Artificial Intelligence Review (2021) 54:3539–3574
https://doi.org/10.1007/s10462-020-09931-5

Improved social spider algorithm for large scale optimization

Emine Baş[1] · **Erkan Ülker**[2]

Published online: 9 November 2020

Abstract
Heuristic algorithms can give optimal solutions for low, middle, and large scale optimization problems in an acceptable time. The social spider algorithm (SSA) is one of the recent meta-heuristic algorithms that imitate the behaviors of the spider to perform global optimization. The original study of this algorithm was proposed to solve low scale continuous problems, and it is not be solved to middle and large scale continuous problems. In this paper, we have improved the SSA and have solved middle and large scale continuous problems, too. By adding two new techniques to the original SSA, the performance of the original SSA has been improved and it is named as an improved SSA (ISSA). In this paper, various unimodal and multimodal standard benchmark functions for low, middle, and large-scale optimization are studied for displaying the performance of ISSA. ISSA's performance is also compared with the well-known and new evolutionary methods in the literature. Test results show that ISSA displays good performance and can be used as an alternative method for large scale optimization.

Keywords Heuristic methods · Optimization · Social spider algorithm · Large-scale dimension

1 Introduction

For the last 20 years, swarm intelligence-based algorithms attract attention. The term of a swarm expresses a community that consists of individuals who are in communication with each other. Swarm intelligence-based algorithms consider social animal and insect behaviors for solving problems. These algorithms imitate behaviors of ant, bee, bacteria, butterfly, etc. Thus, the problems which seem hard can be solved (Parpinelli and Lopes 2011; Yu and Li 2015). A computational optimization methodology involves finding feasible solutions from a finite set of solutions and identifying only the optimal solution(s). Swarm

✉ Emine Baş
emineozcan@selcuk.edu.tr

Erkan Ülker
eulker@ktun.edu.tr

[1] Kulu Vocational School, Selçuk University, 42075 Konya, Turkey

[2] Department of Computer Engineering, Faculty of Engineering and Nature Sciences, Konya Technical University, 42075 Konya, Turkey

intelligence algorithms constitute a sub-class of the computational optimization methodology (Blum 2008). Swarm intelligence algorithms are developed based on the emergence of collective behaviors about a population of interacting individuals in adapting to the local and/or global environments. Examples of swarm intelligence algorithms include Particle Swarm Optimisation (PSO) (Kennedy 2010), Ant Colony Optimisation (ACO) (Dorigo 2006), Bat Algorithm (BA) (Yang 2010a, b), Firefly Algorithm (FA) (Yang 2010a, b), Cuckoo Search Algorithm (CSA) (Gandomi et al. 2013), bee-inspired algorithms (Karaboga 2005; Wong et al. 2008; Masutti and Castro 2016), etc. Particle Swarm Optimization (PSO) is an optimization method, which is developed by Kennedy and Eberhart (1995) inspired by flocking fish and insects. PSO is based on social information sharing between individuals. Each individual is called a particle and the particles which create the population are called a swarm. Each particle adjusts their positions according to the best position in the swarm by using their previous experience. ACO is an algorithm inspired by ants' behaviors. The aim is to find the shortest path to food sources for an ant colony. This heuristic structure is used for finding appropriate solutions for optimization problems (Dorigo 2006). Karaboga and Basturk have designed an artificial bee colony optimization (ABC) algorithm. ABC has classified bees in the hive as three types. Explorer bees are unguided random flying bees. Worker bees search location areas. Observer bees are the bees that use fitness value that they search for a proper solution. Worker and observer bees are used in the local solutions for stabilizing the balance between algorithm exploration and exploitation. Explorer bees are used for the global solution. ABC shows satisfactory performance in applications (Karaboga 2005; Karaboga and Basturk 2007; Akay 2013). Bats emit sound impulses at certain frequencies. Bats can identify their prey and avoid obstacles through the use of this technique. Bat algorithm was developed using this technique (Tawhid and Dsouza 2018). Firefly algorithm is a population-based algorithm because the search process is simple and easy to implement, it has received extensive attention from scholars (Wang and Song 2019). Firefly algorithm (FA) is a new nature-inspired algorithm recently developed by Yang (Yang 2009). Wang et al. gave an improved firefly algorithm that each firefly is guided by a brighter one in the neighborhood to improve the ability of the algorithm (Wang et al. 2017). Yang and Deb propose the cuckoo search algorithm (CSA) as a structured randomized search optimization algorithm motivated by the combination of the holoparasite characteristics and Levy flight foraging configurations of some cuckoo species. (Yang and Deb 2009). Similarly, other creatures are widely studied.

1.1 Large scale global optimization

Many problems in various disciplines such as engineering involve several system (decision) parameters (or variables) with some take on continuous values while others are restricted to a set of discrete values. These discrete variables could take on real values as in thickness of the material, or integer values as in mesh size of abrasives (Liao et al. 2012). There are many discrete optimization problems in the literature (For example Traveling Salesman Problem (TSP), knapsack problem, etc.). They are NP-hard problems (Kuzu et al. 2014). These kinds of problems are difficult to solve by classical methods. Technically, some system parameters can take on infinite values, too. Optimizations with this type of variable are continuous optimizations. Continuous optimizations can have a low or large number of decision variables. Most real-world optimization problems have a large number of decision variables. Problems with such a large number of decision variables are known as Large Scale Global Optimization (LSGO). Many of these problems are complex and difficult to

solve because of their large-scale dimensions. The quality of the final solution dramatically decreases as the size of such problems' dimension increase (dimension > 10) and calculating time is increasing. Many real-life problems are large scale optimization problems. For example optimization of water management problem (Goh et al. 2011), optimization of Norwegian natural gas production and transport (Ray and Yao 2009), optimizing US army stationing (Dell et al. 2008), etc. The increased need for high-quality decision making for such problems has made the topic of solving large scale optimization problems a valuable and challenging research area (Sayed et al. 2015). Many researchers have started to work on large-dimensional optimization in the literature and many metaheuristic algorithms have been proposed as DEVIIC (Sayed et al. 2015), MWOA (Sun et al. 2018), CC (Trunfio et al. 2016), DE (Maucec and Brest 2019), FDA (Nakib et al. 2017), etc.

1.2 Swarm methods for LSGO

Many studies are available for the solution of LSGO problems. Recently, many algorithms based swarm in the literature have been proposed for LSGO. The literature review has focused more on swarm methods since ISSA is a swarm-based algorithm. Table 1 shows some of the earlier research algorithms for LSGO.

1.3 SSA for continuous optimization

Yu and Li (2015) have offered the Original Social Spider Algorithm (SSA) which is configured according to social spider behaviors. Some spider species, which generally live solitary, can live as colonies. Spiders live in colonies can communicate with each other by web structure. The vibration they produce on the web enables this communication. They can make random walks to each other by the vibration information they share (Yu and Li 2015). SSA is a continuous optimization algorithm that solves low-scale optimization problems in its original form and sometimes it seems to remain weak in reaching optimum. El-Bages and Elsayed (2017) proposed an algorithm and have solved the static transmission expansion planning problem. In the proposed method; the DC power flow sub-problem is solved by developing an original SSA based web and adding potential solutions to the result. Yu and Li (2016) developed an original SSA and have solved the Economic Load Dispatch (ELD) formulation. Mousa and Bentahar (2016) have solved the QoS-aware web service selection process. They used the SSA approach. Elsayed et al. (2016) have solved the non-convex economic load dispatch problem. They used an SSA based approach. Baş and Ülker (2020a) have proposed binary SSA for continuous optimization task and Baş and Ülker (2020b) have solved the feature selection problem with binary SSA. Cuevas et al. (2013) develop Social Spider Optimization (SSO) inspired by spider behavior and one of the other swarm intelligence algorithms that are different from SSA. In SSO, the spiders are divided according to their different genders. The algorithm considers two different search agents (spiders): Males and females. Depending on gender, each individual is conducted by a set of different evolutionary operators that mimic different cooperative behaviors that are typically found in the colony. Different studies have been done in the literature with SSO (Cuevas and Cienfuegos 2014; Shukla and Nanda 2016, 2018). The SSA used for discretization in the proposed paper is completely different from SSO.

In this paper, the original SSA is considered. When the literature is analyzed, it is noticed that SSA's performance is not measured for high-dimensional continuous optimization problems. Yu and Li measured the original SSA for 10, 20, and 30 dimensions.

Table 1 The earlier research algorithms for LSGO tasks in the literature

Algorithm	Instruction
DEVIIC	Sayed et al. (2015) solved the LSGO problem with developed technique VIIC with DE
MWOA	Sun et al. (2018) proposed a modified Whale Optimization Algorithm (MWOA) for solving LSGO problems
CC	Trunfio et al. (2016) improved a Cooperative Coevolutionary (CC) approach for solving problems of LSGO
DE	Maucec and Brest (2019) gave a review of recent extensions of the Differential Evolution (DE) algorithm for use in LSGO and present an empirical analysis of DE-based and some other state-of-the-art algorithms for LSGO on the CEC 2013 LSGO benchmark suite
FDA	Nakib et al. (2017) proposed a geometrical fractal decomposition based on hypersphere as an elementary geometric form
SOMAQI	Singh and Agrawal (2016) improved a Self Organizing Migrating Algorithm with Quadratic Interpolation (SOMAQI). They have been extended to LSGO problems for dimensions ranging from 100 to 3000 with a constant population size of 10 only
MCSO	Mohapatra et al. (2017) improved the Modified Competitive Swarm Optimizer (MCSO) for LSGO. The proposed MCSO is applied to solve the standard CEC2008 and CEC2013 large scale unconstrained benchmark optimization problems
IGWO	Long et al. (2018) present Inspired Grey Wolf Optimizer (IGWO) which extends the original grey wolf optimizer by adding two features, namely, a nonlinear adjustment strategy of the control parameter, and a modified position-updating equation based on the personal historical best position and the global best position. They have performed experiments on four classical high-dimensional benchmark functions
ISCA	Long et al. (2019) present an improved version of the Sine Cosine Algorithm (ISCA) for solving high-dimensional problems. A modified position-updating equation by introducing inertia weight is proposed to accelerate convergence and avoid falling into the local optima
MGWO	Mittal et al. (2016) improved Modified Grey Wolf Optimizer (MGWO) for global engineering optimization
µDSDE	Yildiz and Topal (2019) proposed micro Differential Evolution with a Directional Local Search (µDSDE) algorithm using a small population size to solve LSGO problems
JOA	Sun et al. (2016) improved the joint operations algorithm (JOA). They tested JOA on twenty LSGO benchmark functions of the IEEE CEC 2010 special session and four real-life problems

They did not measure the original SSA for 100, 500, and 1000 dimensions. Sometimes heuristic algorithms do not perform well at higher dimensions even if they perform at low dimensions. In this paper, we measured the performance of SSA for 10, 20, 30, 100, 500, and 1000 dimensions. We compared the performance of SSA at low and high scale dimensions. Two new techniques are added for improving the performance of sustained SSA. So, improved SSA (ISSA) is created. ISSA has been tested in low-scale and large-scale dimensions with nineteen different unimodal and multimodal standard benchmark functions. And the results are compared. ISSA is compared with SSA and various well-known methods in the literature for a more detailed interpretation of its performance.

The organization of the paper as follows: related works are analyzed in Sect. 1. In Sect. 2, the original SSA, and Sect. 3, ISSA algorithms are detail examined. In Sect. 4, SSA and ISSA are tested in various unimodal and multimodal benchmark functions in low, middle, and large-scale dimensions. Obtained results are compared with each other and well-known heuristic and new methods in the literature. Finally, the results are discussed.

2 Social spider algorithm (SSA)

Yu and Li (2015) proposed Social Spider Algorithm (SSA). The SSA is inspired by spider behavior in nature and a meta-heuristic algorithm. SSA has designed the search space like a spider web. In SSA, spiders are SSA agents that perform optimization. Spiders produce vibration in their position. The vibration is the most important property that distinguishes SSA from other heuristic algorithms. Thanks to this vibration, information sharing takes place in SSA. Each spider receives the vibration intensities produced by other spiders and decides which direction to perform their random walk. The flowchart for the original SSA is shown in Fig. 1.

2.1 Initialization and fitness evaluation

Required constant parameter sets are adjusted. For example, lower and upper limits of variables, number of maximum iteration, etc. In the SSA, the positions of spiders are randomly generated in the search space. Each spider in the spider population has calculated its fitness value.

