Soft Computing (2021) 25:181–206
https://doi.org/10.1007/s00500-020-05130-0

METHODOLOGIES AND APPLICATION

A novel lifetime scheme for enhancing the convergence performance of salp swarm algorithm

Malik Braik[1] · Alaa Sheta[2] · Hamza Turabieh[3] · Heba Alhiary[4]

Published online: 6 July 2020

Abstract
The performance of any meta-heuristic algorithm depends highly on the setting of dependent parameters of the algorithm. Different parameter settings for an algorithm may lead to different outcomes. An optimal parameter setting should support the algorithm to achieve a convincing level of performance or optimality in solving a range of optimization problems. This paper presents a novel enhancement method for the salp swarm algorithm (SSA), referred to as enhanced SSA (ESSA). In this ESSA, the following enhancements are proposed: First, a new position updating process was proposed. Second, a new dominant parameter different from that used in SSA was presented in ESSA. Third, a novel lifetime convergence method for tuning the dominant parameter of ESSA using ESSA itself was presented to enhance the convergence performance of ESSA. These enhancements to SSA were proposed in ESSA to augment its exploration and exploitation capabilities to achieve optimal global solutions, in which the dominant parameter of ESSA is updated iteratively through the evolutionary process of ESSA so that the positions of the search agents of ESSA are updated accordingly. These improvements on SSA through ESSA support it to avoid premature convergence and efficiently find the global optimum solution for many real-world optimization problems. The efficiency of ESSA was verified by testing it on several basic benchmark test functions. A comparative performance analysis between ESSA and other meta-heuristic algorithms was performed. Statistical test methods have evidenced the significance of the results obtained by ESSA. The efficacy of ESSA in solving real-world problems and applications is also demonstrated with five well-known engineering design problems and two real industrial problems. The comparative results show that ESSA imparts better performance and convergence than SSA and other meta-heuristic algorithms.

Keywords Salp swarm algorithm · Lifetime convergence scheme · Meta-heuristic · Benchmark test functions · Engineering optimization problems

1 Introduction

Meta-heuristic algorithms have become quite popular in solving engineering design problems (Gandomi et al. 2013; Wang et al. 2014), modeling of chemical processes (Sheta et al. 2019), diagnosis of medical images (Braik et al. 2019) and many other problems (Mirjalili et al. 2017; Wang 2018). In recent years, many meta-heuristic algorithms have been proposed and have received excellent acceptance by researchers in the artificial intelligence community due to their prominence in generating low-cost and robust solutions for complex optimization and non-deterministic polynomial problems (Mirjalili et al. 2017). The literature review of meta-heuristic algorithms reveals their efficiency and reliability

Communicated by V. Loia.

✉ Alaa Sheta
shetaa1@southernct.edu

Malik Braik
mbraik@bau.edu.jo

Hamza Turabieh
h.turabieh@tu.edu.sa

Heba Alhiary
hhiary@bau.edu.jo

[1] Department of Computer Science, Al-Balqa Applied University, Salt, Jordan

[2] Computer Science Department, Southern Connecticut State University, 501 Crescent St., New Haven, CT 06515, USA

[3] Department of Information Technology, Taif University, Taif, Saudi Arabia

[4] Department of Computer Information Systems, Al-Balqa Applied University, Salt, Jordan

in solving real-world problems in various fields of study. Ant colony optimization (ACO) algorithm (Maniezzo 1992) inspired by the foraging behavior of ants is very effective in solving structural optimization problems (Luh and Lin 2009), genomics (Greene et al. 2008) and also in traffic area control problems (Sattari et al. 2014). Particle swarm optimization (PSO) is a well-known optimization algorithm that mimics the social behavior of fish schooling or bird flocking (Eberhart and Kennedy 1995). The performance of PSO in solving several complex problems such as image processing (Omran et al. 2006), electric power systems (AlRashidi and El-Hawary 2008) and offshore heavy oil reservoir (Wang and Qiu 2013) is distinctly confirmed in the literature. Salp swarm algorithm (SSA) (Mirjalili et al. 2017) is a recently evolved meta-heuristic algorithm, motivated by the dynamic swarming behavior of salps while searching for food in the oceans (Mirjalili et al. 2017). SSA has been used successfully to solve a large number of real-world problems in many branches of science such as extracting the parameters of the electrical equivalent circuit of a photovoltaic system cell (Abbassi et al. 2019), color image segmentation (Xing and Jia 2019), optimization of software-defined networks (Ateya et al. 2019) and training the weights and biases of a feed-forward neural network (FFNN) (Bairathi and Gopalani 2019). Artificial bee colony (ABC) algorithm (Karaboga and Basturk 2007), cuckoo search (CS) algorithm (Yang and Deb 2009), firefly algorithm (FA) (Yang 2009), bat algorithm (BA) (Yang 2010a), fruit fly optimization algorithm (FOA) (Pan 2012), gray wolf optimization (GWO) (Mirjalili et al. 2014), whale optimization algorithm (WOA) (Mirjalili and Lewis 2016) and moth search (MS) algorithm (Wang 2018) are some of the well-known swarm intelligence-based algorithms. Many of these algorithms have been used successfully to address many problems in several areas such as medical diagnoses (Wang et al. 2017), process control (dos Santos Coelho and Mariani 2012), image processing (El Aziz et al. 2017), text clustering (Rashaideh et al. 2018), feature selection and classification (Hegazy et al. 2018; Ibrahim et al. 2019) and many other applications (Mavrovouniotis et al. 2017; Ali et al. 2019).

With the development of several optimization algorithms, it is difficult to identify which algorithm is most appropriate to solve a given problem. This is due to that most of the meta-heuristic algorithms work on a generalized concept and do not have domain knowledge specific to each problem (KS and Murugan 2017). Many parameters have to be tuned for any meta-heuristic algorithm, which can provide more flexibility and robustness for an optimization algorithm, but this process requires careful initialization (Kumar et al. 2015). However, all meta-heuristic algorithms have their dependent parameters, where the performance of an algorithm is largely dependent on the values of its parameters. These algorithms' dependent parameters may have a large influence on the effectiveness of algorithms' search agents in exploring and exploiting the search space. The process of finding the best parameter setting for an algorithm is expected to enhance the applicability of an algorithm to solve a range of optimization problems in various areas of science. Parameter setting problem for meta-heuristic algorithms reveals the importance of presenting a robust mechanism that can automatically choose the best parameter setting for an algorithm to achieve its optimality in solving a given problem (Yang et al. 2013). However, parameter setting control for meta-heuristic algorithms is still an open problem.

Several parameter setting techniques have recently been evolved for several meta-heuristic algorithms as a part of enhancing optimization algorithms. They tend to show significantly improved performance compared to their meta-heuristic algorithms' counterparts in solving benchmark functions or particular optimization problems. Many researchers first identified the parameters' values of meta-heuristics in solving an optimization problem based on trial and error paradigm (Fallahi et al. 2014; Jain et al. 2019). For instance, Dobslaw (2010) used an automated Design of Experiment (DoE) framework to identify sensible initial parameter settings for problem instances for various meta-heuristic algorithms such as PSO and ACO. Also, a parameter setting method for ACO based on DoE was presented to solve 7 traveling salesman problem instances (Fallahi et al. 2014), where different solutions were obtained for the same problem using different parameter settings. Geem and Sim (2010) proposed a parameter-setting-free (PSF) technique to find the best parameter setting for the harmony search (HS) algorithm. Khadwilard et al. (2012) investigated the performance of FA for solving a job shop scheduling problem with various parameter settings. A method for self-tuning algorithms so that an algorithm to be tuned can be used to automatically tune its parameters was presented by Yang et al. (2013). They used FA to tune itself, where the authors claimed that this method of self-tuning algorithm worked well. Using an upper-level meta-heuristic to decide the most appropriate set of parameters for a low-level meta-heuristic was presented in Crawford et al. (2013). Crawford et al. (2013) applied a genetic algorithm (GA) as an upper-level meta-heuristic to optimize the parameter values of ACO and scatter search meta-heuristics for solving a particular problem. The idea is to transfer the parameter setting effort of one algorithm to another algorithm.

