}$ at the initialization stage of the algorithm. As pseudocode shown in Algorithm1 lines 2–3. Then, running the initialization operator for the entire population. First, using greedy initialization operator to calculate the route of elite individuals $\boldsymbol{N_E}$, and using random initialization operator to calculate the route of rest normal individuals $\boldsymbol{N - N_E}$. Sorting individuals according to the fitness values, Select the best solution $\boldsymbol{Fa_{path}}$ and the second optimal solution $\boldsymbol{Fh_{path}}$ from the initialized population. Then, starting iteration. Using the greedy crossover operator and 2opt [15] & 3opt operator to optimize the population. The details are described in Sect. 3.4. During this period, if a better solution was founded, replace the local optimal solution $\boldsymbol{Lbest_{path}}$. In order to prevent the algorithm from falling into local optimum at the end of stages, a local mutation strategy was designed to jump out of the local optimum. Details in Sect. 3.5. Finally, output the global best solution $\boldsymbol{Gbest_{path}}$. According to the above description and analysis, the steps of the GSSA are summarized as follows:

Table 1. Variable names and its meanings of GSSA

Names	Meanings
D	City distance martix
n	Number of city
Max_{iter}	Maximum iteration
N	Population size
N_E	Number of elite individuals
$Gbest_{path}$	Global best solution
$Lbest_{path}$	Local best solution
t	Present iteration
i	Present individual
Fa_{path}	Best solution
Fh_{path}	Secondary solutions
X_i	Normal solutions

Algorithm1 Pseudocode of GSSA for TSP

import TSP coordinates, calculate the city distance matrix $\boldsymbol{D}$

set up: Max iteration $\boldsymbol{Max_{iter}}$ Population size N, Number of elite individuals $\boldsymbol{N_E}$, initialize $\boldsymbol{Gbest_{path}}$, $\boldsymbol{Lbest_{path}}$ Geedy initialize elite individuals $\boldsymbol{N_E}$, Random initialize normal individuals $\boldsymbol{N - N_E}$, calculate fitness value and find out $\boldsymbol{Fa_{path}}$, $\boldsymbol{Fh_{path}}$. Calculate $\boldsymbol{P_{dp}}$, start iteration.

for t=1 to $\boldsymbol{Max_{iter}}$ do

  calculate fitness value and find out $\boldsymbol{Fa_{path}}$, $\boldsymbol{Fh_{path}}$.

  for i=1 to N do

    if ($\boldsymbol{r_1 \geq P_{dp}}$)

      $\boldsymbol{Fa_{path} \leftarrow GreedyCross(Lbest_{path}, Fa_{path})}$

    else

      $\boldsymbol{Fa_{path} \leftarrow 2opt\&3opt(Fa_{path})}$

    Calculate $\boldsymbol{Fa_{path}}$ fitness value, sorting and find out $\boldsymbol{Fa_{path}}$、$\boldsymbol{Fh_{path}}$、$\boldsymbol{Lbest_{path}}$

    if fit($\boldsymbol{Fa_{path}}$)<fit($\boldsymbol{Lbest_{path}}$)

      $\boldsymbol{Lbest_{path} \leftarrow Fa_{path}}$

    if ($\boldsymbol{r_2 > P_{dp}}$)

      $\boldsymbol{X_i \leftarrow GreedyCross(Lbest_{path}, X_i)}$

    else

      $\boldsymbol{X_i \leftarrow 2opt\&3opt(X_i)}$

    Calculate $\boldsymbol{X_i}$ fitness value, sorting and find out $\boldsymbol{Fa_{path}}$、$\boldsymbol{Fh_{path}}$、$\boldsymbol{Lbest_{path}}$

    if fit($\boldsymbol{Fa_{path}}$)<fit($\boldsymbol{Lbest_{path}}$)

      $\boldsymbol{Lbest_{path} \leftarrow Fa_{path}}$

    if ($\boldsymbol{r_3 > P_{dp}}$)

      $\boldsymbol{X_i \leftarrow GreedyCross(Lbest_{path}, Fh_{path}))}$

    else

      $\boldsymbol{X_i \leftarrow 2opt\&3opt(X_i)}$

    Calculate $\boldsymbol{X_i}$ fitness value, sorting and find out $\boldsymbol{Fa_{path}}$、$\boldsymbol{Fh_{path}}$、$\boldsymbol{Lbest_{path}}$

    if fit($\boldsymbol{Fa_{path}}$)<fit($\boldsymbol{Lbest_{path}}$)