2.2 Vibration generation

All spiders produce vibration intensity using the source position. Equation 1 has shown vibration intensity (Yu and Li 2015). The SSA uses vibrations to activate the propagation process. The vibration attenuation over distance is calculated by Eq. 2 and distance is calculated by Eq. 3 (Yu and Li 2015). "V" represents spiders' vibrations. Each spider receives the vibrations generated by other spiders. A spider selects the strongest of these vibration values. This value shows as V_x^{best}, x represents a spider. Spider x stores the target vibration

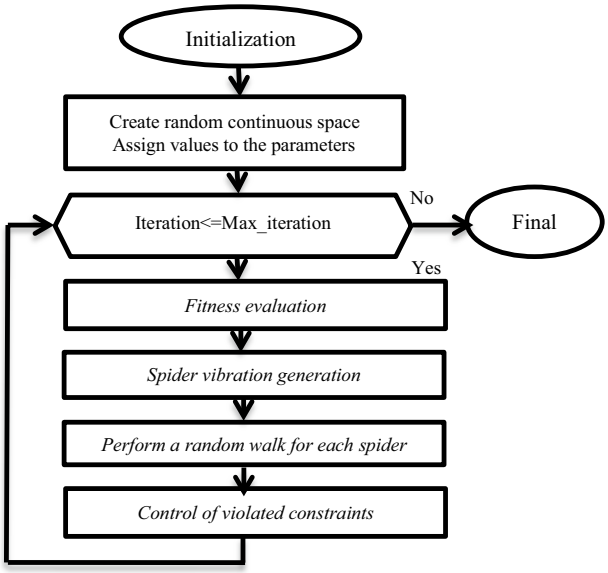

Fig. 1 Flowchart for the original SSA

in the memory as V_x^{target}. Each spider x compares V_x^{best} and V_x^{target} values. If the intensity of the V_x^{best} is greater than V_x^{target}, the V_x^{target} is changed to as V_x^{best}.

$$I(P_x, P_x, t) = \log\left(\frac{1}{f(p_x) - C}\right) + 1 \tag{1}$$

where $I(P_x, P_x, t)$ is the vibration value produced by the spider in the source position at time t. "x" represents a spider. $P_x(t)$ defines the position of spider x at time t and it is shown simply as P_x if the time argument is t. $f(P_x)$ represents the spider of fitness value in its current position. C is a confidently small constant such that all possible fitness values are larger than C value for minimization problems. Logarithms are used for operations with numbers that are too large or too small.

$$I(P_x, P_y, t) = I(P_x, P_x, t) \times exp\left(-\frac{Distance(Px, Py)}{\bar{\sigma} \times r_a}\right) \tag{2}$$

$$Distance(P_x, P_y) = P_x - P_y \tag{3}$$

where $I(P_x,P_y,t)$ is the felt value of the spider's vibration in "x" point, by the spider in "y" point. $P_x(t)$ or simply P_x represents the position of spider x at time t. $P_y(t)$ or simply P_y represents the position of spider y at time t. $I(P_x, P_x, t)$ is the vibration value produced by the spider in the "x" position at time t. $\bar{\sigma}$ represents the mean of the standard deviation of the positions of all spiders in each dimension. r_a is represented as a user-controlled parameter $r_a \in (0, \infty)$. Distance(P_x, P_y) is showing the distance between spider x and spider y.

2.3 Random walk

The random walk of each spider is calculated by Eq. 4 (Yu and Li 2015). Spider x first moves along its previous direction, which is the direction of movement in the previous iteration. The distance along this direction is a random portion of the previous movement. Then spider x approaches P_x^{follow} along each dimension with random factors generated in (0, 1). P_x^{folow} represents the following position. Each spider decides the following position (P_x^{follow}) by looking at the dimension mask.

$$P_x(t+1) = P_x + \left(P_x - P_x(t-1)\right) \times r + \left(P_x^{follow} - P_x\right) \odot R \tag{4}$$

where R is a vector of random float-point numbers generated from zero to one uniformly. Θ denotes element-wise multiplication.

The dimension mask is determined so that a random walk can be performed towards the V_x^{target}. The dimension mask is a {0, 1} binary vector of length D and D is the dimension of the optimization problem. In each iteration, spiders have a probability of $1 - p_c^{cs}$ to change its dimension mask. $p_c \in (0, 1)$ is a user-defined parameter that describes the probability of changing dimension mask. If the dimension mask is decided to be changed, each bit of the vector has a probability of p_m. cs is the number of iterations that the spider has last changed its target vibration. After the dimension mask is determined, a new following position P_x^{follow} is generated based on the dimension mask. P_x^{follow} value is calculated by Eq. 5 (Yu and Li 2015). If the bit value of the dimension mask is 0, it moves to the position of the target spider in the memory of the spider, or the position of a random spider.

$$P_{x,i}^{follow} = \begin{cases} P_{x,i}^{target}, & dimension\_mask_{x,i} = 0 \\ P_{x,i}^{r}, & dimension\_mask_{x,i} = 1 \end{cases} \quad (5)$$

where r is a random integer value generated in [1, population size] and dimension_mask$_{x,i}$ stands for the ith dimension of the dimension mask of spider x.

2.4 Control of violated constraints

After each spider performs a random walk, the spiders may move out of the web. They violate the constraint of the optimization problem. Equation 6 is used to prevent population members from moving out of the search space.

$$P_{x,i}(t+1) = \begin{cases} \left(\overline{U_i} - P_{x,i}\right) \times r & if \;\; P_{x,i}(t+1) > \overline{U_i} \\ \left(P_{x,i} - \underline{L_i}\right) \times r & if \;\; P_{x,i}(t+1) < \underline{L_i}, \end{cases} \quad (6)$$

where $\overline{U_i}$ shows upper border and $\underline{L_i}$ shows a lower border. $P_{x,i}(t+1)$ defines the position of a spider x for the ith dimension at time t+1. r is a random integer value generated in [1, population size].

2.5 r_a, p_c, and p_m parameter values

r_a controls the attenuation rate of the vibration intensity over the distance. $p_c \in (0, 1)$ is a user-defined parameter that describes the probability of changing dimension mask. p_m is a user-controlled parameter defined in (0, 1). The value of r_a is selected from the set {1/10, 1/5, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5, 10}. The values of p_c and p_m are selected from the set {0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99}. Yu and Li are used five 10-dimensional benchmark functions to investigate the impact of these parameters on the performance of SSA. According to their results, r_a, p_c, and p_m values are determined as 1, 0.7, and 0.1, respectively.

3 Improved social spider algorithm (ISSA)

The original Social Spider Algorithm (SSA) is an optimization algorithm that is developed for solving continuous problems. In this paper, SSA is studied and it is improved by adding new two techniques enhancing its performance (ISSA). Original SSA is improved by two different techniques. These techniques are spider blasting and explorer spider memory methods. In spider blasting, the worst spider is determined by the spider to explode in the ISSA. Then, this spider is regenerated as a new spider. Thus, ISSA can discover new points in the search space. Previous good spiders are prevented from being forgotten by using the explorer spider memory technique. Thus, generations are evolved better. This increases ISSA's local search capability.

The working stages of ISSA are shown in Fig. 2 with general headings and details. In this table, newly added steps to SSA are indicated by bold letters. In ISSA, steps 1–4 are run similar to SSA. With steps 5 and 6, the exploration and exploitation ability of ISSA is improved and new better candidate solutions are obtained. The balance between local

Step 1. *Initialization:* Parameters of values are set. Random starting position locations are determined for each spider in the swarm. The target vibration for each spider is assigned.
Step 2. *Fitness evaluation:* Fitness value is calculated for each spider. The best value in the spider population is detected and saved.
Step 3. *Vibration generation*: Vibration value is calculated for each spider calculated by using vibration Equations (1, 2, and 3). The best value in the spider population is detected and saved. The received the best vibration value is compared to the target vibration value for each spider in the population.
Step 4. *Random walk:* Update the dimension mask. P^{follow} value is calculated by Equation 5. Each spider moves to a new position by Equation 4.
Step 5. ***Spider blasting:*** G_{worst} has been determined and omitted. The $spider_{new}$ has been randomly added.
Step 6. ***Explorer spider memory:*** An explorer spider memory area is generated and several explorer spiders are added to the population.
Step 7. *Control of violated constraints:* Violations are checked.
Step 8. *Repeat:* All criteria beginning from Step 2 are repeated until reaching stopping criteria.
Step 9. *Output:* Optimum outputs are obtained.

Fig. 2 The work steps of ISSA

search and global search is achieved. Thus, the existing population is replaced with better individuals.

3.1 Spider blasting

The worst spider (G_{worst}) in the original SSA is hidden throughout each iteration. This spider has a small effect on the optimum result. ISSA has determined and omitted the worst spider (G_{worst}) and has added a new random spider ($spider_{new}$) which can have more effect on the optimum result. The fitness value of the new spider is controlled while adding it. If the fitness value of the new spider is worse than the fitness value of the worst spider (G_{worst}) of the population, it is never added to the population. The system preserves existing population members. If the fitness value of the new spider is better than the fitness value of the worst spider (G_{worst}) of the population, G_{worst} is omitted from the population and a new spider is replaced ($spider_{new}$). So, a predetermined fixed population amount is preserved. This process is repeated throughout each iteration. All processes of spider blasting technique are shown in Fig. 3 only for one iteration.

3.2 Explorer spider memory

An explorer spider memory area is developed for enhancing the performance of ISSA. This technique has two steps. In the first step, the explorer spider memory area is created. In the second step, some amount of spiders are chosen from the explorer spider memory and added to the population.

The first step: An Explorer Spider Memory (ESM) is created which is four times bigger than the number of population in the beginning ($4 \times N$) (N is the population size). Initially, the spiders in the ESM are created in random positions. In each iteration, the spider which has the worst fitness value of ESM (ESM_{worst}), is compared with the fitness value of the spider which has the best fitness value (G_{best}) in population. If the fitness value of G_{best} spider is better, it is added to the ESM. If G_{best} spider is added to ESM, the spider which has the worst fitness value in ESM (ESM_{worst}) is omitted from ESM. So, the number of individuals in ESM is preserved. In this comparison, if the fitness value of G_{best} spider is worse, no replacement is done in ESM. So, the ESM holds the best spiders of the system

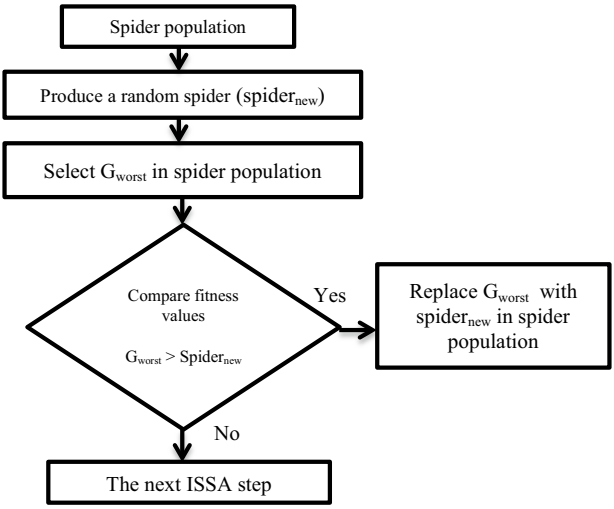

Fig. 3 Spider blasting process

until all earlier iteration up to the current iteration. Steps of ESM create process are shown in Fig. 4 for only one iteration.

The second step: Explorer spider memory is created in the first step. A certain number of explorer spiders are taken and added to the spider population in the second step. Initially, the number of the spider (q), which will be taken from explorer spider memory,

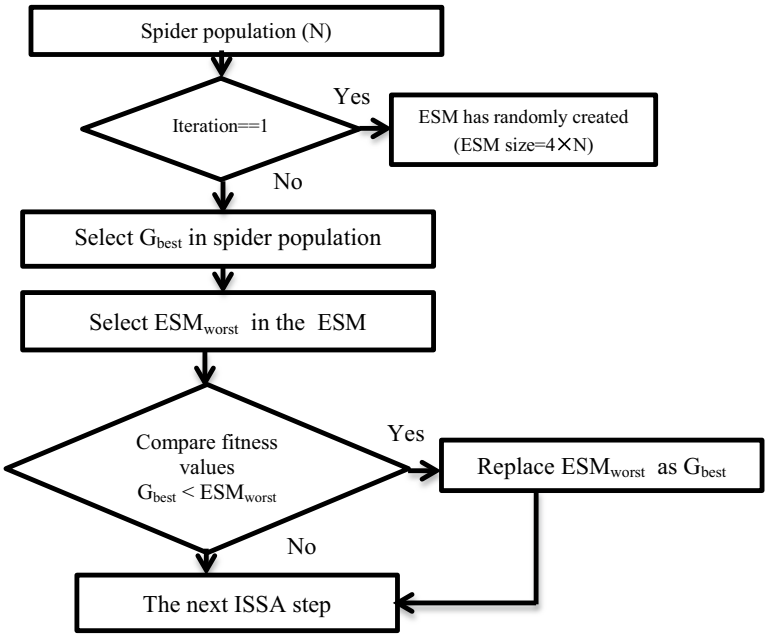

Fig. 4 Explorer spider memory (first step)

is determined. This ratio provides diversity in the population and is determined as a percentage. With this ratio, the amount of spiders, which will be taken from the spider population (N) is determined via Eq. 7.

$$s = ceil\left(\frac{N \times q}{100}\right) \quad s_i = \{s_1, s_2, \ldots, s_N\} \tag{7}$$

where s shows the number of spiders which will be taken from explorer spider memory, N shows spider population, q shows the ratio of the number of spiders which will be taken from explorer spider memory. ceil() shows a mathematic function that rounds up the result.