1.1 SSA challenges

The parameter setting for the dominant parameter of SSA is the foremost problem of SSA, where inappropriate values for the factors of this parameter may drive SSA to stagnate in local optima. Another problem of SSA is that it does not explicitly keep track of its best positions in for-

mer generations, which bounds its exploitation ability and leads to early convergence in some basic benchmark functions. Even though global and local search capabilities of SSA are somewhat convincing of randomization behavior and dynamic swarming behavior of salps, these search capabilities are sometimes limited which causes the solutions to get fall into local optima in some optimization problems.

To overcome these flaws, many researchers have developed hybridized versions of SSA with other algorithms to improve performance compared to their original counterparts. For example, Sayed et al. (2018) employed a hybrid approach between Chaos theory and SSA (CSSA) to solve a set of basic benchmark functions. The obtained results showed that CSSA can improve the convergence rate for SSA. Hegazy et al. (2018) proposed an improved salp swarm algorithm (ISSA) by controlling the parameters of SSA using the DoE framework to enhance the exploration process of SSA for solving feature selection problems. Ibrahim et al. (2019) proposed a hybrid approach between SSA and PSO (SSAPSO) to solve feature selection problems on datasets obtained from the UCI machine learning repository. Ali et al. (2019) proposed a hybrid approach between SSA and weighted L_1-norm optimization to design 2nd to 4th wideband infinite impulse response (IIR) digital differentiators (DDs). This hybridization approach was presented to enhance the exploration of SSA at the initial stages and exploitation at later stages by iteration-level hybridization process to get the global optimum solution. However, these techniques proposed for enhancing the performance of SSA are either very preliminary or computationally complex and require a relatively large computational cost.

From the literature, SSA is a promising optimization algorithm with room for possible improvement and employment in various disciplinary areas. For example, there is no proper discussion about the performance of SSA regarding its tuning parameters. Also, most applications of SSA on real-world problems have not explored the proper parameter settings of SSA. Finally, how do we control the exploration process of SSA up to the extent that SSA does not become so slow to converge? Therefore, a more reliable and computationally efficient approach to enhance SSA is required, but with the simplicity and speed of SSA. This is the motivation behind this study, in which a novel enhancement to SSA is proposed as given below.

1.2 Motivations of the proposed work

In this paper, to overcome the above shortcomings of SSA, we propose a novel lifetime convergence scheme for SSA, referred to as enhanced SSA (ESSA), to reinforce the exploration and exploitation capabilities of SSA. The proposed ESSA works in two stages: In the first stage, a new position updating mechanism is proposed to control the movement of salps in the search space. In this process, the salps that store the coordinates of the best positions that have achieved so far in previous generations are incorporated into the updating position process. In the next stage, a new dominant parameter different from that used in SSA is proposed in ESSA. Then, an innovative methodology to automatically self-tuning this dominant parameter is proposed. This is the main goal of this work, where a novel lifetime convergence process is incorporated with the proposed ESSA in a self-tuning scheme to ensure the ability of exploration at the initial iterations and the capacity of exploitation at the later iterations of the evolutionary process of ESSA. These improvements made to SSA assist the salps to explore different promising areas in the deep ocean and exploit each search area to find food sources. Further, they shall provide great potential for the salps to avoid stagnation in local optima and help them to move toward the global optimum solution with increased precision.

1.3 Contributions

The contribution of this work are as follows:

- Proposing a new position updating process for SSA, referred to as ESSA, that could improve the swarming behavior of the salps.
- Integrating a new lifetime convergence scheme with ESSA, which can be updated at each iteration loop of ESSA to reach optimal convergence. This is an innovation process where ESSA is used to tune itself. It is anticipated that this process will ultimately increase the exploration and exploitation abilities of ESSA to obtain the global optimum (i.e., food), rapidly and potentially efficiently and
- The performance of ESSA is compared with other state-of-the-art meta-heuristic algorithms on basic benchmark functions and several computational engineering design problems commonly used by meta-heuristic optimization algorithms in the literature.

In Sect. 2, the original SSA is described in detail. Section 3 then presents the proposed ESSA and provides a description of the lifetime convergence process proposed. The evaluation, convergence and statistical test results are presented in Sect. 4. Section 5 presents the results of ESSA on engineering design problems compared to other algorithms. Section 6 then presents the results of ESSA on modeling industrial problems. Section 7 presents an analysis and discussion of the obtained results with concluding comments and future scope in Sect. 8.

2 Basics of SSA

Salp swarm algorithm is a new meta-heuristic algorithm inspired by the unique and superior behavior of salps while searching for food in deep ocean (Mirjalili et al. 2017). Foraging behavior of salps is characterized by splitting the salp chain into a leader and followers that move toward food sources into deep ocean. That is, the first position in the chain is for the leader, while the remaining positions in the chain are for the followers. The leader guides the followers while searching for food sources, denoted by F. For n salps in a d-dimensional search space, the position of all salps is defined by a matrix x_n^d. Consider population of salps of size n, the position of the ith salp in SSA can be given as shown in Eq. (1).

$$x_i = \left(x_i^1, x_i^2, \ldots, x_i^k, \ldots, x_i^{d-1}, x_i^d\right) \tag{1}$$

where i represents the ith salp in the population of SSA, d stands for the dimension of the ith salp and x_i^k corresponds to the position of the ith salp in the kth dimension of the search space. Equation (2) can be used to assign the initial position of each salp in the salp chain.

$$x_i = ub^k + r \times (ub^k - lb^k) \tag{2}$$

where lb^k and ub^k are the lower and upper bounds of the ith salp in the kth dimension, respectively, and r is a randomly generated number in the range from 0 to 1.

The position of the leader salp at the first position in the kth dimension, x_1^k, in the salp chain is updated according to Eq. (3) (Mirjalili et al. 2017).

$$x_1^k = \begin{cases} F^k + c_1[(ub^k - lb^k)c_2 + lb^k] & c_3 \geq 0.5 \\ F^k - c_1[(ub^k - lb^k)c_2 + lb^k] & c_3 < 0.5 \end{cases} \tag{3}$$

where F^k identifies the food source position in the kth dimension, ub^k denotes the upper bound of the kth dimension, lb^k denotes the lower bound of the kth dimension, and c_2 and c_3 are random numbers generated uniformly in the interval from 0 to 1.

Equation (3) states that the leader in SSA only updates its location in regard to the source of food F. The c_2 and c_3 parameters decree if the following position of the salp approach ∞ or $-\infty$ with the next movement. The c_1 parameter is defined as shown in Eq. (4) (Mirjalili et al. 2017).

$$c_1 = 2e^{-(\frac{4l}{L})^2} \tag{4}$$

where l and L identify the current iteration and the maximum number of iterations, respectively. The c_1 parameter is the leading and dominant parameter in SSA, as it creates a balance between exploration and exploitation of the search space. The position of the followers is updated according to Eq. (5).

$$x_i^k = \frac{1}{2}(x_1^k + x_i^{k-1}) \tag{5}$$

where $i \geq 2$, x_i^k and x_i^{k-1} stand for the positions of the ith follower salp in the kth and $k-1$th dimensions, respectively, and x_1^k represents the position of the leader salp in the kth dimension.

The fitness function is evaluated at each iteration within the evolutionary process loop of SSA. The position updating processes of the leader and followers are continued until the stopping condition is reached. The basic steps of SSA can be described by the iterative procedural steps illustrated in Algorithm 1.

Algorithm 1: A pseudocode of the basic SSA.

```
Initialize the population of salps x_i (i = 1, 2, ..., n)
while (termination condition is not met) do
    Find the fitness value of each search salp
    F = the source of food that the salps search for
    Update the dominant parameter c1 in SSA using Eq. (3)
    for each salp or referred to as x_i do
        if (i == 1) then
            Update the leader's position using Eq. (3)
        else
            Update the positions of the followers using Eq. (5)
        end if
    end for
    Check the positions of salps according to ub and lb bounds of
    problem variables.
end while
Return the search agent that found the food source F
```

As shown in Algorithm 1, if any of the salps moves out of the search area, it will be returned to the boundaries based upon the simulated steps of SSA. All the steps of SSA in Algorithm 1 except the initialization step are executed repeatedly at each loop until the termination condition is met. It is observed that the food source, F, is updated throughout the evolutionary process of SSA since it is very likely that the salp chain will find a better solution by exploring and exploiting the surrounding environment. The leader and followers can move toward the global optimum throughout iterations.