      $\boldsymbol{Lbest_{path} \leftarrow Fa_{path}}$

    Calculate season constants $\boldsymbol{S_c}$、$\boldsymbol{S_{min}}$

    if ($\boldsymbol{S_c < S_{min}}$)

      Random generating new individual $\boldsymbol{X_{Temp}}$

      $\boldsymbol{X_i \leftarrow E2opt\&3opt(GreedyCross(Lbest_{path}, X_{Temp}))}$

      Calculate $\boldsymbol{X_i}$ fitness value, sorting and find out $\boldsymbol{Fa_{path}}$、$\boldsymbol{Fh_{path}}$、$\boldsymbol{Lbest_{path}}$

      if fit($\boldsymbol{Fa_{path}}$)<fit($\boldsymbol{Lbest_{path}}$)

        $\boldsymbol{Lbest_{path} \leftarrow Fa_{path}}$

output global best solution $\boldsymbol{Gbest_{path}}$

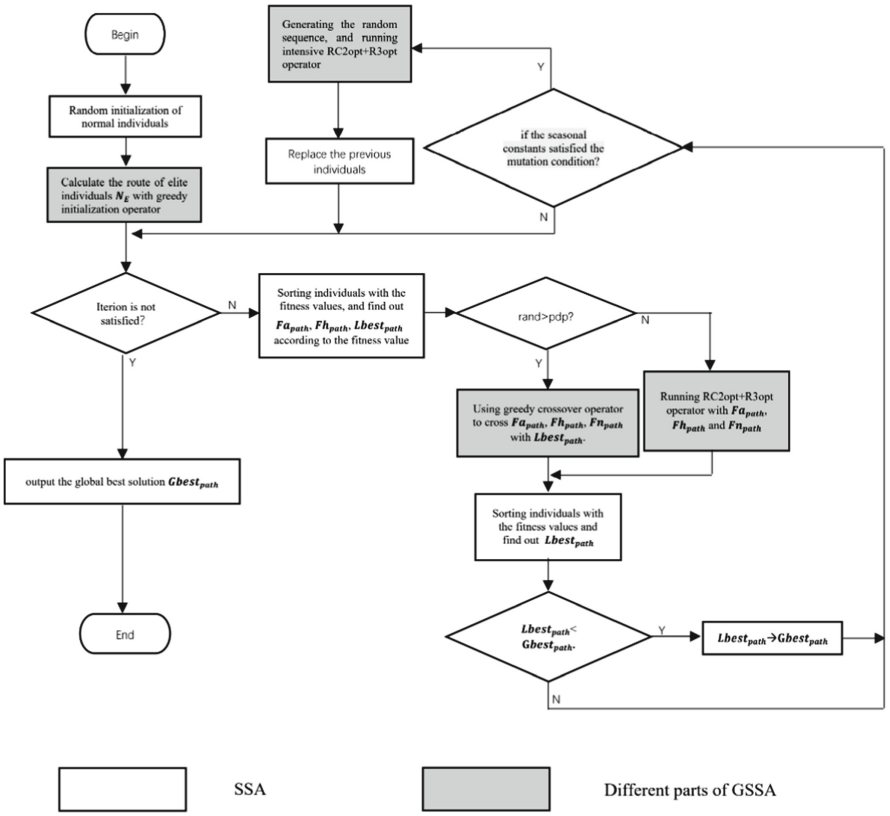

Fig. 1. Flow chart of GSSA

3.3 Greedy Initialize

The idea of the elite individual greedy initialization operator can be described as: First, initialize two arrays $\boldsymbol{S}$, $\boldsymbol{US}$. Where $\boldsymbol{S}$ is an empty array, which used to store the city serial number that has been selected. $\boldsymbol{US}$ is an array which filling with natural number from 1 to n. It used to store the unselected city serial number. Selecting $\boldsymbol{P_{start}}$($\boldsymbol{P_{start}} = 1, 2, 3, \ldots, n$) as the start point of the route. And save it into the S array, and then find the nearest next city according to the greedy strategy until found a completed route. For instance, There are currently 5 cities, and the corresponding city numbers are 1 to 5. Where $\boldsymbol{S} = \{\}$, $\boldsymbol{US} = \{1, 2, 3, 4, 5\}$ according to the initialization phase. Selecting the city with serial number 1 as the starting point ($\boldsymbol{P_{start}} = 1$). Saved $\boldsymbol{P_{start}}$ as $\boldsymbol{P_{present}}$ and store it into the $\boldsymbol{S}$. Delete the current $\boldsymbol{i}$ from $\boldsymbol{US}$. Where the $\boldsymbol{S} = \{1\}$, $\boldsymbol{US} = \{2, 3, 4, 5\}$, as shown in Fig. 2. Find the closest point to $\boldsymbol{P_{present}}$ from the $\boldsymbol{US}$ and save it into the $\boldsymbol{P_{next}}$, then assign $\boldsymbol{P_{next}}$ to $\boldsymbol{P_{present}}$ and repeat the above operation until a complete route is found.

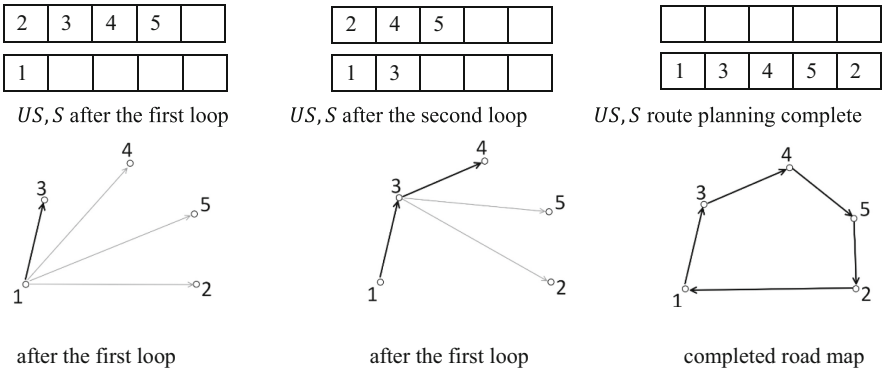

Fig. 2. Array US, S and its route

3.4 Greedy Crossover

The greedy crossover operator mainly utilizes the idea of crossover operation from Genetic Algorithm. The crossover operation of the genetic algorithm refers to the exchange of part of the genes between two paired chromosomes in a certain way. And generated two new individuals. The two-point Crossover operator, which is randomly setting two crossover points in the individual coding string, and then exchange the part of gene. However, using the genetic crossover operator to solve the TSP problem may result in repeated sequences, which makes a duplicates elimination operator was proposed to solve this problem.

For instance, there are two sets of city sequences x_i, x_j before the update, where $x_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $x_j = \{1, 7, 5, 3, 8, 9, 2, 6, 10, 4\}$. Running the crossover operator with individuals x_i, x_j. Assuming that the exchange points are $c_1 = 4$, $c_2 = 7$ respectively, the selected gene fragments are shown in the following Fig. 3.

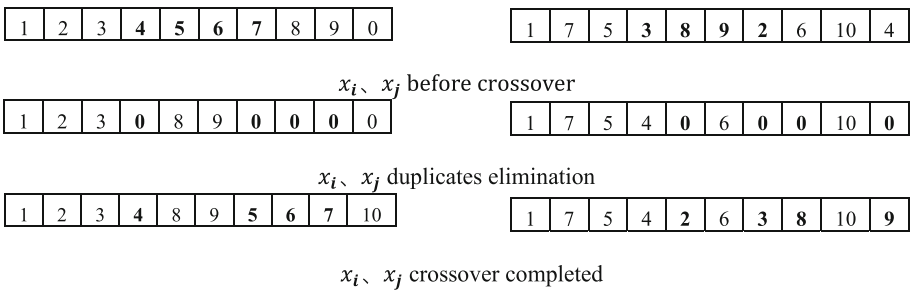

Fig. 3. x_i, x_j crossover operator

Perform the crossover operation on x_i, x_j respectively, and replace the 4^{th} to 7^{th} elements in x_i with $\{3, 8, 9, 2\}$, and replace the 4^{th} to 7^{th} elements in x_j with $\{4, 5, 6, 7\}$. Using duplicates elimination operator to eliminate the same elements on x_i,x_j.