In every 100 iterations, the s explorer spiders are determined randomly from ESM. The fitness values of s explorer spiders compare with the worst fitness value in the population (G_{worst}). If the s_i explorer spider has better fitness value than G_{worst} spider, the G_{worst} spider in the population is saved as s_i explorer spider. Otherwise, no action has been taken. Thus, by adding randomly selected s explorer spiders from ESM, the population will have better individuals. With this technique, ISSA's local search capability has been developed. Figure 5 shows adding s_i explorer spider in every 100 iterations of the process. The working stages of explorer spider memory technical are shown in Fig. 6 with general headings and details.

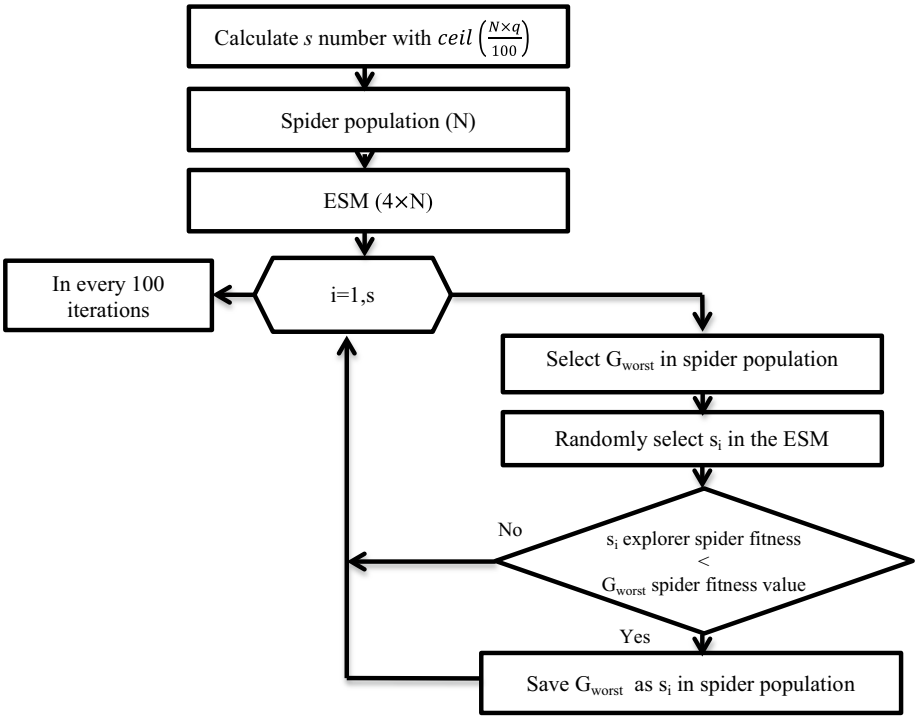

Fig. 5 Explorer spider memory (second step)

Step 1. An Explorer Spider Memory (ESM) is created which is four times bigger than the number of population in the beginning (*4×N*) (N is the population size).
Step 2. Initially, the spiders in the ESM are created in random positions.
Step 3. In each iteration, the spider which has the worst fitness value of ESM (ESM_{worst}), is compared with the fitness value of the spider which has the best fitness value (G_{best}) in population. If the fitness value of G_{best} spider is better, it is added to the ESM.
Step 4. Determine the amount of s as a percentage via Equation 7.
Step 5. In every 100 iterations, the s explorer spiders are determined randomly from ESM.
Step 6. The fitness values of s explorer spiders compare with the worst fitness value in the population (G_{worst}).
Step 7. If the s_i explorer spider has better fitness value than G_{worst} spider, the G_{worst} spider in the population is saved as s_i explorer spider. Otherwise, no action has been taken.

Fig. 6 The work steps of explorer spider memory technical

4 Experimental results and analysis

Improved Social Spider Algorithm (ISSA) ve original Social Spider Algorithm (SSA) are tested on Matlab R2014a that installed over windows 7, 64 bit, the system of 2.30 GHz processor with 4 GB RAM. Common parameters used in ISSA and SSA comparison, which are population size (N), dimension (n), maximum iteration value, r_a value, p_c value, p_m value, the Explorer Spider Memory (ESM) size, and q value used in ISSA are shown in Table 2.

In the literature, benchmark test functions have been used for evaluating the performance of metaheuristic algorithms. A strong relationship between metaheuristic algorithms and numerical test problems or test functions is found in the existing literature. Generally, the introduction of any newmetaheuristic algorithm is often accompanied by a set of benchmark test functions that are used to prove efficiency of the algorithm (Garden and Engelbrecht 2014). Algorithms that perform well on a set of numerical optimization problems are considered as effective methods for solving real-world problems. Any optimization algorithm, including metaheuristic algorithms, tend to find the best solution (but not always guaranteed to have obtained) as quickly as possible. Test functions contain different features. For example single-mode and multi-mode, regular and irregular, separable and inseparable, etc. Modality defines the number of peaks in the problem landscape. These peaks form local minima and global minima locations. Unimodal Functions: These functions have one valley and one global minima location where the best solution resides. Such functions are considered to be easy to solve. Metaheuristic algorithms can be tested on these functions for evaluating local searchability. Multimodal Functions: The functions in this category maintain more than one solutions but true global best is one. Such functions have many local minima locations but one true global minima. Hence any metaheuristic algorithm needs to travel all the landscape to find the true global best solution. These functions are difficult to solve and are good for testing the global searchability of any algorithm

Table 2 Parameters setup for original SSA and ISSA

Algorithms	Population size	Dimension(n)	Maximum iteration	r_a	p_c	p_m	q	The ESM size
SSA	25	{10, 20, 30, 100, 500, 1000}	1000	1	0.7	0.1	–	–
ISSA	25	{10, 20, 30, 100, 500, 1000}	1000	1	0.7	0.1	10	4×N

(Hussain et al. 2017). In this paper, a variety of unimodal and multimodal benchmarking functions have been selected to evaluate the performance of ISSA's both exploitation and exploration capabilities. In experimental studies, nineteen different unimodal and multimodal benchmark functions, which are shown in Table 3 with ISSA and SSA, are solved separately.

Benchmark functions divide into two groups according to their type. Unimodal benchmark functions f1–f10, which have only one global optimum and can evaluate the exploitation capability of the investigated algorithms. Functions f11–f19 (multimodal), which include many local optima whose number increases exponentially with the problem size, become highly useful when the purpose is to evaluate the exploration capability of the investigated algorithms. These functions are obtained from various sources (Yu and Li 2015; Surjanovic and Bingham 2019; Cuevas et al. 2013). Long et al. (2017) group benchmark functions according to the shape of the objective function. According to the shape of objective functions, test functions can be categorized into six groups: (1) many local minima, (2) plate-shaped, (3) bowl-shaped, (4) valley-shaped, (5) steep ridge/drops, and (6) others. Table 4 shows benchmark functions which are divided according to the shape of the objective function.

For each benchmark function, two criteria are used to evaluate the performance of each algorithm to solve this function. These criteria include (1) average (Mean) and (2) standard deviation (SD). They are defined in Table 5. Where the *test* represents the total number of runs.

4.1 ISSA and SSA for low-scale dimensions

Original SSA and ISSA are tested in three different low-scale dimensions (n = {10, 20, 30}) for nineteen different benchmark functions. Each benchmark function is run ten times. Mean and standard deviation are done for obtained benchmark function results. Mean and standard deviation comparison results, which are obtained for unimodal benchmark functions by choosing dimension = {10, 20, 30}, population size = 25, and maximum iteration = 1000 values for original SSA and ISSA, are shown in Table 6. The results of ISSA's superior performance are marked in bold. According to the results, ISSA has shown better performance than SSA in every function except for f8 unimodal benchmark functions.

Wilcoxon signed-rank test aims to determine meaningful differences between the average of two samples (Acılar 2013). If it will be used for a comparison of two algorithms' outputs, the test practically evaluates the mutual behaviors of two algorithms. In this paper, ISSA and SSA algorithms are operated with ten times various unimodal and multimodal benchmark functions with the low-scale dimensions = {10, 20, 30}, population size = 25, and maximum iteration = 1000 equal parameter and ten trials are done in each dimension for each benchmark function and ten data sets are obtained. Whether there is a significant difference in this data set is searched by using the Wilcoxon signed-rank test. The results are shown in Table 9 for unimodal benchmark functions and Table 10 shows the multimodal benchmark. In Wilcoxon signed-rank test, alpha = 0.05 meaning level for num = 10 (num is data set number) (Acılar 2013). Two hypotheses are evaluated in the results. In the H_0 hypothesis, $M_{data1} = M_{data2}$ and in H_1 hypothesis, $M_{data1} \neq M_{data2}$. H_0 hypothesis is rejected in h = 1 values and it is called that there is a semantic difference between the algorithms. H_O hypothesis is accepted in h = 0 values and it is so-called that there is no semantic difference between the algorithms. p values of the hypothesis are calculated by using Matlab R2014a software. According to the results, the results obtained in f3 and f19 are

Table 3 Classical benchmark functions used in the experimental study

Type	Name	Function	Range	Dimension	f_{min}
Unimodal	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	Schwefel 1.2	$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_J \right)^2$	$[-100,100]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	Quartic	$f_3(x) = \sum_{i=1}^{n} i x_i^4$	$[-1.28,1.28]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	Powell	$f_4 = \sum_{i=1}^{n/4} \left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} - x_{4i}\right)^4$	$[-4,5]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$[-30,30]^n$	n = {10,20,30,100, 500,1000}	X* = (1,…,1); f(X*) = 0
	Dixon&Price	$f_6(x) = \left(x_1 - 1\right)^2 + \sum_{i=2}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$	$[-10,10]^n$	n = {10,20,30,100, 500,1000}	$x_i = 2^{-((2^{\wedge}i-2)/2^{\wedge}i)}$; f(X*) = 0
	Sum of Squares	$f_7(x) = \sum_{i=1}^{n} i x_i^2$	$[-10,10]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	Zakharov	$f_8(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$	$[-5,10]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	sum_of_different powers	$f_9(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$[-1,1]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0
	Rotated hyper-Ellipsoid	$f_{10}(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	$[-65536,65536]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); f(X*) = 0

Table 3 (continued)

Type	Name	Function	Range	Dimension	f_{min}
Multimodal	Michalewicz	$f_{11}(x) - \sum_{i=1}^{n} sin(x_i) sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	$[0,\pi]^n$	n = {10,20,30,100, 500,1000}	n = 10: $f(X^*) = -9{,}66015$; n = 20: $f(X^*) = -19{,}6370$; n = 30: $f(X^*) = -29{,}6309$
	Trid	$f_{12}(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	$[-n^2,n^2]^n$	n = {10,20,30,100, 500,1000}	i = 1,2,…,n); xi = i(n + 1 − i) $f(X^*) = -n(n+4)(n-1)/6$
	Levy	$f_{13}(x) = 0.1\left\{ \sin^2(3\pi x_i) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n)\right] \right\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4); u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & x_i < -a < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-10,10]^n$	n = {10,20,30,100, 500,1000}	X* = (1,…,1); $f(X^*) = 0$
	Schwefel	$f_{14}(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500,500]^n$	n = {10,20,30,100, 500,1000}	X* = (420.9687,…,420.9687) $f(X^*) = -418{,}9829xn$
	Rastrigin	$f_{15}(x) = \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i) + 10\right]$	$[-5.12,5.12]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); $f(X^*) = 0$
	Ackley	$f_{16}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + exp(1)$	$[-32,32]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); $f(X^*) = 0$
	Griewank	$f_{17(x)} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); $f(X^*) = 0$
	Salmon	$f_{18}(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	$[-100,100]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); $f(X^*) = 0$
	Noise	$f_{19}(x) = \sum_{i=1}^{n} ix_i^4 + random[0,1)$	$[-1.28,1.28]^n$	n = {10,20,30,100, 500,1000}	X* = (0,…,0); $f(X^*) = 0$

Table 4 Groups of benchmark functions (Long et al. 2017; Surjanovic and Bingham 2019)

I	Many local minima	*f13, f14, f15, f16, f17,f18, f19*
II	Bowl-shaped	*f1, f2, f3, f7, f9,f10,f12*
III	Valley-shaped	*f5, f6*
IV	Plate-shaped	*f8*
V	Steep Ridges/Drops	*f11*
VI	Others	*f4*

Table 5 The comparison criteria

Mean	$Mean = \frac{1}{n_{run}} \times \sum_{i=1}^{n_{run}} Fitness\_func_i$
Standard deviation	$Std.Dev. = \sqrt{\frac{1}{n_{run}} \sum_{i=1}^{n_{run}} \left(gbest_i - Mean\right)^2}$
Best	$Fitness\_func\_Best = min_{i=1n_{run}} Fitness\_func\_Best_i$
Worst	$Fitness\_func\_Worst = max_{i=1n_{run}} Fitness\_func\_Worst_i.$

equal and the Wilcoxon signed-rank test is not applied. According to the obtained results, there is a semantic difference between the results obtained from ISSA and SSA heuristic algorithms.