3 Proposed ESSA

This section details the proposed position updating process for SSA. Then, it explains the proposed self-tuning method applied to improve parameter setting and convergence process for the proposed algorithm.

3.1 Proposed mathematical model for ESSA

In SSA and ESSA, the position of the food source needs to be updated iteratively. The position of the leader and the accompanying salps can be updated at dimension k as given below:

$$x_{i,l+1}^{k} = F^{k} + \triangle x \tag{6}$$

where $\triangle x$ represents the new update and $x_{i,l+1}^{k}$ stands for the position of the ith salp at iteration $l+1$ at dimension k.

The term $\triangle x$ needs to be chosen carefully to achieve two essential aspects: *exploration* and *exploitation* of the search space. Further, this term should support the convergence path of the search agents (i.e., salps) toward the food source. Tuning $\triangle x$ is a significant matter that has a foremost impact on the overall performance of ESSA. This term plays a vital role in exploring the search space by helping the individuals of ESSA to move efficiently to the food. Therefore, it is useful to support updating this term with a lifetime convergence mechanism to achieve a proper convergence process and keep the exploration rate high throughout iterations.

In this work, $\triangle x$ was presented as a function of a lifetime parameter, c_1, as shown in Eq. (7).

$$\begin{aligned} x_{l+1} &= f(F, \triangle x) \\ \triangle x &= f(x_l, c_1, ub, lb, c_2) \end{aligned} \tag{7}$$

where F represents the position of the food source, x_l represents the current solution to a given problem at iteration l, c_2 is a randomly generated number in the range from 0 to 1 and f is a nonlinear mapping function from a given solution, x_l, to a new solution x_{l+1}.

In the position updating process of ESSA in Eq. (7), ESSA tends to generate a new solution x_{l+1} at iteration $l+1$ for a given problem that is better than the current solution x_l at iteration l. In ESSA, the positions of the salp leader and its accompanying salps are updated with a mathematical formula different from that used in SSA. The update mechanism of salps in ESSA is given in Eq. (8).

$$x_{i,l+1}^{k} = \begin{cases} F^{k} + c_1 x_{i,l}^{k} \left[\left(ub^{k} - lb^{k} \right) c_2 + lb^{k} \right] & c_3 \geq 0.5 \\ F^{k} - c_1 x_{i,l}^{k} \left[\left(ub^{k} - lb^{k} \right) c_2 + lb^{k} \right] & c_3 < 0.5 \end{cases} \tag{8}$$

where $i \leq n/2$ indicates the leader of the salp chain and the accompanying salps of the leader, $x_{i,l+1}^{k}$ and $x_{i,l}^{k}$ are the next and current positions of the salp leader and its accompanying salps at iterations $l+1$ and l, respectively, the parameter c_1 was used to dominate the movement of salps of ESSA and is reformulated in ESSA as shown below:

$$\begin{aligned} c_1 &= \lambda_1 e^{K} \\ K &= -\lambda_2 \sqrt{\lambda_3 \times l} \end{aligned} \tag{9}$$

where the coefficients λ_1, λ_2 and λ_3 are used to update c_1 at each iteration loop of ESSA.

In ESSA, the position of the followers, representing the $i > n/2$ salps, are automatically updated using Eq. (5). The positions of the population of salps are updated iteratively with the help of the proposed lifetime convergence process. This process is used to control the parameter c_1 through the evolutionary process of ESSA throughout iterations. This parameter consists of three coefficients, λ_1, λ_2 and λ_3, that are needed to update iteratively to automatically tune c_1 using ESSA itself.

These three coefficients λ_1, λ_2 and λ_3 are proposed to provide a proper setting for c_1, where the proposed lifetime convergence process can empower the salp chain to explore more search space and exploit each promising area while searching for food sources. This is to reach an effective convergence process for ESSA, which can further improve the performance of ESSA in solving optimization problems. This proposed parameter setting method is described below.

3.2 Parameter setting-based ESSA

The proposed ESSA was used to tune the coefficients of the parameter c_1 using itself. This process is referred to as a self-tuning method. Mathematically speaking, we can describe the mathematical model of ESSA as:

$$x_{i,l+1}^{k} = f\left(F, x_{i,l}^{k}, \mathbf{p}, ub, lb, c_2 \right) \tag{10}$$

where the parameter vector $\mathbf{p} = (\lambda_1, \lambda_2, \lambda_3)$. As the parameter vector $\mathbf{p}$ is the target of the proposed self-tuning parameter method, the representation formula shown in Eq. (10) can be reduced to:

$$x_{i,l+1}^{k} = f\left(x_{i,l}^{k}, \mathbf{p} \right) \tag{11}$$

It is known that there is a global optimum, $f_{\min}(x)$, for each optimization problem to be optimized. This optimality depends on both ESSA itself and the optimization problem, ϕ, to be addressed. That is, the optimality to be realized is to maximize the performance of:

$$\zeta = f(\phi, \mathbf{p}) \tag{12}$$

This optimality is then denoted as:

$$\zeta_{*} = f_{*}(\phi, \mathbf{p}_{*}) \tag{13}$$

where the parameter $\mathbf{p}_{*}$ is the optimal parameter setting for c_1.

For an optimization problem, the goal is to find the global optima f_* for a function $f(x)$ in a d-dimensional search space. That is, to minimize the function:

$$f(x) = (x_1, x_2, \ldots, x_d) \tag{14}$$

To optimize a problem, there is a need to find $f_{\min}$ which may be close to the actual global optimum f_*. For a predetermined tolerance ρ, this may require L_ρ iterations to realize $|f_{\min} - f_*| \leq \rho$. Obviously, L_ρ will largely depend on both the problem objective and the coefficients λ_1, λ_2 and λ_3 of vector $\mathbf{p}$ that can be used to tune c_1.

The aim of this self-tuning method is to find $\mathbf{p}_*$ in order to achieve the best setting for c_1 with the minimum number of iterations L_ρ. This idea could be addressed as a multi-objective optimization problem with two objectives: The first objective is for the given problem, ϕ, and the second objective is for the number of iterations, L_ρ. This parameter tuning method can be written as follows.

$$\begin{aligned} &\text{Minimize} \quad f(\mathbf{x}) \\ &\text{Minimize} \quad L_\rho = \text{ESSA}(f(x), \mathbf{p}) \end{aligned} \tag{15}$$

where L_ρ is the number of iterations carried out to achieve a pre-identified tolerance ρ so that the solution $f_{\min}$ is close enough to the true global solution f_*, respecting that $|f_{\min} - f_*| \leq \rho$

This means that for tolerance ρ, there will be a pool of best parameter settings with a minimum L_ρ. But since ρ is usually given, the normal way to solve this problem is to use the so-called λ_1, λ_2 and λ_3-constraints or ρ-constraint method. So, for a given $\rho \geq 0$, we can change one of the targets (i.e., $f(\mathbf{x})$) into a constraint, so that Eq. (16) becomes a single-objective optimization problem with a constraint as shown in Eq. (16).

$$\text{Minimize } L_\rho = ESSA(f(x), \mathbf{p}) \tag{16}$$

Subject to:

$$f(\mathbf{x}) \leq \rho \tag{17}$$

Ideally suited, the vector $\mathbf{p}_*$ must be adequately robust, where any slight variation in $\mathbf{p}_*$ should not significantly affect the accuracy of ESSA in solving various types of problems. This implies that $\mathbf{p}_*$ should fall into a flat range, not at a sharp peak in the parameter landscape.

For ESSA to converge properly, we employed exponential formulas for λ_1, λ_2 and λ_3 similar to that used for c_1 in Eq. (4), as shown in Eqs. (18), (19) and (20), respectively.

$$\lambda_1 = \theta_1 e^{-(\frac{l}{L})^2}, \quad \theta_1 \in (1, 10) \tag{18}$$

$$\lambda_2 = \theta_2 e^{-(\frac{l}{L})^2} \quad \theta_2 \in (0.25, 2) \tag{19}$$

$$\lambda_3 = \theta_3 e^{-(\frac{l}{L})^2} \quad \theta_3 \in (1, 3) \tag{20}$$

where the parameters to be tuned in this work become θ_1, θ_2 and θ_3, where the parameters λ_1, λ_2 and λ_3 can be obtained from Eqs. (18), (19) and (20), respectively, and then c_1 using Eq. (9).