Where $x_i = \{1, 2, 3, 0, 8, 9, 0, 0, 0, 10\}$, $x_j = \{1, 7, 5, 4, 0, 6, 0, 0, 10, 0\}$. Filling the non-appearing city sequence into the marker positions in turn, which makes $x_i = \{1, 2, 3, 4, 8, 9, 5, 6, 7, 10\}, x_j = \{1, 7, 5, 4, 2, 6, 3, 8, 10, 9\}$ sing greedy initialization operator for crossover area. For instance, x_i={1,2,3,4,8,9,5,6,7,10}. The crossover area of x_i is {4, 8, 9, 5}. Using the greedy initialization operator to reconstruct the route of the crossover area which makes the crossover area becomes {4, 9, 8, 5}. Then, using the idea of 2opt operator, the crossover area is connected into the original x_{i1} crossover area in original order and reverse order. $x_{i1} = \{1, 2, 3, 4, 9, 8, 5, 6, 7, 10\}$, $x_{i2} = \{1, 2, 3, 5, 8, 9, 4, 6, 7, 10\}$, and the route of x_{i1}, x_{i1} can be seen at Fig. 4. After the fitness value evaluation, it is found that the distance x_{i1} is shorter, so that makes $x_i = \{1, 2, 3, 4, 9, 8, 5, 6, 7, 10\}$. Similarly, perform the above operations on x_j, and compare the fitness values of x_{j1}, x_{j2} to select the best one as the city sequence output of this crossover area.

3.5 Local Mutation Strategy

In order to prevent the algorithm from falling into a local optimum in the middle and late iterations, a local mutation strategy is proposed. Where S_c is used to measure the difference between the normal individuals and the optical individuals, and S_{min} is used to evaluate the iterative stage of the algorithm. The calculation formulas of S_c, S_{min} are as follows:

$$S_c = \frac{\sqrt{|f(L_{best}) - f(x_N)|^2}}{1000} \tag{10}$$

$$S_{min} = \frac{1000 \times e^{-5}}{365^{\frac{Max_{iter}}{2.5t}}} \tag{11}$$

where f is the fitness function, Max_{iter} is the maximum number of iterations, and t is the current number of iterations. The value of Sc decreases with the fitness difference between the best individual and the worst individual in the current population according to the Eq. 10. At the same time, it can be concluded from Eq. 11 that $S_{\min}$ would increase with the increasing number of iterations t. S_{c}, $S_{\min}$ will change at the same time as the number of iterations increases. When $S_{\mathrm{c}}, < S_{\min}$, perform the mutation operation. Which is, generate a set of random sequences by system. In order to make the mutant individual converge quickly, using greedy crossover operator to cross the mutant with L_{best}. And using enchanced 2opt&3opt operator improving its quality. Finally, replace the current individual x_i with the newly generated mutant individual.

4 Experimental Studies

In order to verify the effectiveness of the proposed greedy initialization operator for elite individuals, Sect. 4.1 would use greedy initialization operator and random initialization operator to conduct control experiments with different proportions of elite individuals and normal individuals. After that, in Sect. 4.2, the crossover operator of Genetic Algorithm, greedy crossover operator, greedy crossover operator + 2opt&3opt and greedy

crossover operator + 2opt&3opt + local mutation strategy will be used to iteratively update and run the four operators, and compared their results. Finally, Sect. 4.3 will compare GSSA with excellent algorithms in recent literature. All the TSP instance are selected from TSPLIB [16].

4.1 The Best Proportion of Initialized Individuals

In order to find the best proportion of initialized individuals, a proportion experiment of elite individuals and normal individuals would be done. Elite individuals use greedy initialization operators to generate routes, while normal individuals use random initialization operators to generate routes. In the iterative process of the algorithm, greedy crossover combined with 2opt&3opt operator to update. The proportion of N_E and $N\text{-}N_E$ is shown in the Table 2. Population size N = 100. Since it is an evaluation of the quality of the solution in the initialization, set the maximum number of iterations $Max_{iter} = 100$. And the local mutation operator is temporarily eliminated from the algorithm. Where Avg.D stands for average value of length. P.D.A (%) = (Avg.D − optima) × 100%, which used to measure the standard deviation from the average value and the theoretical optimum. Where optima is the theoretical optimal value for the current TSP instance. The sum P.D.A represents the sum of the standard deviations from the mean optimal value for all tested TSP instances. This section selects 5 datasets of different scales TSP instances for testing, and the instances name are given in Table 2.