4.2 ISSA and SSA for middle and large-scale dimensions

Original SSA and ISSA are tested in three different large-scale dimensions = {100, 500 1000} for nineteen different benchmark functions. Each benchmark function is run ten times. Mean and standard deviation are done for obtained benchmark function results. Mean and standard deviation comparison results, which are obtained for unimodal benchmark functions by choosing dimension = {100, 500 1000}, population size = 25, and maximum iteration = 1000 values for SSA and ISSA, are shown in Table 11. The results of ISSA's superior performance are marked in bold. According to the results, ISSA has shown better performance than SSA in every function except for f2 and f9 unimodal benchmark functions.

In this paper, ISSA and SSA algorithms are operated with ten times various nineteen unimodal and multimodal benchmark functions with the large-scale dimensions = {100, 500, 1000}, population size = 25, and maximum iteration = 1000 equal parameter and ten trials are done in each dimension for each benchmark function and ten data sets are obtained. The results of the Wilcoxon signed-rank test are shown in Tables 14 and 15 for unimodal and multimodal benchmark functions, respectively. According to the results obtained, the results obtained in f3 and f19 are equal and the Wilcoxon signed-rank test is not applied. According to the obtained results, there is a semantic difference between the results obtained from ISSA and SSA.

Fitness value comparison results, which are obtained for nineteen unimodal and multimodal benchmark functions by choosing large-scale dimension = {100}, population size = 25, and maximum iteration = 1000 parameter values for original SSA and ISSA, are shown in Fig. 7.

Table 6 The comparison of SSA and ISSA with dimension (n) = {10, 20, 30}, the maximum number of iterations = 1000 in unimodal benchmark functions

F	n = 10				n = 20				n = 30			
	SSA		ISSA		SSA		ISSA		SSA		ISSA	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f1(x)	6.35E−05	3.10E−05	**1.28E−05**	**1.07E−05**	9.82E−02	4.39E−02	**1.94E−02**	**1.08E−02**	1.60E+00	5.70E−01	**2.90E−01**	**9.55E−02**
f2(x)	3.03E+01	4.82E+01	**2.45E+01**	**2.12E+01**	1.06E+02	1.83E+02	**2.22E+01**	**1.95E+01**	2.99E+02	3.38E+02	**2.52E+02**	**2.50E+02**
f3(x)	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**
f4(x)	3.68E−02	6.33E−02	**6.15E−03**	**8.00E−03**	2.26E+00	1.56E+00	**2.90E−01**	**1.90E−01**	4.95E+00	3.12E+00	**1.42E+00**	**1.09E+00**
f5(x)	6.46E+01	2.43E+01	**2.00E+01**	**1.28E+01**	6.81E+02	3.35E+02	**1.78E+02**	**6.24E+01**	2.83E+03	1.40E+03	**8.80E+02**	**2.14E+02**
f6(x)	5.43E+00	1.50E+01	**1.13E+00**	**1.54E+00**	8.52E+00	1.25E+01	**6.75E+00**	**4.33E+00**	1.64E+01	1.25E+01	**6.07E+00**	**2.06E+00**
f7(x)	1.04E−05	9.12E−06	**5.34E−06**	**8.06E−06**	1.35E−02	7.17E−03	**1.23E−03**	**5.33E−04**	2.22E−01	9.05E−02	**5.53E−02**	**3.48E−02**
f8(x)	9.04E+00	**2.15E+00**	**3.51E+01**	6.13E+00	**8.60E+01**	3.19E+01	1.76E+02	**2.38E+01**	**2.37E+02**	**3.76E+01**	3.87E+02	5.29E+01
f9(x)	6.10E−18	1.45E−17	**6.35E−19**	**1.68E−18**	9.54E−16	1.89E−15	**5.95E−18**	**1.12E−17**	7.13E−15	1.21E−14	**1.27E−15**	**2.48E−15**
f10(x)	3.74E+00	3.64E+00	**1.98E+00**	**2.40E+00**	3.42E+04	2.32E+04	**7.12E+03**	**4.42E+03**	1.32E+06	4.29E+05	**4.45E+05**	**1.34E+05**

Mean and standard deviation comparison results for multimodal benchmark functions by choosing low-scale dimensions (n = {10, 20, 30}), population size = 25, and maximum iteration = 1000 values for original SSA and ISSA, are shown in Table 7. The results of ISSA's superior performance are marked in bold. According to the mean and standard deviation results, ISSA has shown better performance than the original SSA in many multimodal benchmark functions

Table 7 The comparison of SSA and ISSA with dimension (n) = {10, 20, 30}, maximum number of iterations = 1000 in the multimodal benchmark functions

F	n = 10				n = 20				n = 30			
	SSA		ISSA		SSA		ISSA		SSA		ISSA	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f11(x)	−4.41E+00	4.64E−01	**−5.05E+00**	**2.28E−01**	−6.60E+00	6.19E−01	**−7.43E+00**	**1.50E−01**	−8.59E+00	5.16E−01	**−9.50E+00**	**2.48E−01**
f12(x)	**−6.70E+04**	4.06E+04	−7.16E+01	**1.68E+01**	**−3.94E+03**	1.89E+03	−2.83E+03	**1.43E+03**	**−7.11E+06**	3.69E+06	2.37E+03	**3.47E+02**
f13(x)	1.81E−06	1.34E−06	**3.46E−07**	**4.25E−07**	2.05E−03	1.13E−03	**1.65E−03**	**7.74E−04**	2.49E−02	1.41E−02	**5.21E−03**	**1.97E−03**
f14(x)	−1.66E+03	5.22E+02	**−1.78E+03**	**2.79E+02**	−2.36E+03	**7.24E+02**	**−3.23E+03**	1.03E+03	−2.66E+03	**3.38E+02**	**−3.69E+03**	3.73E+02
f15(x)	5.26E−01	4.75E−01	**1.99E−01**	**3.98E−01**	1.56E+01	3.07E+00	**4.90E+00**	**1.91E+00**	4.48E+01	4.97E+00	**1.19E+01**	**2.54E+00**
f16(x)	1.81E−02	1.74E−02	**2.14E−03**	**1.27E−03**	2.28E−01	8.54E−02	**1.36E−01**	**5.18E−02**	8.42E−01	2.49E−01	**2.29E−01**	**6.02E−02**
f17(x)	5.71E−02	2.67E−02	**2.87E−02**	**1.53E−02**	3.45E−01	1.08E−01	**1.15E−01**	**6.99E−02**	9.77E−01	**2.99E−02**	**5.83E−01**	1.51E−01
f18(x)	**2.50E+01**	**2.15E+00**	4.81E+01	4.01E+00	**5.62E+01**	**4.06E+00**	7.95E+01	5.00E+00	**8.84E+01**	**8.57E+00**	1.10E+02	1.03E+01
f19(x)	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**

Best and worst comparison results for unimodal and multimodal benchmark functions by choosing low-scale dimensions (n = {10, 20, 30}), population size = 25, and maximum iteration = 1000 values for SSA and ISSA, are shown in Table 8. The results of ISSA's superior performance are marked in bold

Table 8 The comparison of SSA and ISSA with dimension (n) = {10, 20, 30}, the maximum number of iterations = 1000 in the unimodal and multimodal benchmark functions

F	n = 10				n = 20				n = 30			
	SSA		ISSA		SSA		ISSA		SSA		ISSA	
	Best	Worst	Best	Worst	Best	Worst	Best	Worst	Best	Worst	Best	Worst
f1(x)	7.64E−06	1.23E−04	**3.02E−06**	4.12E−05	**2.49E−02**	1.72E−01	4.39E−03	3.49E−02	9.11E−01	4.01E+00	**1.37E−01**	4.25E−01
f2(x)	2.47E+00	1.68E+02	**2.36E+00**	7.37E+01	1.74E+00	6.32E+02	**1.60E+00**	4.85E+01	7.77E+01	1.23E+03	**4.31E+01**	8.33E+02
f3(x)	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f4(x)	3.84E−03	2.24E−01	**1.01E−04**	2.89E−02	5.03E−01	5.29E+00	**5.42E−02**	7.71E−01	1.84E+00	1.25E+01	**3.56E−01**	3.11E+00
f5(x)	2.32E+01	1.04E+02	**5.47E+00**	4.96E+01	2.49E+02	1.45E+03	**9.44E+01**	2.81E+02	1.15E+03	5.99E+03	**6.06E+02**	1.22E+03
f6(x)	1.24E−01	5.03E+01	**1.06E−01**	5.36E+00	**1.16E+00**	4.50E+01	2.74E+00	1.70E+01	7.79E+00	5.21E+01	**3.04E+00**	1.11E+01
f7(x)	8.18E−07	2.99E−05	**2.15E−07**	2.55E−05	4.52E−03	2.92E−02	**6.38E−04**	2.42E−03	1.41E−01	4.08E−01	**1.77E−02**	1.24E−01
f8(x)	**5.81E+00**	1.27E+01	2.24E+01	4.36E+01	3.27E+01	1.42E+02	**1.42E+01**	2.15E+02	1.63E+02	2.99E+02	**2.51E+02**	4.60E+02
f9(x)	5.47E−22	4.19E−19	**6.58E−23**	5.66E−18	1.57E−17	6.35E−15	**8.26E−21**	3.63E−17	8.74E−17	4.20E−14	**2.51E−17**	8.65E−15
f10(x)	4.73E−01	1.33E+01	**6.22E−02**	7.80E+00	1.17E+04	9.30E+04	**3.23E+03**	1.54E+04	4.44E+05	1.91E+06	**1.82E+05**	5.79E+05
f11(x)	−5.27E+00	−3.81E+00	**−5.40E+00**	−4.71E+00	−7.71E+00	−5.84E+00	**−7.82E+00**	−7.28E+00	−9.39E+00	−7.94E+00	**−9.85E+00**	−8.94E+00
f12(x)	**−1.68E+05**	−2.46E+04	−2.84E+03	−1.72E+02	**−7.00E+03**	−1.78E+03	−5.44E+03	−9.80E+02	**−1.71E+07**	−2.62E+06	1.69E+03	2.83E+03
f13(x)	2.25E−07	4.39E−06	**7.33E−09**	1.11E−06	5.80E−04	4.57E−03	**5.71E−04**	3.00E−03	1.02E−02	6.15E−02	**2.41E−03**	8.24E−03
f14(x)	−2.80E+03	−8.57E+02	**−2.89E+03**	−1.32E+03	−3.79E+03	−1.75E+03	**−4.91E+03**	−1.85E+03	−3.15E+03	−1.97E+03	**−3.33E+03**	−2.15E+03
f15(x)	6.37E−02	1.37E+00	**4.53E−06**	9.95E−01	1.20E+01	2.23E+01	**2.09E+00**	8.22E+00	3.43E+01	5.16E+01	**7.12E+00**	1.51E+01
f16(x)	5.45E−03	6.69E−02	**8.51E−04**	5.20E−03	7.54E−02	3.77E−01	**7.50E−02**	2.61E−01	3.47E−01	1.11E+00	**1.40E−01**	3.34E−01
f17(x)	2.29E−02	1.01E−01	**1.50E−03**	5.80E−02	1.56E−01	4.98E−01	**3.61E−02**	3.00E−01	9.31E−01	1.02E+00	**3.05E−01**	8.75E−01
f18(x)	**1.80E+00**	4.34E+01	2.95E+00	3.74E+01	**5.17E+00**	9.64E+01	6.16E+00	9.45E+01	**8.84E+01**	3.35E+02	9.68E+01	1.10E+02
f19(x)	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00