3.3 Time complexity of ESSA

The computational complexity of ESSA can be given as:

$$\mathcal{O}(v(L(sd + cs + ms^2))) \tag{21}$$

where v represents the number of runs, L is the number of iterations, s displays the number of solutions, d shows the number of variables of the optimization problem (i.e., dimension), c denotes the cost of the objective function and m stands for the number of objectives.

3.4 Space complexity of ESSA

The space complexity of ESSA in terms of the memory space depends on the parameters of both the search agents and the dimension of the given problem. This identifies the amount of space that ESSA needs during the initialization process. Hence, the space complexity of ESSA can be defined as:

$$\mathcal{O}(sd) \tag{22}$$

ESSA also requires additional space for other parameters such as λ_1, λ_2, λ_3, $\mathbf{p}$ and c_1, but this extra space is not critical, and therefore, the space complexity for ESSA remains within the range given in Eq. (22).

The parameters in Eq. (21) have an effect on the speed of ESSA while searching for the global optimal solution. The search time to find the global optimum increases as the values for these parameters increase. The number of iterations and search agents required to satisfactorily find the global solution depends on the complexity of the given problem. Therefore, the values of these parameters should be chosen carefully to precisely solve the problem.

3.5 Analysis of ESSA

The parameters λ_1, λ_2 and λ_3 in Eq. (9) were tuned through Eqs. (18), (19) and (20), respectively. These values were iteratively updated by a self-tuning method-based ESSA itself to reach the optimality, so that c_1 can promote the exploration and exploitation of ESSA, where the global minimum f_* can be obtained. The main procedures of ESSA can be summarized by the iterative steps described in Algorithm 2.

Algorithm 2: A pseudocode of ESSA.

Initialize the population of n salps x_i $(i = 1, 2, \ldots, n)$ with consideration of ub and lb bounds for the problem's variables
$L \leftarrow$ Maximum number of iterations
$l \leftarrow$ Current iteration
$\rho \leftarrow$ A predetermined tolerance
$\mathbf{p} \leftarrow (\lambda_1, \lambda_2, \lambda_3)$
$L_\rho(f(x), \mathbf{p})) \leftarrow$ Objective function for ESSA
$f(x) \leftarrow$ Objective function for the problem
$f \leftarrow$ Fitness solution of the population
Evaluate the fitness value of each salp in the initial population
while ($l < L$ OR $f_{\min} > \rho$) **do**
  Computer the fitness of each search agent
  F = the best search salp
  Update the parameters λ_1, λ_2 and λ_3 of the vector $\mathbf{p}$ using Eqs. (18), (19) and (20), respectively.
  Update the essential parameter c_1 using Eq. (9)
  for each salp ($i.e.$, x_i) **do**
    if ($i \leq n/2$) **then**
      Update the position of the leading salps using Eq. (8)
    end if
    if ($i > n/2$ AND $i \leq n$) **then**
      Update the position of the follower salps using Eq. (5)
    end if
  end for
  Modify the salp positions based on ub and lb limits
  Evaluate and find the fitness value f for each search agent (i.e., salp)
  Solve the objective function min $L_\rho(f(x), \mathbf{p})$
  Find the optimality solution $f_{\min}$ within ρ
  $l = l + 1$
end while
Obtain the number of iterations L_ρ conducted to find $f_{\min}$
Obtain the best parameter setting $\mathbf{p}_*$ (i.e., best values for $\lambda_1, \lambda_2, \lambda_3$)
Return the best search salp F

In Algorithm 2, ESSA first initializes the population of salps concerning the upper and lower bounds of the problem variables. Then this algorithm computes the objective value for each salp and updates the parameters λ_1, λ_2 and λ_3 using Eqs. (18), (19) and (20), respectively. The objective value and these parameters are updated inside the evolutionary process of ESSA. These parameters need to decrease gradually in terms of θ_1, θ_2 and θ_3, respectively. These parameters are necessary to update the vector $\mathbf{p}$. The next step is to use $\mathbf{p}$ to update the parameter c_1 using Eq. (9) and update the position of the leader and followers using Eqs. (8) and (5), respectively. This process is repeated at each iteration loop of ESSA, allowing ESSA to update this parameter by ESSA itself. Essentially, this self-tuning parameter method-based ESSA achieves two goals at once: finding the best setting for the parameter c_1 and finding the optimal objective value. The food source and position of the leader and followers will be updated during the optimization process because the salp chain is very probably to pursue a better solution by exploring and exploiting the surrounding area. The imitations of salps' behavior in Algorithm 2 show that the salp chain movement modeled in the proposed ESSA can explore and exploit the area around both stationary and moving food sources. So, the salp chain has the potency to move toward the global optimum throughout iterations. Based on the above illustration of the strengths of the proposed ESSA, it is expected to be effective in solving optimization problems.

4 Experimental results and discussion

Many evaluation experiments were conducted in this section to clarify the theoretical assertions discussed the proposed ESSA in the previous section.

1. First, twenty-three standard benchmark functions are used to evaluate the performance of the proposed algorithm.
2. Second, qualitative results are obtained and presented to assess the general efficacy of the proposed ESSA in optimizing a collective set of benchmark test functions.
3. Third, the efficiency of the proposed ESSA is compared with the original SSA and other compelling metaheuristic algorithms.
4. Fourth, convergence curves are provided to substantiate the accuracy and reliability levels of ESSA compared to SSA and other algorithms.

4.1 Benchmark test functions

It is always beneficial to use a set of standard testbeds with different characteristics to conveniently and confidently test the performance of any new algorithm on different benchmark functions and compare it with other algorithms. The diversity of test functions allows observing and testing the ability of any new algorithm from different perspectives.

A set of twenty-three basic benchmark functions are used to demonstrate the efficiency of the proposed algorithm compared to the standard SSA and other existing algorithms in the literature. These functions can be grouped into three main categories: unimodal (Dhiman and Kumar 2017), multimodal (Mirjalili 2015) and fixed-dimension multimodal (Dhiman and Kumar 2017). These functions are mathematically described in "Appendix A." A detailed description of the characteristics of the unimodal functions, (F_1–F_7), multimodal functions (F_8–F_{13}) and fixed-dimension multimodal functions (F_{14}–F_{23}) is presented in Table 14, where Dim indicates the dimension of the function space, range stands for the search space limits of the function and $f_{\min}$ indicates the best-reported value.

The unimodal test functions have only one global optimum with no local optima. These functions are highly suitable to explore the convergence behavior and exploitative efficiency of the proposed algorithm. Multimodal and fixed-dimension