Table 2. The impact to initialization phase of different proportion N_E: $N\text{-}N_E$

N_E: $N\text{-}N_E$	Kroe100 (22068)		Kroa200 (29368)		Pr299 (48191)	
	P.D.A	Avg.D.	P.D.A	Avg.D.	P.D.A	Avg.D.
0:10	0.21	22114.13	0.10	29399.93	0.55	48458.31
1:9	0.28	22129.02	0.14	29407.76	0.38	48375.10
2:8	0.23	22119.87	0.07	29388.48	0.43	48402.15
3:7	0.22	22116.90	0.11	29399.03	0.32	48344.08
4:6	**0.19**	**22109.17**	0.16	29415.28	**0.31**	**48339.76**
5:5	0.19	22109.99	0.15	29414.01	0.37	48368.03
6:4	0.23	22119.83	**0.04**	**29378.49**	0.31	48341.07
7:3	0.26	22125.15	0.09	29395.17	0.66	48507.10
8:2	0.24	22120.83	0.12	29402.28	0.34	48357.09
9:1	0.24	22120.69	0.15	29413.45	0.36	48362.08
10:0	0.27	22127.18	0.23	29436.38	0.35	48359.24

(*continued*)

Table 2. (*continued*)

N_E: N-N_E	Rd400 (15281)		U574 (36905)		Sum
	P.D.A	Avg.D.	P.D.A	Avg.D.	P.D.A
0:10	**1.07**	**15444.07**	2.36	37774.62	4.29
1:9	1.60	15525.98	2.58	37858.60	4.98
2:8	1.16	15458.30	2.34	37767.03	4.23
3:7	1.34	15485.42	**2.17**	**37706.65**	4.16
4:6	1.43	15500.06	2.42	37798.24	4.51
5:5	1.37	15489.74	2.53	37837.50	4.61
6:4	1.41	15496.48	2.62	37871.73	4.61
7:3	1.52	15512.79	2.36	37775.24	4.89
8:2	1.10	15449.75	2.70	37900.65	4.50
9:1	1.11	15450.30	2.22	37724.78	**4.08**
10:0	1.10	15449.20	2.49	37824.64	4.44

* Set the best value in bold.

Although different proportions of initialization individuals have different effects on different scales of examples, still we can use P.D.A to evaluate the convergence of individuals with different TSP instances in the early stage of the algorithm. First, by comparing the values of P.D.A, it is found that with the increase scale of the TSP instance, P.D.A increases. Second, in order to find the best proportion of N-N_E in different scale TSP instances, comparing the sumP.D.A values of scale TSP instances, and find the smallest sumP.D.A value, and this proportion is used for the individuals initialization of the GSSA algorithm. Obviously, when N_E: N-N_E = 9:1, the value of sumP.D.A is the smallest. Therefore, N_E: N-N_E = 9:1 is selected as the best proportion of the individual initialization.

4.2 Comparison of SSA and Improved Operator

In order to verify the effectiveness of each improved operator, each new operator was added to the standard squirrel search algorithm (SSA), and three algorithms (SSA, SSA+GC, SSA+GC+LMS) were obtained. The three algorithms are compared to verify the effect of each improved operator on the performance of the algorithm for solving TSP problems.

The improved operators are added to the algorithm in turn. First test the SSA algorithm, which using the two-point crossover operator of the genetic algorithm to update the individuals in the SSA framework. Then test the SSA+GC algorithm, which replace the two-point crossover operator with the greedy crossover+2opt&3opt operator for comparison. Finally, the SSA+GC+LMS algorithm is tested. On the basis of using the greedy crossover+2opt&3opt operator, a local mutation strategy is added, which is the GSSA (Table 3).