Table 9 Wilcoxon signed-rank test on SSA and ISSA in dimension (n) = {10, 20, 30}, maximum number of iterations = 1000, population size = 25 in unimodal benchmark functions

F	n = 10				n = 20				n = 30			
	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)
f1(x)	4.40E−04	1	4.40E−04	1	7.69E−04	1	7.69E−04	1	1.83E−04	1	1.83E−04	1
f2(x)	0.6776	**0**	0.6776	**0**	0.1620	**0**	0.1620	**0**	0.5205	**0**	0.5205	**0**
f3(x)	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**
f4(x)	0.0013	1	0.0013	1	3.30E−04	1	3.30E−04	1	7.69E−04	1	7.69E−04	1
f5(x)	7.69E−04	1	7.69E−04	1	0.0013	1	0.0013	1	4.40E−04	1	4.40E−04	1
f6(x)	0.0376	1	0.0376	1	0.0890	**0**	0.0890	**0**	1.83E−04	1	1.83E−04	1
f7(x)	1.83E−04	1	1.83E−04	1	5.83E−04	1	5.83E−04	1	1.83E−04	1	1.83E−04	1
f8(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	2.46E−04	1	2.46E−04	1
f9(x)	0.0073	1	0.0073	1	2.46E−04	1	2.46E−04	1	0.9097	**0**	0.9097	**0**
f10(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	5.83E−04	1	5.83E−04	1

Table 10 Wilcoxon signed-rank test on SSA and ISSA in dimension (n) = {10, 20, 30}, maximum number of iterations = 1000, population size = 25 in multimodal benchmark functions

F	n = 10				n = 20				n = 30			
	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)
f11(x)	0.0073	1	0.0073	1	1.83E−04	1	1.83E−04	1	3.30E−04	1	3.30E−04	1
f12(x)	0.0452	1	0.0452	1	0.5205	**0**	0.5205	**0**	0.0010	1	0.0010	1
f13(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f14(x)	0.7337	**0**	0.7337	**0**	0.2413	**0**	0.2413	**0**	0.5205	**0**	0.5205	**0**
f15(x)	2.46E−04	1	2.46E−04	1	0.0073	1	0.0073	1	2.46E−04	1	2.46E−04	1
f16(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f17(x)	0.0013	1	0.0013	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f18(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f19(x)	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**

Table 11 The comparison of SSA and ISSA with dimension (n) = {100, 500, 1000}, maximum number of iterations = 1000 in unimodal benchmark functions

F	n = 100				n = 500				n = 1000			
	SSA		ISSA		SSA		ISSA		SSA		ISSA	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f1(x)	5.98E+02	1.24E+02	**3.05E+02**	**4.75E+01**	8.22E+04	5.20E+03	**5.73E+04**	**3.79E+03**	3.19E+05	1.50E+04	**2.28E+05**	**1.04E+04**
f2(x)	**4.03E+04**	**1.27E+04**	5.28E+04	2.81E+04	**4.03E+07**	6.04E+06	4.23E+07	**5.60E+06**	**3.53E+08**	2.54E+07	3.80E+08	**1.99E+07**
f3(x)	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**
f4(x)	1.98E+02	4.97E+01	**5.52E+01**	**1.47E+01**	1.18E+04	1.41E+03	**5.92E+03**	**4.74E+02**	4.44E+04	2.27E+03	**2.41E+04**	**1.34E+03**
f5(x)	5.95E+04	1.33E+04	**1.91E+04**	**4.41E+03**	5.05E+07	3.17E+06	**1.82E+07**	**2.22E+06**	2.72E+08	8.13E+06	**1.04E+08**	**3.61E+06**
f6(x)	1.55E+03	4.56E+02	**7.99E+02**	**2.81E+02**	5.34E+06	7.33E+05	**2.07E+06**	**3.45E+05**	5.69E+07	4.16E+06	**2.20E+07**	**1.51E+06**
f7(x)	2.52E+02	4.44E+01	**1.13E+02**	**1.71E+01**	1.79E+05	9.40E+03	**1.23E+05**	**8.32E+03**	1.39E+06	**4.89E+04**	**9.85E+05**	7.51E+04
f8(x)	2.51E+03	**2.67E+02**	**2.19E+03**	4.01E+02	1.53E+20	**2.39E+19**	**1.17E+20**	2.50E+19	6.17E+22	1.33E+22	**5.22E+22**	**7.21E+21**
f9(x)	**3.91E−13**	**6.49E−13**	1.18E−07	9.23E−08	1.88E−04	4.94E−04	**1.10E−05**	**9.98E−06**	1.10E+00	3.71E−01	**3.38E−04**	**2.91E−04**
f10(x)	6.68E+09	1.13E+09	**4.29E+09**	**7.51E+08**	6.67E+12	3.41E+11	**4.29E+12**	**2.06E+11**	6.07E+14	**5.46E+12**	**2.01E+14**	3.42E+13

Mean and standard deviation comparison results for multimodal benchmark functions by choosing large-scale dimensions (n = {100, 500, 1000}), population size = 25, and maximum iteration = 1000 values for SSA and ISSA, are shown in Table 12. The results of ISSA's superior performance are marked in bold. According to the mean and standard deviation results, ISSA has shown better performance than SSA in many multimodal benchmark functions

Table 12 The comparison of SSA and ISSA with dimension (n) = {100, 500, 1000}, the maximum number of iterations = 1000 in the multimodal benchmark functions

F	n = 100				n = 500				n = 1000			
	SSA		ISSA		SSA		ISSA		SSA		ISSA	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f11(x)	−2.12E+01	1.64E+00	**−2.26E+01**	**1.06E+00**	−7.60E+01	**1.35E+00**	**−8.44E+01**	1.82E+00	−1.41E+02	**2.36E+00**	**−1.57E+02**	3.02E+00
f12(x)	**−1.53E+09**	7.11E+08	−8.96E+08	**5.81E+08**	3.39E+11	**2.76E+11**	**−3.22E+11**	1.83E+12	6.44E+13	1.33E+13	**1.69E+13**	**1.30E+12**
f13(x)	4.90E+00	6.06E−01	**3.03E+00**	**5.42E−01**	4.41E+02	3.98E+01	**2.56E+02**	**3.15E+01**	1.52E+03	7.06E+01	**1.03E+03**	**8.22E+01**
f14(x)	−4.63E+03	**3.65E+02**	**−5.21E+03**	9.93E+02	−1.07E+04	2.20E+03	**−1.11E+04**	**1.19E+03**	−1.39E+04	3.13E+03	**−1.46E+04**	**2.14E+03**
f15(x)	5.30E+02	2.09E+01	**4.09E+02**	**2.05E+01**	4.64E+03	7.80E+01	**4.14E+03**	**2.19E+01**	1.,03E+04	1.11E+02	**9.29E+03**	**7.24E+01**
f16(x)	4.67E+00	2.87E−01	**3.69E+00**	**1.27E−01**	1.30E+01	2.41E−01	**1.11E+01**	**1.93E−01**	1.54E+01	1.34E−01	**1.37E+01**	**2.58E−02**
f17(x)	6.20E+00	1.01E+00	**3.69E+00**	**3.89E−01**	7.67E+02	4.14E+01	**5.34E+02**	**2.73E+01**	2.84E+03	9.79E+01	**1.93E+03**	**7.57E+01**
f18(x)	**2.23E+02**	1.10E+01	2.26E+02	**3.63E+00**	6.02E+02	1.29E+01	**5.85E+02**	**1.09E+01**	8.77E+02	**1.22E+01**	**8.57E+02**	2.06E+01
f19(x)	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**

Best and worst comparison results for unimodal and multimodal benchmark functions by choosing large-scale dimensions (n = {100, 500, 1000}), population size = 25, and maximum iteration = 1000 values for SSA and ISSA, are shown in Table 13. The results of ISSA's superior performance are marked in bold. According to the results, ISSA has a better performance in most cases

Table 13 The comparison of SSA and ISSA with dimension (n) = {100, 500, 1000}, the maximum number of iterations = 1000 in the unimodal and multimodal benchmark functions

F	n = 100				n = 500				n = 1000			
	SSA		ISSA		SSA		ISSA		SSA		ISSA	
	Best	Worst	Best	Worst	Best	Worst	Best	Worst	Best	Worst	Best	Worst
f1(x)	3.77E+02	8.13E+02	**2.34E+02**	3.85E+02	7.31E+04	9.09E+04	**5.26E+04**	6.57E+04	2.99E+05	3.40E+05	**2.14E+04**	2.52E+05
f2(x)	1.95E+04	6.09E+04	**1.52E+04**	1.20E+05	**2.82E+07**	4.76E+07	3.68E+07	4.79E+07	3.60E+08	3.96E+08	**3.15E+08**	4.00E+08
f3(x)	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f4(x)	9.39E+01	2.58E+02	**3.16E+01**	7.57E+01	1.02E+04	1.50E+02	**5.26E+03**	6.80E+03	4.10E+04	4.91E+04	**1.92E+04**	2ç37E+04
f5(x)	4.39E+04	8.18E+04	**1.41E+04**	2.92E+04	4.68E+07	5.76E+07	**1.44E+07**	2.15E+07	2.59E+08	2.83E+08	**8.62E+07**	9.88E+07
f6(x)	9.60E+02	2.70E+03	**5.22E+02**	1.58E+03	4.20E+06	6.68E+06	**1.44E+06**	2.89E+06	4.78E+07	6.41E+07	**1.94E+07**	2.47E+07
f7(x)	1.98E+02	3.28E+02	**8.24E+01**	1.42E+02	1.59E+05	1.90E+05	**1.11E+05**	1.39E+05	1.32E+06	1.47E+06	**8.77E+05**	1.12E+06
f8(x)	2.16E+03	2.98E+03	**1.45E+03**	2.66E+03	1.20E+20	2.00E+20	**7.16E+19**	1.44E+20	3.76E+22	8.27E+22	**3.56E+22**	6.49E+22
f9(x)	**4.52E−15**	2.20E−12	1.32E−08	2.99E−07	4.35E−06	1.67E−03	**1.18E−06**	3.16E−05	6.66E−01	1.85E+00	**2.40E−05**	1.02E−03
f10(x)	5.11E+09	8.47E+09	**3.20E+09**	5.71E+09	6.33E+12	7.02E+12	**4.00E+12**	4.74E+12	6.01E+14	6.12E+14	**1.66E+14**	2.35E+14
f11(x)	−2.42E+01	−1.83E+01	**−2.55E+01**	−2.13E+01	−7.81E+01	−7.38E+01	**−8.75E+01**	−8.24E+01	−1.43E+02	−1.36E+02	**−1.63E+02**	−1.53E+02
f12(x)	**−2.75E+09**	−7.39E+08	−2.34E+09	−2.87E+08	−4.88E+11	4.64E+11	**−5.81E+12**	3.10E+11	4.99E+13	9.75E+13	**1.49E+13**	1.92E+13
f13(x)	3.99E+00	5.92E+00	**2.27E+00**	4.22E+00	3.94E+02	4.94E+02	**2.19E+02**	3.29E+02	**1.43E+03**	1.65E+03	9.26E+03	1.17E+03
f14(x)	−5.13E+03	−3.94E+03	**−7.56E+03**	−5.49E+03	**−1.54E+04**	−7.39E+03	−1.19E+04	−1.04E+04	−2.07E+04	−9.88E+03	**−2.20E+04**	−1.18E+04
f15(x)	4.93E+02	5.68E+02	**4.02E+02**	4.70E+02	4.47E+03	4.74E+03	**4.13E+03**	4.21E+03	1.01E+04	1.04E+04	**9.26E+03**	9.45E+03
f16(x)	4.06E+00	5.12E+00	**3.50E+00**	3.97E+00	1.27E+01	1.36E+01	**1.11E+01**	1.17E+01	1.51E+01	1.56E+01	**1.37E+01**	1.37E+01
f17(x)	5.07E+00	8.19E+00	**3.22E+00**	4.37E+00	6.91E+02	8.64E+02	**4.91E+02**	5.80E+02	2.67E+03	3.02E+03	**1.88E+03**	2.12E+03
f18(x)	**2.09E+02**	2.45E+02	2.25E+02	2.37E+02	5.89E+02	6.32E+02	**5.80E+02**	6.09E+02	8.54E+02	8.89E+02	**8.11E+02**	8.54E+02
f19(x)	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00