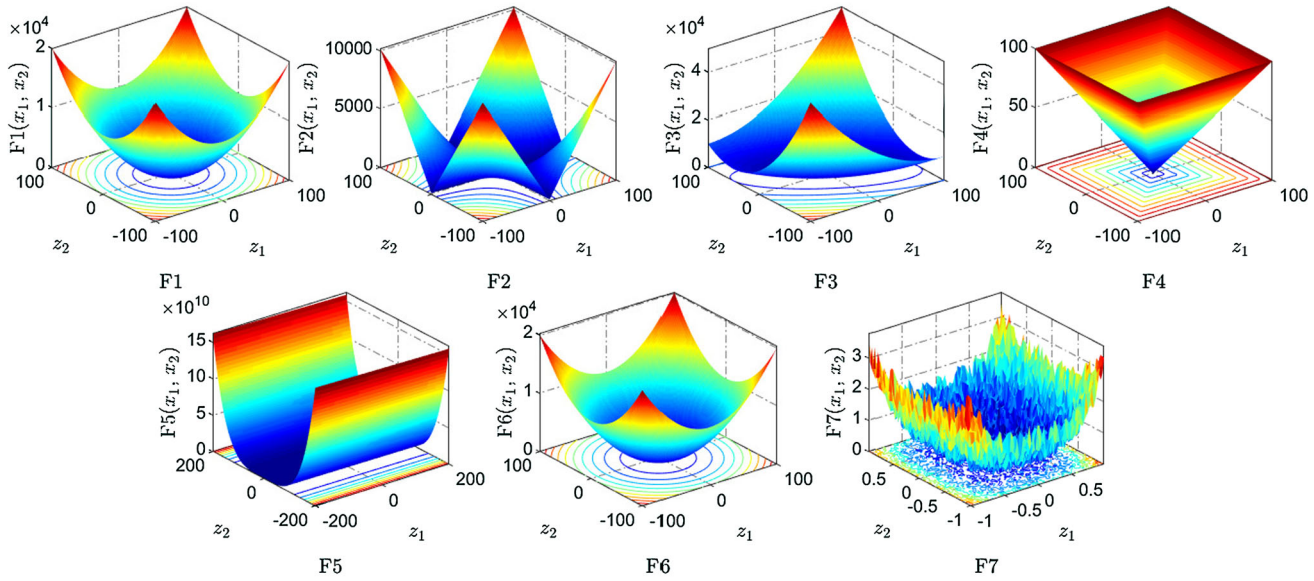

Fig. 1 Parameter space of the unimodal benchmark functions

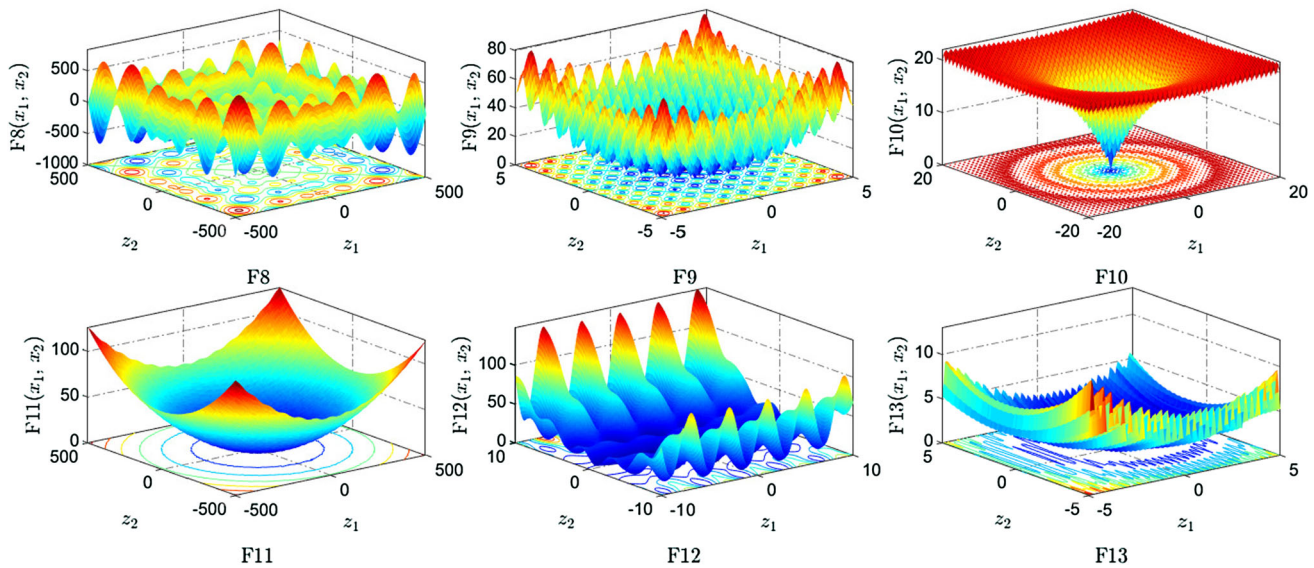

Fig. 2 Parameter space of the multimodal benchmark functions

multimodal test functions face the existence of several local optimum solutions and more than one global optimum. These functions are very appropriate to verify local optimum avoidance and study the explorative behavior of the proposed algorithm. All of these benchmark functions represent minimization problems.

Figures 1, 2 and 3 show the landscapes of unimodal, multimodal and fixed-dimension multimodal benchmark functions in two-dimensional versions, respectively.

To verify the accuracy of the proposed ESSA, its results on the functions described in "Appendix A" are compared to the results of the standard SSA (Mirjalili et al. 2017) and the results of six well-known algorithms: moth-flame optimization (MFO) (Mirjalili 2015), multi-verse Optimizer (MVO) (Mirjalili et al. 2016), sine–cosine algorithm (SCA) (Mirjalili 2016), gravitational search algorithm (GSA) (Rashedi et al. 2009), GA (Bonabeau et al. 1999) and HS (Geem et al. 2001). The parameter settings of these algorithms are given in the following subsection.

4.2 Experimental setup

The parameter settings of the proposed ESSA and other algorithms such as SSA, MFO, MVO, SCA, GSA, GA and HS are

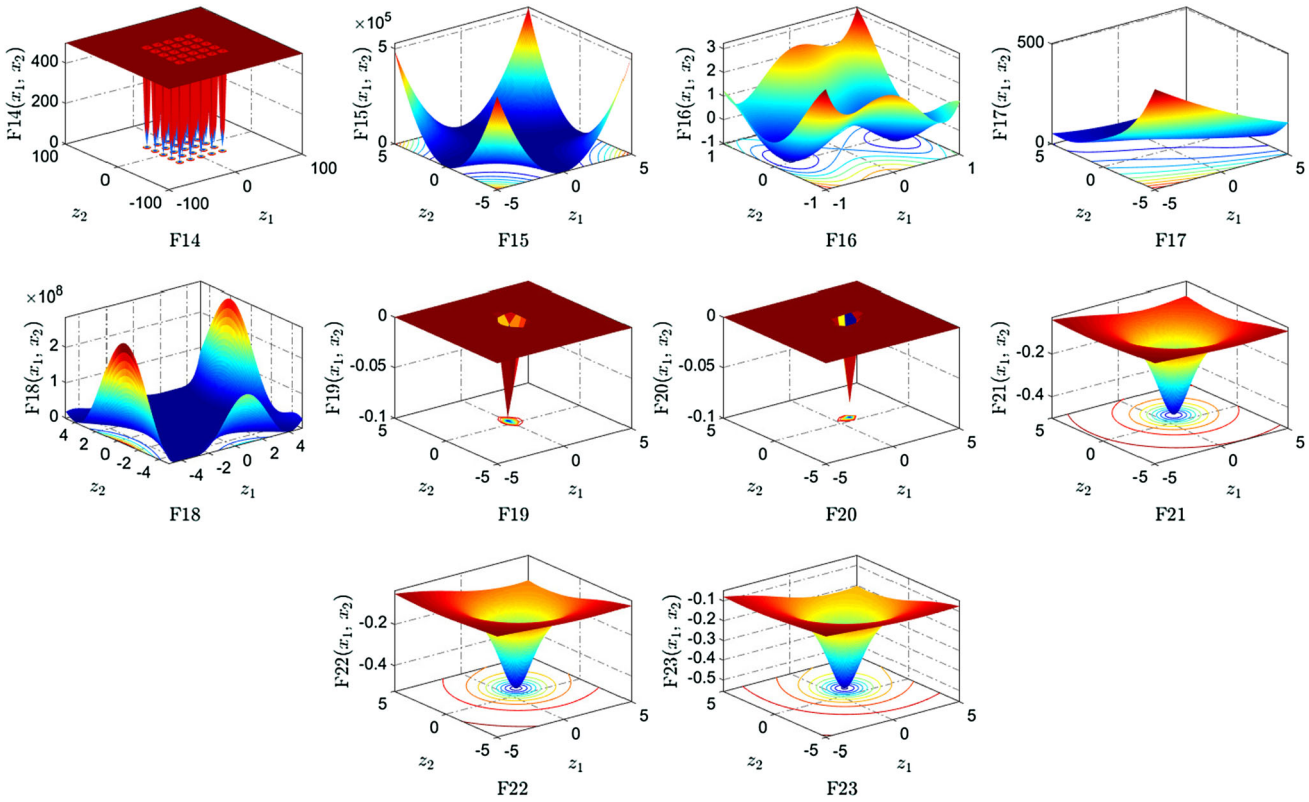

Fig. 3 Parameter space of the fixed-dimension multimodal benchmark functions

tabulated in Table 1. The parameter settings given in Table 1 are set according to the literature.