Table 3. Test results of SSA, SSA+GC, SSA+GC+LMS (GSSA)

Problem	Optima	Method	Avg.	Best	S.D.	PEB (%)	PEA (%)
Krob100	22141	SSA	25404.96	25010.50	228.36	12.96	14.74
		SSA+GC	22175.12	22139.07	31.43	−0.01	0.15
		SSA+GC+LMS	**22168.20**	**22139.07**	**29.13**	**−0.01**	**0.12**
Eil101	629	SSA	729.01	715.05	6.50	13.68	15.90
		SSA+GC	640.68	640.21	0.95	1.78	1.86
		SSA+GC+LMS	**640.31**	**640.21**	**0.17**	**1.78**	**1.80**
Pr124	59030	SSA	66145.09	62603.74	1280.45	6.05	12.05
		SSA+GC	59030.76	59030.76	0.00	0.00	0.00
		SSA+GC+LMS	**59030.76**	**59030.76**	**0.00**	**0.00**	**0.00**
Ch150	6528	SSA	7069.80	7018.05	19.27	7.51	8.30
		SSA+GC	6539.15	6530.90	12.44	0.04	0.17
		SSA+GC+LMS	**6531.04**	**6530.90**	**0.63**	**0.05**	**0.04**
Pr152	73682	SSA	79369.37	79066.34	167.36	7.31	7.72
		SSA+GC	73724.40	73683.64	63.23	0.00	0.06
		SSA+GC+LMS	**73718.04**	**73683.64**	**59.58**	**0.00**	**0.05**
Kroa200	29368	SSA	34278.80	33744.78	235.37	14.90	16.72
		SSA+GC	29381.53	29369.41	24.23	0.00	0.05
		SSA+GC+LMS	**29373.26**	**29369.41**	**8.07**	**0.00**	**0.02**
Pr299	48191	SSA	57573.16	56821.25	430.62	17.91	19.47
		SSA+GC	48278.20	48198.78	37.67	0.02	0.18
		SSA+GC+LMS	**48211.05**	**48194.92**	**18.92**	**0.01**	**0.04**
Rd400	15281	SSA	18069.10	18268.90	59.31	18.24	19.55
		SSA+GC	15374.63	15326.76	31.28	0.30	0.61
		SSA+GC+LMS	**15350.24**	**15300.55**	**30.88**	**0.13**	**0.45**
Pr439	107217	SSA	125566.03	123865.35	924.40	15.53	17.11
		SSA+GC	108234.93	107407.15	931.50	0.18	0.95
		SSA+GC+LMS	**108085.30**	**107268.67**	**873.06**	**0.05**	**0.81**
U574	36905	SSA	44491.75	44416.23	33.16	20.35	20.56
		SSA+GC	37669.82	37664.07	8.78	2.06	2.07
		SSA+GC+LMS	**37421.35**	**37352.18**	**45.28**	**1.21**	**1.40**

* Set the best value in bold.