Table 14 Wilcoxon signed-rank test on SSA and ISSA in dimension (n) = {100, 500, 1000}, maximum number of iterations = 1000, population size = 25 in unimodal benchmark functions

F	n = 100				n = 500				n = 1000			
	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)
f1(x)	3.30E−04	1	3.30E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f2(x)	0.3847	**0**	0.3847	**0**	0.4659	**0**	0.4659	**0**	0.2201	**0**	0.2201	**0**
f3(x)	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**
f4(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	8.75E−05	1	8.75E−05	1
f5(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.73E−04	1	1.73E−04	1
f6(x)	0.0010	1	0.0010	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f7(x)	1.83E−04	1	1.83E−04	1	1.50E−04	1	1.49E−04	1	1.83E−04	1	1.83E−04	1
f8(x)	0.1041	**0**	0.1041	**0**	0.0173	1	0.0173	1	0.1212	**0**	0.1212	**0**
f9(x)	1.83E−04	1	1.83E−04	1	0.1859	**0**	0.1859	**0**	1.83E−04	1	1.83E−04	1
f10(x)	3.30E−04	1	3.30E−04	1	1.45E−04	1	1.45E−04	1	8.45E−05	1	8.45E−05	1

Table 15 Wilcoxon signed-rank test on SSA and ISSA in dimension (n) = {100, 500, 1000}, maximum number of iterations = 1000, population size = 25 in multimodal benchmark functions

F	n = 100				n = 500				n = 1000			
	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)	ISSA(p)	(h)	SSA(p)	(h)
f11(x)	0.0539	**0**	0.0539	**0**	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f12(x)	0.0312	1	0.0312	1	0.0022	1	0.0022	1	1.83E−04	1	1.83E−04	1
f13(x)	3.30E−04	1	3.30E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1
f14(x)	1.11E−04	1	1.11E−04	1	0.2088	**0**	0.2088	**0**	0.4274	**0**	0.4274	**0**
f15(x)	8.75E−05	1	8.75E−05	1	8.75E−05	1	8.75E−05	1	1.11E−04	1	1.11E−04	1
f16(x)	1.83E−04	1	1.83E−04	1	8.75E−05	1	8.75E−05	1	1.10E−04	1	1.10E−04	1
f17(x)	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.83E−04	1	1.50E−04	1	1.49E−04	1
f18(x)	0.1042	**0**	0.1042	**0**	0.0056	1	0.0056	1	1.21E−04	1	1.21E−04	1
f19(x)	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**	NaN	**0**

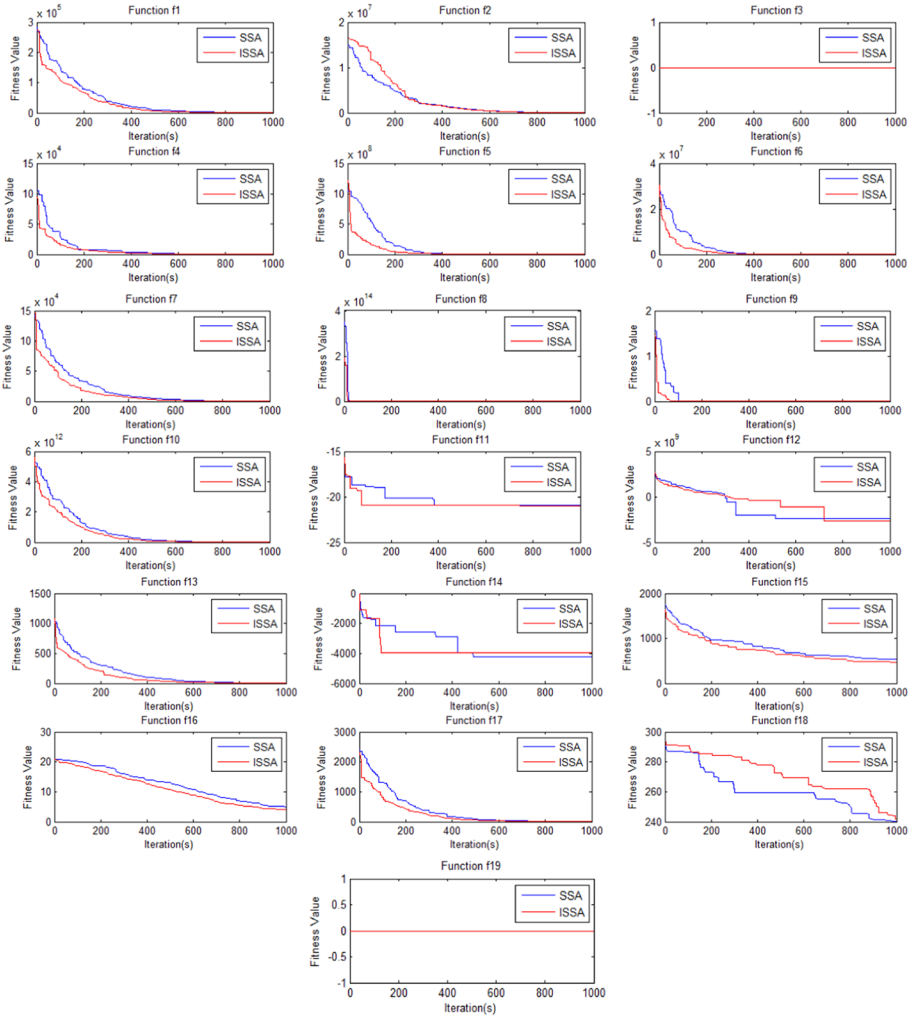

Fig. 7 Convergence graphics for SSA and ISSA for benchmark functions f1-f19 on dimension = 100

Fitness value comparison results, which are obtained for nineteen unimodal and multimodal benchmark functions by choosing large-scale dimension = {500}, population size = 25, and maximum iteration = 1000 parameter values for SSA and ISSA, are shown in Fig. 8.

Fitness value comparison results, which are obtained for unimodal and multimodal benchmark functions by choosing large-scale dimension = {1000}, population size = 25, and maximum iteration = 1000 parameter values for SSA and ISSA, are shown in Fig. 9.

4.3 Comparison of ISSA with other metaheuristics algorithms

Case Study 1: In addition to the SSA algorithm, the ISSA algorithm is compared to well-known heuristic algorithms and the new algorithms selected from the literature.

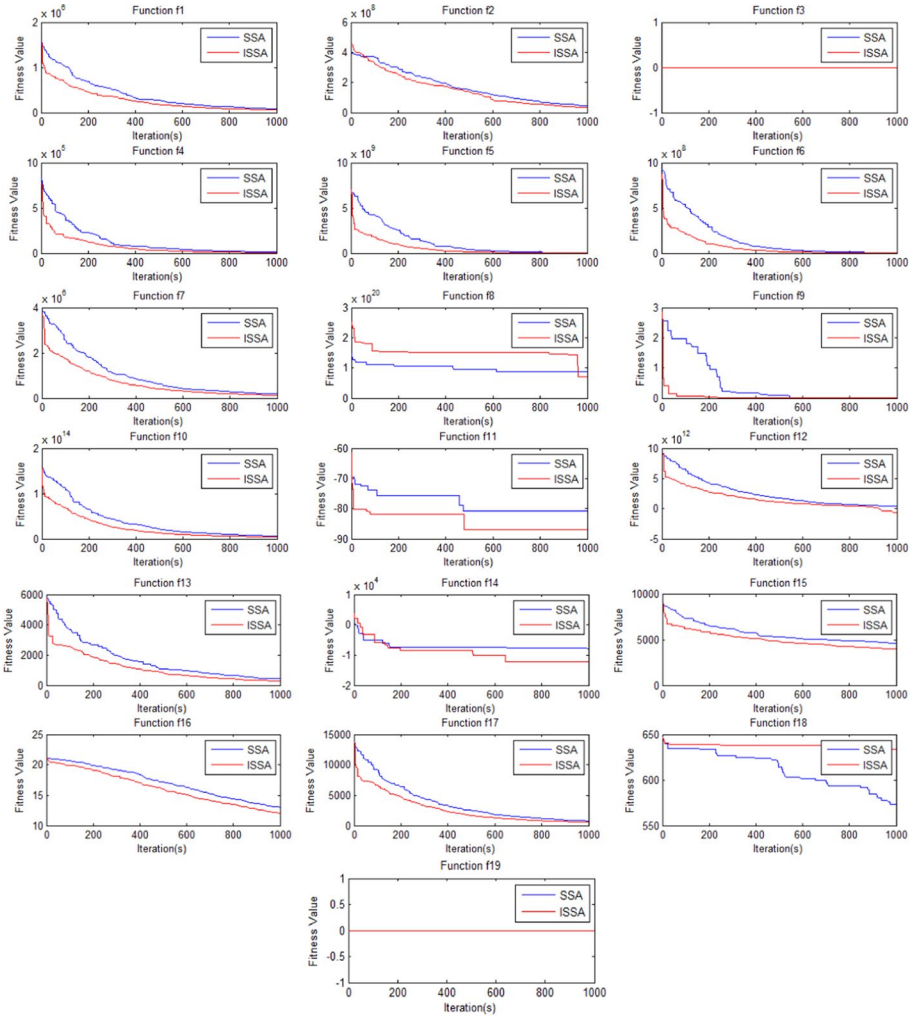

Fig. 8 Convergence graphics for SSA and ISSA for benchmark functions f1-f19 on dimension = 500

Well-known heuristic algorithms of the literature and recently developed algorithms HS, GA, DE, FF, CLPSO, GL-25, CMA-ES, CoDE, ABC, PSO, DE, SalpSA, TLBO, CMA-ES, GSA, FA, SFLA, GWO, IGWO, NGWO, and MGWO which are compared with ISSA, are shown in Table 16.

Case Study 2: ISSA is also compared with four state-of-the-art algorithms: CLPSO, GL-25, CMA-ES, and CoDE. f15, f18, and f19 multimodal test functions are used to investigate the performance of ISSA and other four state-of-the-art algorithms. For clarity, the results of the best algorithms are marked in bold. The comparison results are shown in Table 18. ISSA finds better results for test functions in dimension = 30.

Case Study 3: ABC, PSO, DE, SalpSA, CMA-ES, and ISSA algorithms are compared according to two different criteria in large-scale dimensions (dimension = {1000}). These are mean which represents average value and SD which represents

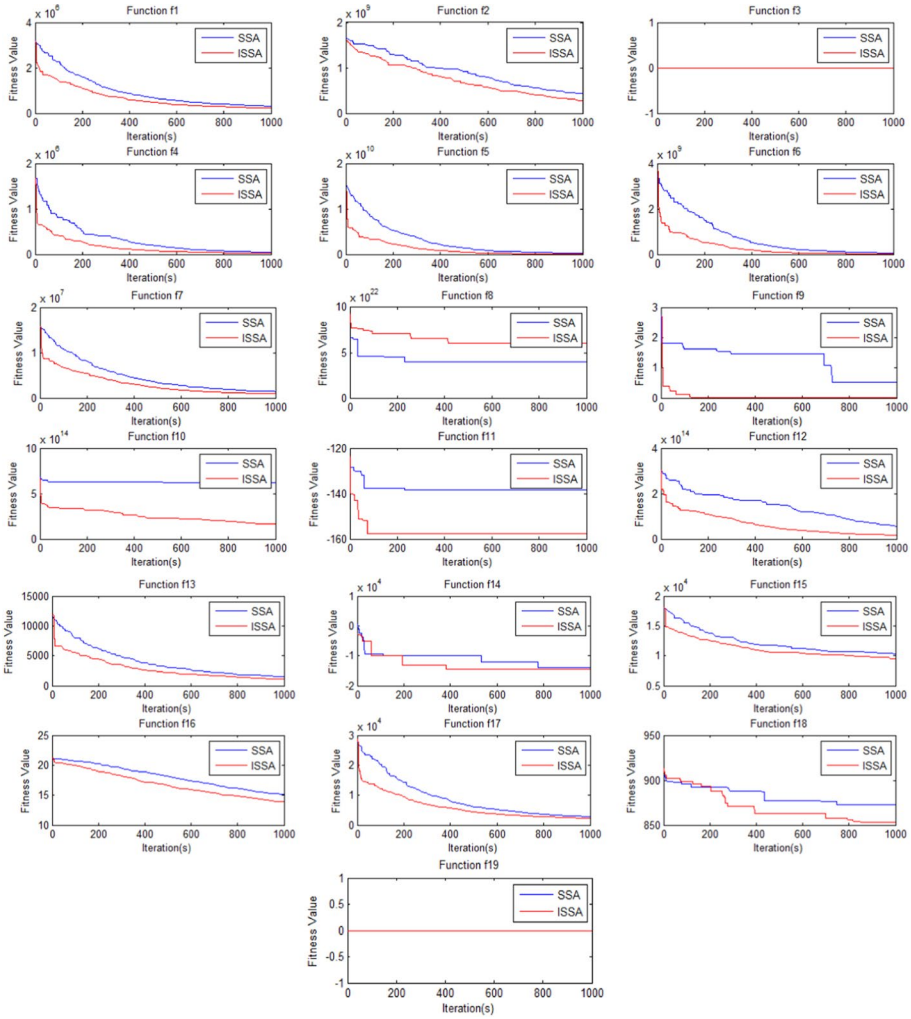

Fig. 9 Convergence graphics for SSA and ISSA for benchmark functions f1-f19 on dimension = 1000

standard deviation. Parameter sets, which are used for comparing algorithms, are shown in Table 19. The comparison results are shown in Table 20. For clarity, the results of the best algorithms are marked in bold. According to the comparison results, ISSA has displayed an extremely good performance in all selected benchmark functions except for f9 and f15 for the comparison.