The number of search agents and the maximum number of iterations of ESSA and each algorithm is set to 1000 and 30, respectively. The stopping criterion for each algorithm is set according to the maximum number of iterations except for ESSA that is also considered a tolerance threshold.

In the proposed ESSA, we used the same initialization procedure as the other comparative algorithms. In this sense, the parameter settings for ESSA concerning the number of search agents and a maximum number of iterations are similar to each algorithm and are similar to what was reported in the literature. For other control parameters of each algorithm, the values in the latest version of each algorithm are employed to ensure the best performance. For instance, each algorithm has its control parameters and each algorithm is generally distinguished based on its solution update strategy, but these parameter settings are important to be the same to provide a fair comparison between ESSA and other algorithms.

4.3 Performance comparison

The mean and standard deviation (STD) measures were used to compare the performance of ESSA to those of the standard SSA and other optimization algorithms on different benchmark functions. Each of these algorithms is run 30 independent times on each test function, to compute meaningful statistics. Each run conducts 1000 times of iterations. The mean and STD results are computed at the last iteration of each algorithm on each test function to get the optimal solution and provide a fair comparison among all comparative algorithms.

4.3.1 Evaluation of unimodal functions

The functions F_1–F_7 are unimodal and allow algorithms to assess their exploitation capability. The mean and STD results obtained by ESSA, the standard SSA and other metaheuristic algorithms on unimodal Functions F_1–F_7 are given in Table 2. The best results are shown in bold throughout the paper.

Table 2 shows that the proposed ESSA is capable of delivering encouraging results, and outperforms the standard SSA and other algorithms in all of the unimodal functions. The better average values on these functions reveal that ESSA performs better than other algorithms on average and that the STD results reveal that this superiority is stable. Due to the existence of a single optimum in the unimodal functions, these functions can benchmark exploitation and convergence

Table 1 Parameter setup of the compared algorithms

Algorithm	Parameter	Value
ESSA	Number of generations	1000
	Search agents	1000
SSA (Mirjalili et al. 2017)	Number of generations	1000
	Search agents	30
MFO (Mirjalili 2015)	Logarithmic spiral	0.75
	Convergence constant	[− 1, − 2]
	Number of generations	1000
	Search agents	30
MVO (Mirjalili et al. 2016)	Traveling distance rate	[0.6, 1]
	Wormhole existence prob.	[0.2, 1]
	Number of generations	1000
	Search Agents	30
SCA (Mirjalili 2016)	Number of elites	2
	Number of generations	1000
	Search agents	30
GSA (Rashedi et al. 2009)	Alpha coefficient	20
	Gravitational constant	100
	Number of generations	1000
	Crossover and Mutation	0.9, 0.05
GA (Bonabeau et al. 1999)	Population size	30
	Number of generations	1000
	Harmony memory and rate	30, 0.95
HS (Geem et al. 2001)	Discrete set and fret width	17,700, 1
	Neighboring value rate	0.30
	Number of generations	1000

Table 2 Computed results for the unimodal function-based ESSA and other algorithms

Functions	ESSA		SSA		MFO		MVO	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
F_1	**3.23e−101**	**1.79e−101**	1.24e−08	2.67e−09	3.15E−04	5.99E−04	2.81E−01	1.11E−01
F_2	**6.18e−101**	**1.20e−101**	2.44e−02	1.06e−01	3.71E+01	2.16E+01	3.96E−01	1.41E−01
F_3	**3.59e−101**	**2.04e−101**	1.93e−09	1.10e−09	4.42E+03	3.71E+03	4.31E+01	8.97
F_4	**3.88e−101**	**1.37e−101**	1.54e−05	3.20e−06	6.70E+01	1.06E+01	8.80E−01	2.50E−01
F_5	**4.57**	**1.10**	2.20e+02	4.87e+02	3.50E+03	3.98E+03	1.18E+02	1.43E+02
F_6	**3.08e−33**	**1.28e−08**	7.15e−10	1.84e−10	1.66E−04	2.01E−04	3.15E−01	9.98E−02
F_7	**1.61e−05**	**1.42e−04**	7.06e−03	4.99e−03	3.22E−01	2.93E−01	2.02E−02	7.43E−03
Functions	SCA		GSA		GA		HS	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
F_1	3.55E−02	1.06E−01	1.16E−16	6.10E−17	1.95E−12	2.01E−11	7.86E−10	8.11E−09
F_2	3.23E−05	8.57E−05	1.70E−01	9.29E−01	6.53E−18	5.10E−17	5.99E−20	1.11E−17
F_3	4.91E+03	3.89E+03	4.16E+02	1.56E+02	7.70E−10	7.36E−09	9.19E−05	6.16E−04
F_4	1.87E+01	8.21	1.12	9.89E−01	9.17E+01	5.67E+01	8.73E−01	1.19E−01
F_5	7.37E+02	1.98E+03	3.85E+01	3.47E+01	5.57E+02	4.16E+01	8.91E+02	2.97E+02
F_6	4.88	9.75E−01	1.08E−16	4.00E−17	3.15E−01	9.98E−02	8.18E−17	1.70E−18
F_7	3.88E−02	5.79E−02	7.68E−01	2.77	6.79E−04	3.29E−03	5.37E−01	1.89E−01

Table 3 Computed results for the multimodal functions based on the proposed algorithm and other algorithms

Functions	ESSA		SSA		MFO		MVO	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
F_8	− 2.2e+03	3.31e+02	− 2.81e+03	2.86e+02	− 8040	880	− 6920	919
F_9	**1.42e−14**	**6.70e−6**	1.58e+01	6.73	1.63E+02	3.74E+01	1.01E+02	1.89E+01
F_{10}	**8.88e−16**	**1.44e−15**	7.18e−01	8.21e−01	1.60E+01	6.18	1.15	7.87E−01
F_{11}	**1.11e−16**	**7.45e−02**	2.59e−01	1.29e−01	5.03E−02	1.74E−01	5.74E−01	1.12E−01
F_{12}	**4.71e−32**	**1.85e−02**	2.92e−01	5.95e−01	1.26	1.83	1.27	1.02
F_{13}	**1.34e−32**	**1.22e−01**	2.19e−03	4.47e−03	7.24E−01	1.48	6.60E−02	4.33E−02
Functions	SCA		GSA		GA		HS	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
F_8	− 3810	283	− 2750	572	− 5110	437	**− 469**	**394**
F_9	2.23E+01	3.25E+01	3.35E+01	1.19E+01	1.23E−01	4.11E+01	4.85E−02	3.91E+01
F_{10}	1.55E+01	8.11	8.25E−09	1.90E−09	5.31E−11	1.11E−1	2.83E−08	4.34E−07
F_{11}	3.01E−01	2.89E−01	8.19	3.70	3.31E−06	4.23E−05	2.49E−05	1.34E−04
F_{12}	5.21E+01	2.47E+02	2.65E−01	3.14E−01	9.16E−08	4.88E−07	1.34E−05	6.23E−04
F_{13}	2.81E+02	8.63E+02	5.73E−32	8.95E−32	6.39E−02	4.49E−02	9.94E−08	2.61E−07

speed of algorithms. Therefore, the results in Table 2 show that ESSA avails from high exploitation and convergence speed of the unimodal functions.

4.3.2 Evaluation of multimodal functions

Multimodal functions may involve many local optima that can grow exponentially. These functions have the ability to assess the exploration capability of algorithms. Table 3 shows the mean and STD results obtained by ESSA and other algorithms over 30 independent runs for the functions F_8–F_{13}.

As per the results in Table 3, it may be observed that ESSA again reports significantly better results than those obtained by SSA and other algorithms for most of the test functions including F_9–F_{13}, and it is very competitive in the other test functions. These findings confirm that ESSA has a high value in terms of exploration power. The better results can be observed in both mean and STD measures, which point out how robust and well ESSA is when addressing these functions.