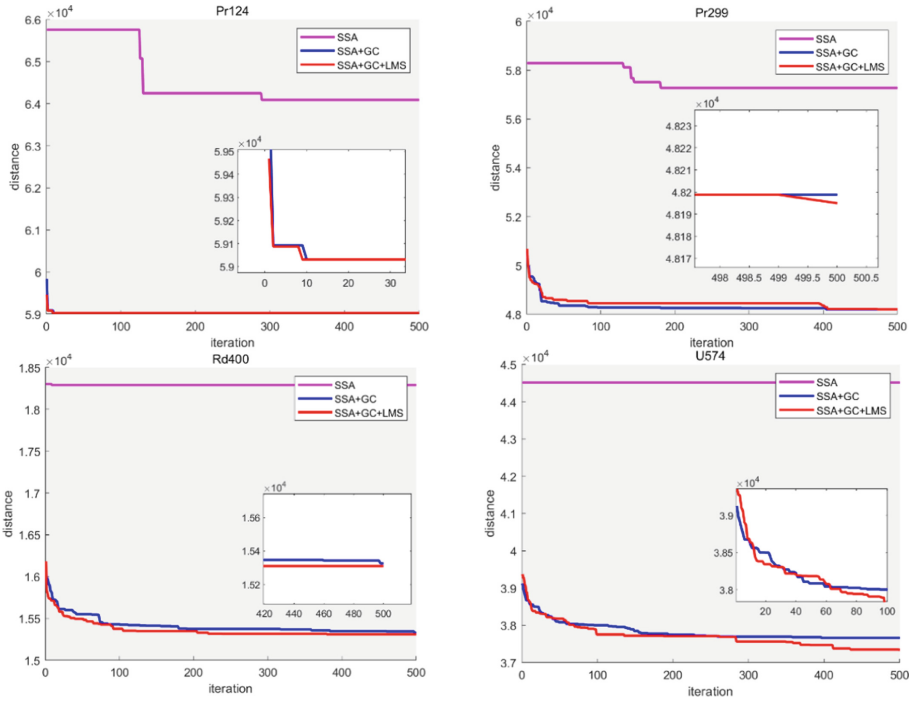

Fig. 4. Convergence curve of pr124 krob200 pr439 u574

4.3 Comparing with Latest Research

(1) GSSA vs TO-GWO

Reference [17] redesigned the gray wolf optimization algorithm (GWO)[18], combined exchange, shift and symmetric transformation operators to solve the permutation-encoded traveling salesman problem (TSP), and named as transformation operator based grey wolf optimizer (TO-GWO). In TO-GWO, each wolf represents a possible solution of TSP, and the wolf interacts with the leading wolf through exchange, shift and symmetry operators to obtain the optimal solution of TSP. In order to improve the local search ability of the algorithm in solving discrete problems, the 2-opt algorithm is also been used. According to the experiments results, the algorithm is efficiency in solving TSP problems (Table 4).

(2) GSSA vs DSFLA

In [19], a heuristic-based discrete shuffled frog leaping algorithm was proposed, and the traveling salesman problem was used as the test problem. First, a new individual generation operator is designed using the nearest neighbor information. Then, four improved

Table 4. The comparison result of TO-GWO and GSSA on TSP instances.

Problem		TO-GWO [17]				GSSA			
Name	Optima	Avg.	Best	S.D.	P.D.A	Avg.	Best	S.D.	P.E.A
kroa100	21282	**21285.44**	**21285.44**	0.00	0.02	**21285.44**	**21285.44**	0.00	0.01
Pr124	59030	**59030**	**59030**	0.00	0.00	**59030**	**59030**	0.00	0.00
Ch150	6528	6539.87	6530.92	19.27	8.30	**6531.04**	**6530.90**	0.63	0.04
Kroa200	29368	29646.05	29468	114.71	0.95	**29373.26**	**29369.41**	8.07	0.02
Pr299	48191	48402.87	48307.04	67.42	0.44	**48211.05**	**48194.92**	18.92	0.01
Rd400	15281	15567.63	15507.42	59.31	18.24	**15350.24**	**15300.55**	30.88	0.13
Pr439	107217	108037.04	107819.04	149.34	0.76	108085.30	**107268.67**	873.06	0.81
D493	35002	35641.26	35443.72	85.62	1.83	**35313.05**	**35201.27**	54.99	0.89
U574	36905	37862.83	37711.45	70.97	2.60	**37421.35**	**37352.18**	45.28	1.21
Rat783	8806	9222.56	9149.38	27.21	4.73	**8972.71**	**8960.12**	8.24	1.89
Pr1002	259045	268402.25	266636.49	813.68	3.61	**262503.30**	**261915.65**	393.31	1.34

* Set the best value in bold.

search strategies are designed to improve the algorithm performance. And the effectiveness of the new individual generation operator and four improved strategies is verified by experiments (Table 5).

Table 5. The comparison result of DSFLA and GSSA on TSP instances.

Problem		DSFLA [19]				GSSA			
Name	Optima	Avg.	Best	S.D.	P.D.A	Avg.	Best	S.D.	P.E.A
Att48	33522	33567.27	**33522**	54.97	4.25	**33523**	33523	0.00	0.01
Rat99	1211	**1216.80**	**1211**	0.84	1.87	1219.24	1219.24	0.00	0.68
Kroa100	21282	21312.03	**21282**	50.01	0.14	**21285.44**	21285.44	0.00	0.02
Eil101	629	**632.90**	**629**	3.65	0.62	640.31	640.21	0.17	1.78
Lin105	14379	14423.93	**14379**	55.82	0.31	**14383**	14383	0.00	0.03
Pr124	59030	59503.43	**59030**	0.80	0.00	**59030**	**59030**	0.00	0.00
Ch130	6110	6211.97	6140	47.66	1.67	**6123.48**	**6110.72**	11.42	0.22
Pr144	58537	58632.10	58537	93.42	0.16	**58535.22**	**58535.22**	0.00	0.00
Ch150	6528	6562.83	6533	12.95	0.53	**6531.04**	**6530.90**	0.63	0.04
pr152	73682	73970.97	73682	271.23	0.39	**73718.04**	**73683.64**	104.36	0.05
Kroa200	29368	29671.37	29499	135.84	1.03	**29373.26**	**29369.41**	8.07	0.02

* Set the best value in bold.