Case Study 4: GSA, FA, SFLA, and ISSA algorithms are compared according to two different criteria in large-scale dimensions (dimension = {500}); mean and SD For the compared algorithms, the population size value is determined as 50, the maximum iteration value is determined as 1000, and the dimension value is determined as 500 equally. Parameter sets, which are used for comparing algorithms, are shown in Table 21. The comparison results are shown in Table 22. For clarity, the results of the best algorithms are marked in bold. According to the comparison results, ISSA has

Table 16 Keys commonly used for comparative algorithms

Key	Method Name	References
HS	Harmony Search	Shayanfar and Gharehchopogh (2018)
GA	Genetic Algorithm	Shayanfar and Gharehchopogh (2018)
DE	Differential Evolution	Shayanfar and Gharehchopogh (2018)
FF	Farmland Fertility	Shayanfar and Gharehchopogh (2018)
CLPSO	Comprehensive Learning Particle Swarm Optimizer	Liang et al. (2006)
GL-25	Global and local real-coded genetic algorithms	Garcia-Martinez et al. (2008)
CMA-ES	Covariance Matrix Adaptation Evolution Strategy	Hansen and Ostermeier (2001)
CoDE	Composite DE	Wang et al. (2011)
ABC	Artificial Bee Colony	Liang et al. (2006)
PSO	Particle Swarm Optimization	Liang et al. (2006)
DE	Differential Evolution	Liang et al. (2006)
SalpSA	Salp Swarm Algorithm	Liang et al. (2006)
CMA-ES	Covariance Matrix Adaptation Evolution Strategy	Liang et al. (2006)
GSA	Gravitational Search Algorithm	Liu et al. (2019)
FA	Firefly Algorithm	Liu et al. (2019)
SFLA	Shuffled Frog-Leaping Algorithm	Liu et al. (2019)
GWO	Grey Wolf Optimizer	Mirjalili et al. (2014)
IGWO	Inspired Grey Wolf Optimizer	Long et al. (2018)
NGWO	GWO with Nonlinear Control Parameter	Long (2016)
MGWO	Modified Grey Wolf Optimizer	Mittal et al. (2016)
ISSA	Improved Social Spider Algorithm	

HS, GA, DE, FF, and ISSA algorithms are compared according to best criteria which represents the best result in low-scale dimension = {10, 20, 30}. For the compared algorithms, population size is determined as 30, maximum iteration value is determined as 1500, and dimension value is determined as {10, 20, 30} equally. The comparison results are shown in Table 17. According to the comparison results, ISSA has displayed an extremely good performance in f1, f6, f8, f15, and f17 benchmark functions

displayed an extremely good performance in all selected benchmark functions for the comparison.

Case Study 5: GWO, IGWO, NGWO, MGWO, and ISSA algorithms are compared according to mean and SD in large-scale dimensions (dimension = {100, 500, 1000}). For the compared algorithms, the population size value is determined as 30, maximum iteration value is determined as 500, and dimension value is determined as {100, 500, 1000} equally. Parameter sets, which are used for comparing algorithms, are shown in Table 23. The comparison results are shown in Table 24. According to the results, ISSA has performed well in f19 except for f16 and f17.

f2, f10, f12, and f14 benchmark functions are also used as a test function for SSA and ISSA and obtained results are compared with SSA and ISSA. But, related performance comparisons could not be performed and are not mentioned in this paper as there are not many papers that are studied with the benchmark functions in the literature.

Table 17 ISSA and its comparison with comparative algorithms in FF, HS, GA, and DE (Dimension = {10, 20, 30})

Function	Dimension	HS	GA	DE	FF	ISSA
f1	10-Best	–	–	–	**6.3576E−013**	1.90E−11
	20-Best	–	–	–	0.1798E−01	**4.77E−06**
	30-Best	5.919E+03	5.517E+00	2.481E+01	2.6571E+00	**7.42E−04**
f6	10-Best	–	–	–	0.49062E+00	**1.15E−03**
	20-Best	–	–	–	9.2627E+00	**1.80E−01**
	30-Best	4.057E+04	1.207E+01	2.650E+01	4.5644E+01	**1.03E+00**
f8	10-Best	–	–	–	0.4757E+00	**5.71E−03**
	20-Best	–	–	–	1.2038E+01	**2.83E+00**
	30-Best	2.960E+02	1.122E+01	1.414E+02	3.60277E+01	**3.50E+01**
f11	10-Best	–	–	–	**−8.9921E+00**	−5.66E+00
	20-Best	–	–	–	**−1.11411E+01**	−7.78E+00
	30-Best	−1.477E+01	−2.473E+01	−1.259E+01	**−2.79383E+01**	−1.03E+01
f15	10-Best	–	–	–	1.6394 E+00	**4.88E−07**
	20-Best	–	–	–	7.4641E+00	**1.46E+00**
	30-Best	1.330E+02	2.994E+01	2.998E+01	3.1277E+01	**1.35E+01**
f16	10-Best	–	–	–	**7.9936E−15**	1.52E−06
	20-Best	–	–	–	**1.5377E−09**	6.88E−04
	30-Best	1.338E+01	2.595E+00	2.302E+00	**9.582E−05**	6.14E−03
f17	10-Best	–	–	–	4.2780E−3	**8.99E−05**
	20-Best	–	–	–	2.5874E−2	**7.56E−04**
	30-Best	4.375E+01	1.080E−01	9.989E−03	1.03149E−01	**4.01E−03**

Table 18 ISSA and its comparison with comparative algorithms in CLPSO, GL-25, CMA-ES, and CoDE (Dimension = 30)

F		CLPSO	GL-25	CMA-ES	CoDE	ISSA
f15	SD	1.53E+01	6.45E+01	1.16E+01	1.16E+01	**2.54E+00**
f18	SD	**2.15E−01**	4.88E+00	7.60E−01	3.27E−01	5.39E+00
f19	SD	1.73E+03	4.25E+02	2.16E+06	1.38E−02	**0.00E+00**

Table 19 Parameters settings of comparative algorithms in ABC, PSO, DE, SalpSA, CMA-ES, and ISSA

Algorithms	Population size	Maximum iteration	Dimension
ABC	100	1000	1000
PSO	100	1000	1000
DE	100	1000	1000
SalpSA	30	1000	1000
CMA-ES	240	500	1000
ISSA	30	1000	1000

Table 20 ISSA and its comparison with comparative algorithms in ABC, PSO, DE, SalpSA, and CMA-ES. (Dimension = 1000)

F	Features	ABC	PSO	DE	SalpSA	CMA-ES	ISSA
f1	Mean	3.08E+06	3.52E+04	7.34E+05	1.92E+05	9.75E+03	**3.11E+03**
	SD	4.06E+04	4.83E+03	2.49E+04	1.02E+04	6.64E+02	**1.25E+02**
f3	Mean	2.38E+05	7.54E+01	3.74E+04	1.08E+03	5.33E+01	**0.00E+00**
	SD	3.80E+03	1.79E+01	1.45E+03	8.34E+01	1.02E+01	**0.00E+00**
f5	Mean	1.46E+10	7.00E+06	3.18E+09	7.85E+07	8.03E+05	**9.21E+04**
	SD	2.64E+08	9.97E+05	1.78E+08	6.39E+07	1.31E+05	**9.81E+03**
f7	Mean	1.52E+07	1.43E+05	2.51E+06	8.56E+05	1.40E+05	**8.07E+03**
	SD	2.08E+05	1.51E+04	6.44E+04	6.48E+05	7.42E+03	**4.43E+03**
f9	Mean	2.29E+00	**6.73E−36**	2.43E+00	5.69E−07	5.27E+00	5.35E−06
	SD	3.55E−01	**1.21E−35**	4.39E−01	2.06E−06	1.74E+00	5.35E−06
f13	Mean	1.87E+05	3.54E+03	9.38E+04	9.34E+03	2.14E+04	**5.03E+02**
	SD	4.61E+03	4.63E+02	1.12E+03	1.56E+03	1.44E+03	**9.25E+01**
f15	Mean	1.77E+04	**3.57E+03**	1.35E+04	6.32E+03	1.01E+04	8.50E+03
	SD	1.13E+02	1.25E+02	**8.59E+01**	1.41E+02	1.30E+02	3.87E+02
f16	Mean	2.11E+01	9.74E+00	1.87E+01	1.43E+01	5.65E+00	**2.38E+00**
	SD	6.61E−03	1.54E−01	5.55E−02	1.24E−01	8.38E−02	**4.44E−16**
f17	Mean	2.77E+04	2.89E+02	6.44E+03	1.67E+03	8.68E+01	**2.01E+01**
	SD	6.32E+02	2.95E+01	1.33E+02	4.52E+01	5.46E+00	**5.25E−01**
f18	Mean	1.77E+02	2.64E+01	9.91E+01	5.29E+01	**1.42E+01**	7.30E+02
	SD	1.31E+00	5.81E−01	9.80E−01	7.44E−01	8.23E−01	**1.09E−01**
f19	Mean	2.38E+05	1.28E+02	3.65E+04	1.23E+03	5.73E+01	**0.00E+00**
	SD	2.60E+03	3.60E+01	6.21E+02	9.34E+01	4.09E+00	**0.00E+00**

Table 21 Parameters settings of comparative algorithms in GSA, FA, SFLA, and ISSA

Algorithms	Population size	Maximum iteration	Dimension
GSA	50	1000	500
FA	50	1000	500
SFLA	50	1000	500
ISSA	50	1000	500

5 Conclusion

Original Social Spider Algorithm (SSA) is a heuristic algorithm that is created by imitating spider behaviors in nature. SSA organizes the search space of optimization problems as a high dimensional spider web. SSA can easily be adapted to real-world problems and any heuristic algorithm. In this paper, the existing SSA optimization algorithm in the literature is improved with two new techniques; spider blasting and explorer spider memory. It is named as improved SSA (ISSA). In the paper, steps which are developed in ISSA, are explained details. To evaluate the performance of ISSA, we have adopted a set of benchmark functions that cover a large variety of different optimization problem types. Each benchmark function is performed in low, middle, and large-scale

Table 22 ISSA and its comparison with comparative algorithms in GSA, FA, and SFLA (Dimension = 500)

F	Features	GSA	FA	SFLA	ISSA
f1	Mean	2.28E+04	1.96E+04	3.39E+05	**3.03E+04**
	SD	1.16E+03	8.11E+03	3.13E+03	**9.04E+01**
f4	Mean	2.37E+03	4.79E+04	5.32E+05	**3.02E+03**
	SD	3.50E+02	2.95E+04	1.98E+04	**1.01E+02**
f13	Mean	9.68E+02	3.43E+03	1.48E+04	**1.62E+02**
	SD	1.66E+02	1.80E+03	2.43E+02	**9.54E+00**
f15	Mean	**1.53E+03**	4.15E+03	8.17E+03	4.21E+03
	SD	7.83E+01	5.10E+02	8.34E+01	**3.08E+01**
f16	Mean	6.95E+00	1.95E+01	2.10E+01	**6.72E+00**
	SD	6.72E−02	3.93E−01	1.22E−02	**2.49E−02**
f17	Mean	4.40E+03	1.75E+03	3.43E+02	**3.36E+01**
	SD	7.77E+01	7.94E+02	4.37E+00	**3.22E+00**
f19	Mean	1.43E+02	4.38E+03	4.67E+04	**0.00E+00**
	SD	1.58E+01	2.03E+03	9.73E+02	**0.00E+00**