4.3.3 Evaluation of fixed-dimension functions

The mean and STD results of the fixed-dimension multimodal benchmark functions, F_{14} to F_{23}, obtained by ESSA and other algorithms over 30 independent runs, are presented in Table 4.

The results in Table 4 show that the proposed ESSA has arrived at average results for the functions F_{14}–F_{19} which are considerably better than those obtained by the standard SSA. There is no significant difference between the results of the proposed ESSA and those obtained by the standard SSA on functions F_{20}, F_{21}, F_{22} and F_{23}. Moreover, the difference is very small on functions F_{15} and F_{16}. Moreover, the mean results reported by ESSA outperform those reported by other algorithms. This confirms that ESSA has successfully identified the best global solutions and can efficiently explore the search space. This high exploration gives ESSA the power to avoid many local optima in a fixed-dimension multimodal search space.

In short, the average results and small STD values in Tables 2, 3 and 4 affirm that ESSA is robust, is stable and has large capabilities in exploring and exploiting the search space, even if the search space is complex.

4.4 Convergence analysis

Convergence curves are the most widespread qualitative results for optimization algorithms in the literature. In convergence curves, the best solutions obtained so far are generally stored at each iteration loop of the meta-heuristic algorithms. Thereafter, each convergence curve is drawn as a line to be eligible to describe how well an algorithm approximates the global optimum solution over a predetermined number of iterations. The convergence analysis aims to provide a deeper understanding of the exploitation and exploration processes of the proposed ESSA. Moreover, the convergence analysis results can judge the performance of ESSA compared to other algorithms. The convergence curves of ESSA, SSA, MFO, MVO, SCA, GSA, GA and HS for some benchmark test functions are presented in Fig. 4.

Table 4 Computed results for the fixed-dimension multimodal functions based upon ESSA and other algorithms

Functions	ESSA		SSA		MFO		MVO	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
F_{14}	**0.998**	**3.62e−02**	1.031	1.81483e−01	2.21	1.80	0.998	9.14E−1
F_{15}	**3.07e−04**	**4.64e−04**	2.13e−03	4.96e−03	1.58E−03	3.50E−03	7.15E−03	1.26E−02
F_{16}	**−1.0316**	**5.83e−16**	−1.0316	1.44e−14	−1.03	0.00	−1.03	4.74E−08
F_{17}	**3.97e−01**	**0.0**	8.22e−01	7.30e−02	3.98E−01	1.13E−16	3.98E−01	1.15E−07
F_{18}	**3.0**	**1.04e−17**	3.0	7.31e−14	3.00	4.25E−15	5.70	1.48E+01
F_{19}	−3.85	3.77e−13	−3.86	3.77e−14	−3.86	3.16E−15	−3.86	3.53E−07
F_{20}	−3.19	1.97e−04	−3.24	5.79e−02	−3.23	6.65E−02	−3.23	5.37E−02
F_{21}	−10.1	2.60	−7.55	3.32	−6.20	3.52	−7.38	2.91
F_{22}	−10.4	2.53	−8.80	2.98	−7.95	3.20	−8.50	3.02
F_{23}	−10.5	2.59	−9.48	2.43	−7.50	3.68	−8.41	3.13

Functions	SCA		GSA		GA		HS	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
F_{14}	1.26	6.86E−01	3.61	2.96	4.39	4.41E−02	6.79	1.12
F_{15}	1.01E−03	3.75E−04	6.84E−03	7.37E−03	7.36E−03	2.39E−04	5.15E−03	3.45E−04
F_{16}	−1.03	3.23E−05	−1.03	0.00	−1.04	4.19E−07	−1.03	3.64E−08
F_{17}	3.99E−01	7.61E−04	3.98E−01	1.13E−16	3.98E−01	3.71E−17	3.99E−01	9.45E−15
F_{18}	3.00	2.25E−05	3.01	3.24E−02	3.01	6.33E−07	3.00	1.94E−10
F_{19}	−3.86	2.55E−03	**−3.22**	**4.15E−01**	−3.30	4.37E−10	−3.29	9.69E−04
F_{20}	−2.84	3.71E−01	**−1.47**	**5.32E−01**	−2.39	4.37E−01	−2.17	1.64E−01
F_{21}	**−2.28**	**1.80**	−4.57	1.30	−5.19	2.34	−7.33	1.29
F_{22}	−3.99	1.99	−6.58	2.64	−2.97	1.37E−02	**−1.00**	**2.89E−04**
F_{23}	−4.49	1.96	−9.37	2.75	−3.10	2.37	**−2.46**	**1.19**

The convergence curves in Fig. 4 divulge that ESSA delivers encouraging convergence responses in all of the evaluated test functions. In these curves, ESSA exhibits three featured converging behaviors while optimizing these benchmark functions. In the first few iteration steps, ESSA experiences sudden changes and converges very slowly to reach the most promising areas of the search space due to the adaptive self-parameter setting mechanism of the parameter c_1. This means that ESSA has a modest feebleness in exploring the search space in the first few iterations. This behavior is evident in functions F_9 and F_{11}. In the second converging behavior and beyond the next 150 iterations, ESSA gradually converges toward the global optimal solutions. This means that the exploration process of ESSA is fast and capable of locating the global solution using a modest number of iterations. This behavior is clearly evident in functions F_1–F_4. The third converging behavior can be inferred in the final iteration steps, where there is a clear convergence to find the global optimal solution or near-global optimal solution (i.e., food source) as obviously observed in functions F_{10}, F_{13} and F_{15}. This efficiency of ESSA in the convergence curves is attributed to the proposed self-tuning parameter method, which allows the individuals of ESSA to explore the search space and move around the food source to locate the global optimum.

By comparing the convergence curves obtained by ESSA to those obtained by other algorithms, ESSA provides a rapid convergence than other algorithms in most of the test functions such as F_1–F_4 and F_9–F_{11}, and it is very competitive to others in the other test functions. On the other hand, GSA presented promising convergence curves better than ESSA in some functions such as F_5, F_7, F_{12}, F_{13} and F_{15}.

Figure 5 shows the plot of c_1 and the variations of λ_1, λ_2 and λ_3 over the course of iterations of ESSA for some unimodal, multimodal and fixed-dimension multimodal functions. The values of these parameters used to plot c_1 were taken as 4, 0.44 and 1.1, respectively, where these values represent the optimal values for these tuned parameters in most of the optimization problems. Figure 5 shows that the variations of the parameters λ_1, λ_2 and λ_3 are very small at the beginning, and then, the values of these parameters fluctuate relatively large at the middle before approaching to small variations at the end. This narrow variation range of these parameters implies that the search agents of ESSA exploit each promising area to find the global optimum solution. These parameters need to reduce gradually in terms of