5 Conclusions

The experiments compare the effects of the proposed operators and strategies on the performance of SSA to verify their effectiveness. The cumulative improvement strategies (SSA+GC, SSA+GC+LMS) are compared with the SSA, and the influence of each improved operator and strategy on SSA is verified. The experimental results show that SSA+GC, SSA+GC+LMS outperform SSA on all instances. And when the TSP problem size is less than 150, the performances of SSA+GC and SSA+GC+LMS are close. When the TSP problem size is greater than 150, SSA+GC+LMS (GSSA) is better. The above results show that each strategy can improve the solution accuracy and stability of the algorithm. The third group compares the proposed GSSA with TO-GWO, DFSLA in the existing literature. The results showed that GSSA was significantly better than TO-GWO, DSFLA under different instances. The above results show that the algorithm has high accuracy and good stability in solving TSP problems.

Acknowledgment. This work is supported by National Science Foundation of China under Grants No. U21A20464, 62066005.

References

1. Junger, M., Reinelt, G., Rinaldi, G.: Chapter 4: the traveling salesman problem In: Handbooks in Operations Research and Management Science, vol. 7, pp. 225–330 (1995)
2. Alexandridis, A., Paizis, E., Chondrodima, E., Stogiannos, M.: A particle swarm optimization approach in printed circuit board thermal design. Integr. Comput. Aided Eng. **24**(2), 143–155 (2017)
3. Savla, K., Frazzoli, E., Bullo, F.: Traveling salesperson problems for the Dubins vehicle. IEEE Trans. Autom. Control **53**(6), 1378–1391 (2008)
4. Liao, T.Y.: On-line vehicle routing problems for carbon emissions reduction. Comput. Aided Civ. Infrastruct. Eng. **32**(12), 1047–1063 (2017)
5. Hacizade, U., Kaya, I.: GA based traveling salesman problem solution and its application to transport routes optimization. IFAC-PapersOnLine **51**(30), 620–625 (2018)
6. Purcaru, C., Precup, R.E., Iercan, D., Fedorovici, L.O., David, R.C., Dragan, F.: Optimal robot path planning using gravitational search algorithm. Int. J. Artif. Intell. **10**(13), 1–20 (2013)
7. Saraswathi, M., Murali, G.B., Deepak, B.B.V.L.: Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm. Procedia Comput. Sci. **133**, 510–517 (2018)
8. Ali, M.K.M., Kamoun, F.: Neural networks for shortest tour computation and routing in computer networks. IEEE Trans. Neural Netw. **4**(5), 941–953 (1993)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
10. Papadimitriou, C.H.: Euclidean traveling salesman problem is NP-complete. Theoret. Comput. Sci. **4**, 237–244 (1977)
11. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature **406**(6791), 39–42 (2000)
12. Ozsoydan, F.B., Baykasoglu, A.: A swarm intelligence-based algorithm for the set-union knapsack problem. Future Gener. Comput. Syst. **93**, 560–569 (2019)
13. Yang, X.-S.: Swarm intelligence based algorithms: a critical analysis. Evol. Intell. **7**(1), 17–28 (2013). https://doi.org/10.1007/s12065-013-0102-2

14. Jain, M., Singh, V., Rani, A.: A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol. Comput. Sci. **44**, 148–175 (2019)
15. Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**(6), 791–812 (1958)
16. Reinelt, G.: TSPLIB-a traveling salesman problem library. ORSA J. Comput. **3**(4), 267–384 (1991)
17. Panwar, K., Deep, L.: Transformation operators based grey wolf optimizer for travelling salesman problem. J. Comput. Sci. **101454**, 55 (2021)
18. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**(3), 46–61 (2014)
19. Huang, Y., Shen, X.-N., You, X.: A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem. Appl. Soft Comput. **107085**, 102 (2021)