Table 23 Parameters settings of comparative algorithms in GWO, IGWO, NGWO, MGWO, AWGWO, and ISSA

Algorithms	Population size	Maximum iteration	Dimension
GWO	30	500	{100, 500, 1000}
IGWO	30	500	{100, 500, 1000}
NGWO	30	500	{100, 500, 1000}
MGWO	30	500	{100, 500, 1000}
ISSA	30	500	{100, 500, 1000}

different dimensions (dimension = {10, 20, 30, 100, 500, 1000}) and each benchmark function is run ten times. The mean and standard deviation values of each benchmark function are calculated. The performances of SSA and ISSA are tested in chosen benchmark functions and the results are compared. According to the obtained results, ISSA has displayed superior performance in most of the chosen benchmark functions. This proves that new techniques, which are added to SSA, enhances SSA performance. Well-known heuristic algorithms in the literature, which include ABC, PSO, DE, SalpSA, TLBO, CMA-ES, GSA, FA, SFLA, GWO, IGWO, NGWO, and MGWO are compared with ISSA in middle and large-scale dimensions. Obtained results have shown that ISSA's performance is very good compared to other heuristic methods in the literature and has shown that it can be used as an alternate method for contracted optimization. Also, well-known heuristic algorithms in the literature, which include HS, GA, DE, FF, CLPSO, GL-25, CMA-ES, and CoDE are compared with ISSA in low-scale dimensions.

In future studies, hybrid ISSA will be created by using different heuristic methods. Weight value will be added to ISSA's search space agents and the balance between the blasting and exploration will be provided. ISSA will be adapted to real-world problems.

Table 24 ISSA and its comparison with comparative algorithms in GWO, IGWO, NGWO, and MGWO (Dimension (n) = {100, 500, 1000})

F	n	GWO		IGWO		NGWO		MGWO		ISSA	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f15	100	9.24E+00	8.36E+00	**2.72E−05**	**6.93E−05**	7.56E−03	8.37E−03	7.50E+00	6.31E+00	6.54E+02	2.31E+01
	500	8.81E+01	2.89E+01	**1.09E−01**	**2.46E−01**	3.43E+00	6.68E+00	5.03E+01	1.73E+01	5.02E+03	7.63E+01
	1000	1.83E+02	6.45E+01	**1.25E+00**	**7.56E−01**	8.78E+00	9.73E+00	1.49E+02	4.76E+01	1.08E+04	1.24E+02
f18	100	3.40E−01	5.48E−02	**2.00E−01**	**0.00E+00**	**2.00E−01**	**0.00E+00**	3.40E−01	5.48E−02	2.95E+02	6.56E+00
	500	1.06E+00	**5.48E−02**	**3.00E−01**	1.00E−01	4.00E−01	0.00E+00	9.00E−01	0.00E+00	6.74E+02	9.80E+00
	1000	1.60E+00	1.00E−01	**4.00E−01**	**0.00E+00**	6.00E−01	**0.00E+00**	1.80E+00	1.00E−01	9.67E+02	1.40E+01
f19	100	6.52E−03	3.45E−03	2.83E−03	1.68E−03	2.99E−03	1.01E−03	8.12E−03	3.16E−03	**0.00E+00**	**0.00E+00**
	500	4.61E−02	1.39E−02	1.55E−02	5.89E−03	1.90E−02	1.15E−02	4.33E−02	1.03E−02	**0.00E+00**	**0.00E+00**
	1000	1.63E−01	1.90E−01	1.53E−02	9.40E−03	3.28E−02	1.05E−02	1.59E−01	3.32E−02	**0.00E+00**	**0.00E+00**

According to the promised results of the ISSA, we attempt to use it in future works in several other applications such as machine learning, image processing, and data mining.

References

Acılar AM (2013) Yapay Bağışıklık Algoritmaları Kullanılarak Bulanık Sistem Tasarımı, Konya, Turkey, Ph.D. thesis. (**in Turkish**)

Akay B (2013) A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. Appl Soft Comput 13:3066–3091

Baş E, Ülker E (2020a) A binary social spider algorithm for continuous optimization task. Soft Comput. https://doi.org/10.1007/s00500-020-04718-w

Baş E, Ülker E (2020b) An efficient binary social spider algorithm for feature selection problem. Expert Syst Appl 146:113185

Blum C, Li X (2008) Swarm intelligence in optimization. In: Swarm intelligence. Springer, pp 43–85

Cuevas E, Cienfuegos M (2014) A new algorithm inspired in the behavior of the social-spider for constrained optimization. Expert Syst Appl 4:412–425

Cuevas E, Cienfuegos M, Zaldívar D, Pérez-Cisneros M (2013) A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Syst Appl 40:6374–6384

Dell RF, Ewing PL, Tarantino WJ (2008) Optimally stationing army forces. Interfaces 38(6):421–435

Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. Comput Intell Mag IEEE 1(4):28–39

El-Bages MS, Elsayed WT (2017) Social spider algorithm for solving the transmission expansion planning problem. Electr Power Syst Res 143:235–243

Elsayed WT, Hegazy YG, Bendary FM, El-Bages MS (2016) Modified social spider algorithm for solving the economic dispatch problem. Eng Sci Technol Int J 19:1672–1681

Gandomi AH, Yang X-S, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35

Garcia-Martinez C, Lozano M, Herrera F, Molina D, Sanchez AM (2008) Global and local real-coded genetic algorithms based on parent-centric crossover operators. Eur J Oper Res 185(3):1088–1113

Garden RW, Engelbrecht AP (2014) Analysis and classification of optimization benchmark functions and benchmark suites. In: Proceedings of IEEE CEC 2014, pp 1641–1649

Goh CK, Lim D, Ma L, Ong YS, Dutta P (2011) A surrogate-assisted memetic co-evolutionary algorithm for expensive constrained optimization problems. In: IEEE congress on paper presented at the evolutionary computation (CEC)

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195

Hussain K, MohdSalleh MN, Cheng S, Naseem R (2017) Common benchmark functions for metaheuristic evaluation: a review. Int J Inform Vis 1(4):2

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report-tr06. Erciyes University, Engineering Faculty, Computer Engineering Department

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony. J Glob Optim 39(3):459–471

Kennedy J (2010) Particle swarm optimization. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning. Springer, Boston, pp 760–766

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, Perth, WA, pp 1942–1948

Kuzu S, Önay O, Şen U, Tunçer M, Yıldırım FB, Keskintürk T (2014) Gezgin satıcı problemlerinin metasezgiseller ile çözümü. J Bus Fac 43(1):1–27 **(in Turkish)**

Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10:281–295

Liao TW, Kuo RJ, Hu JTL (2012) Hybrid ant colony optimization algorithms for mixed discrete-continuous optimization problems. Appl Math Comput 219:3241–3252

Liu Y, Chen S, Guan B, Xu P (2019) Layout optimization of large-scale oil–gas gathering system based on combined optimization strategy. Neurocomputing 19:10

Long W (2016) Grey wolf optimizer based on nonlinear adjustment control parameter. In: 4th International conference on sensors, mechatronics, and automation (ICSMA 2016), advances in intelligent systems research, p 136

Long Q, Wu C, Wang X, Wu Z (2017) A modified quasisecant method for global optimization. Appl Math Model 51:21–37

Long W, Jiao J, Liang X, Tang M (2018) Inspired grey wolf optimizer for solving large-scale function optimization problems. Appl Math Model 60:112–126
Long W, Wu T, Liang X, Xu S (2019) Solving high-dimensional global optimization problems using an improved sine cosine algorithm. Expert Syst Appl 123:108–126
Masutti TAS, Castro LN (2016) TSPoptBees: a bee-inspired algorithm to solve the traveling salesman problem. In: Proceedings of the 2016 5th IIAI international congress on advanced applied informatics, IIAI-AAI), 2016, pp 593–598
Maucec MS, Brest J (2019) A review of the recent use of differential evolution for large-scale global optimization: an analysis of selected algorithms on the CEC 2013 LSGO benchmark suite. Swarm Evolut Comput 50:1–18
Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69(3):46–61
Mittal N, Singh U, Sohi BS (2016) Modified grey wolf optimizer for global engineering optimization. Appl Comput Intell Soft Comput 2016:1–16
Mohapatra P, Das KN, Roy S (2017) A modified competitive swarm optimizer for large scale optimization problems. Appl Soft Comput 59:340–362
Mousa A, Bentahar J (2016) An efficient QoS-aware web services selection using social spider algorithm. In: The 13th international conference on mobile systems and pervasive computing (MobiSPC 2016), procedia computer science, vol 94, pp 176–182
Nakib A, Ouchraa S, Shvai N, Souquet L, Talbi E-G (2017) Deterministic metaheuristic based on fractal decomposition for large-scale optimization. Appl Soft Comput 61:468–485
Parpinelli RS, Lopes HS (2011) New inspirations in swarm intelligence: a survey. Int J Bio-inspired Comput 3(1):1–16
Ray T, Yao X (2009) A cooperative coevolutionary algorithm with correlation-based adaptive variable partitioning. In: IEEE congress on paper presented at the evolutionary computation, CEC'09
Sayed E, Essam D, Sarker R, Elsayed S (2015) A decomposition-based evolutionary algorithm for large scale constrained problems. Inf Sci 316:457–486
Shayanfar H, Gharehchopogh FH (2018) Farmland fertility: a new metaheuristic algorithm for solving continuous optimization problems. Appl Soft Comput 71:728–746
Shukla UP, Nanda SJ (2016) Parallel social spider clustering algorithm for high dimensional datasets. Eng Appl Artif Intell 56:75–90
Shukla UP, Nanda SJ (2018) A binary social spider optimization algorithm for unsupervised band selection in compressed hyperspectral images. Expert Syst Appl 97:336–356
Singh D, Agrawal S (2016) Self-organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. Appl Soft Comput 38:1040–1048
Sun G, Zhao R, Lan Y (2016) Joint operations algorithm for large-scale global optimization. Appl Soft Comput 38:1025–1039
Sun Y, Wang X, Chen Y, Liu Z (2018) A modified whale optimization algorithm for large-scale global optimization problems. Expert Syst Appl 114:563–577
Surjanovic S, Bingham D (2019) Virtual library of simulation experiments: test functions and datasets. http://www.sfu.ca/ssurjano
Tawhid MA, Dsouza KB (2018) Hybrid binary bat enhanced particle swarm optimization algorithm for solving feature selection problems. Appl Comput Inform
Trunfio A-G, Topa P, Was J (2016) A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution. Inf Sci 372:773–795
Wang C-F, Song W-X (2019) A novel firefly algorithm based on gender difference and its convergence. Appl Soft Comput J 80:107–124
Wang Y, Cai ZX, Zhang QF (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
Wang H, Wang W, Zhou X, Sun H, Zhao J, Yu X, Cui Z (2017) Firefly algorithm with neighborhood attraction. Inform Sci 382:374–381
Wong LP, Low MYH, Chong CS (2008) A bee colony optimization algorithm for traveling salesman problem. In: Proceedings of the second asia international conference on modeling and simulation, pp 818–823
Yang XS (2009) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer, pp 169–178
Yang XS (2010a) A new metaheuristic bat-inspired algorithm. In: Gonzalez JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, pp 65–74
Yang X-S (2010b) Firefly algorithm, stochastic test functions and design optimization. Int J Bio-Inspired Comput 2(2):78–84

Yang XS, Deb S (2009) Cuckoo search via Levy flights. In: Proceedings of the world congress on nature and biologically inspired computing (NaBIC 2009, India). IEEE Publications, New York, pp 210–214
Yildiz YE, Topal AO (2019) Large scale continuous global optimization based on micro differential evolution with local directional search. Inf Sci 477:533–544
Yu JJQ, Li VOK (2015) A social spider algorithm for global optimization. Appl Soft Comput 30:614–627
Yu JJQ, Li VOK (2016) A social spider algorithm for solving the non-convex economic load dispatch problem. Neurocomputing 171(C):955–965