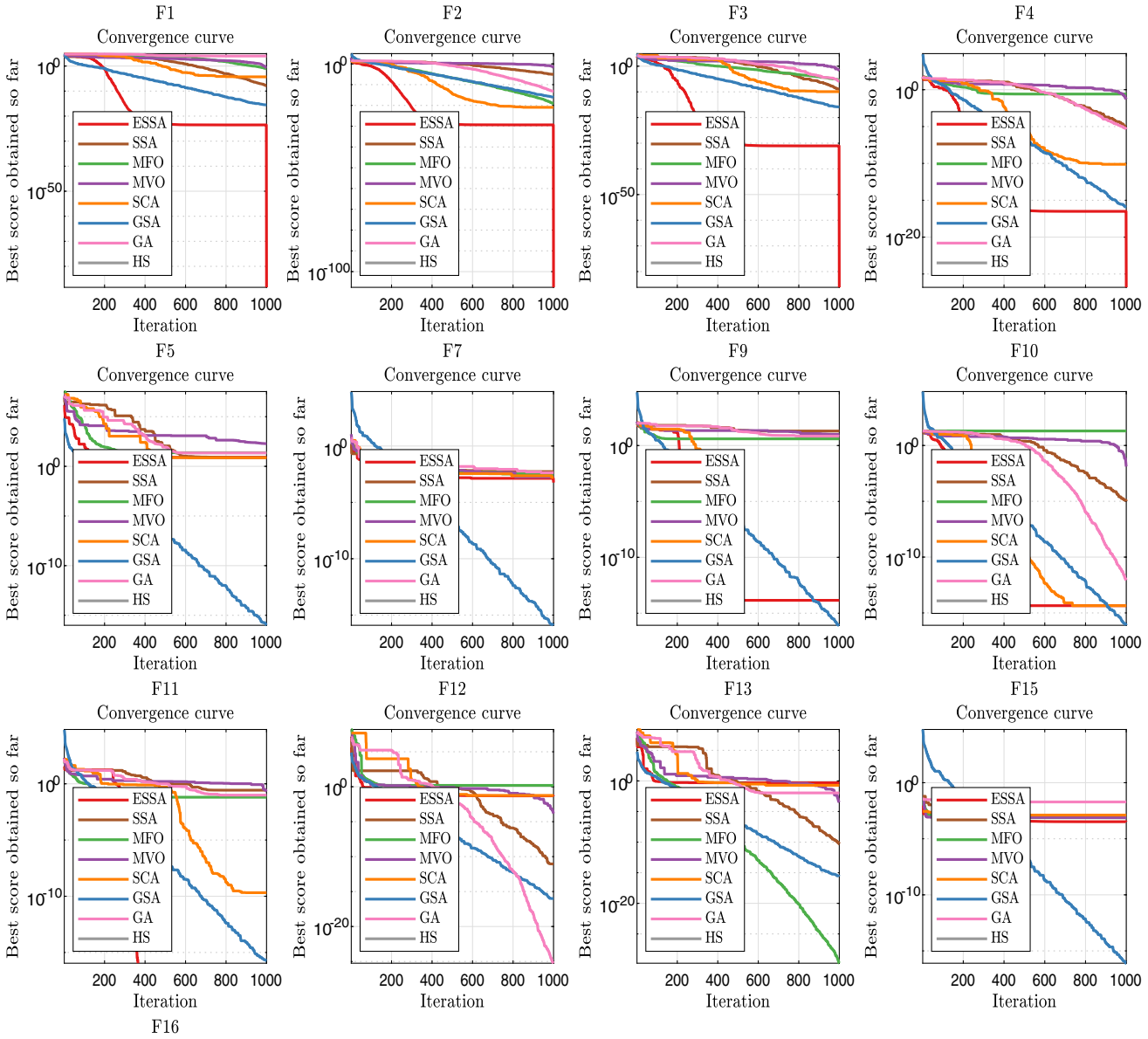

Fig. 4 Comparison of convergence curves of ESSA and other meta-heuristic algorithms obtained for some benchmark functions

θ_1, θ_2 and θ_3, respectively. However, the best settings for these parameters mostly depend on the nature of the problem. The following comments can be drawn based upon these results:

- The optimal setting for λ_1, λ_2 and λ_3 in c_1 often depends on the nature of the problem, and there is no unique best setting for these parameters for all problems. However, Fig. 5 shows that proper values for these parameters can be chosen as 4 ± 0.1, 0.44 ± 0.1 and 1.1 ± 0.1, respectively. The choice of these values should work appropriately for most problems, but not all problems.
- The small variations of the parameters λ_1, λ_2 and λ_3 mean that the actual setting of these parameters is not critical to a given problem, and therefore, there is no need to use a very large number of iterations and search agents to reach very optimal settings.
- Some parameters are more sensitive than others. In the present case, λ_3 needs more fine-tuning than λ_1 and λ_2, due to its smaller variations.

4.5 Statistical test analysis

Statistical tests are necessary to ensure that the results in Tables 2, 3, 4 are not generated by chance. These tests are essential to demonstrate the reliability and accuracy of ESSA in obtaining the average and STD results in all of the 30 independent runs. To compare the average results of the algorithms presented in Tables 2, 3 and 4, and emphasize the

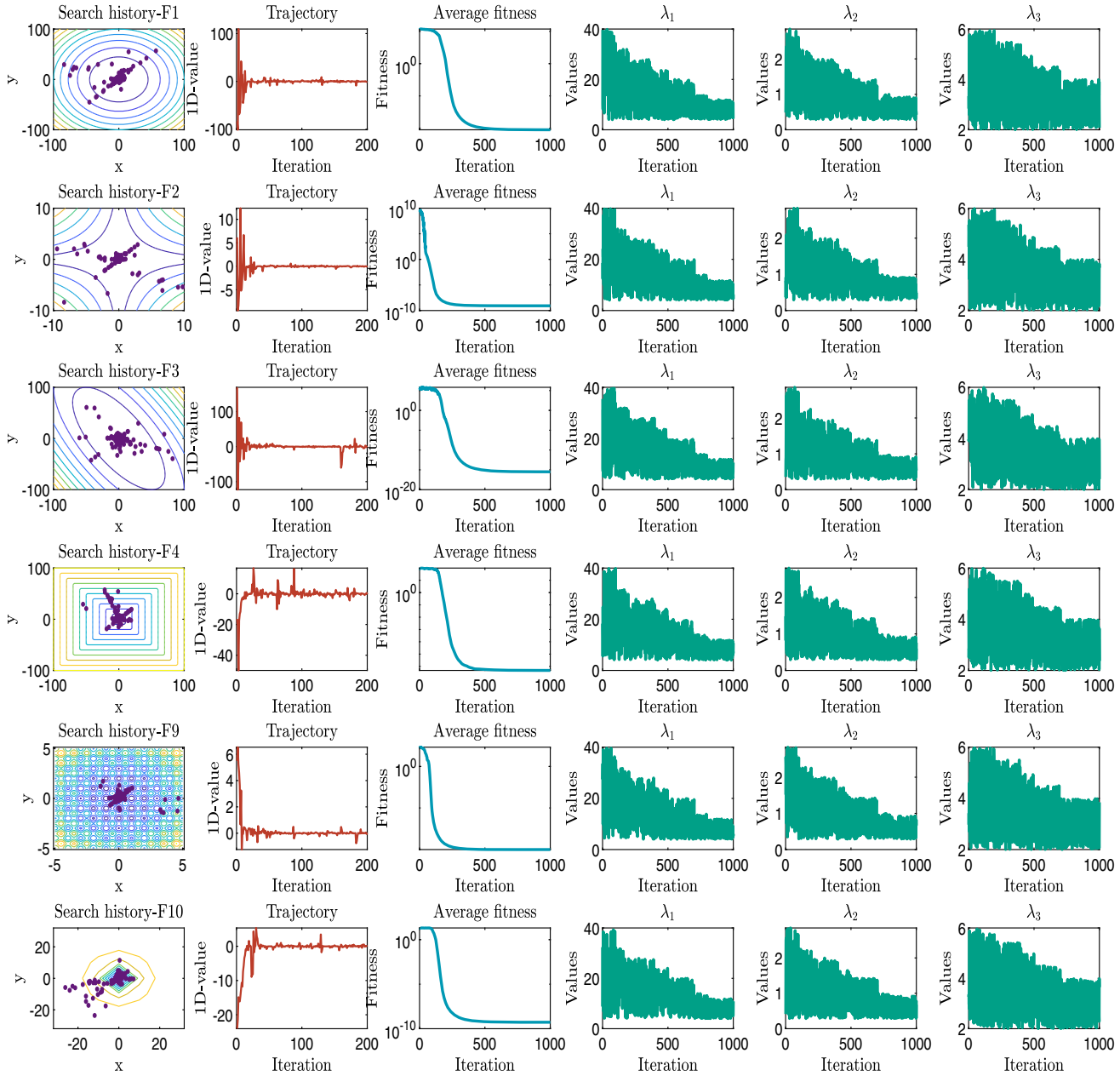

Fig. 5 Plot curves for the parameter c_1 ($\lambda_1 = 4.0$, $\lambda_2 = 0.44$ and $\lambda_3 = 1.1$) and the variations of these parameters over the course of iterations

significance of the results, statistical tests including Friedman's and Holm's tests (Pereira et al. 2015) are conducted to identify whether the results obtained by ESSA deviate from the results of other algorithms in a statistically significant manner.

Friedman's test aims to determine whether there is a considerable difference between the findings of the various algorithms. This test is based on a null hypothesis that there is no remarkable difference in the performance of the comparative algorithms. The best performing algorithm gets the lowest rank, while the worst performing algorithm obtains the highest rank. In case the p value divulged by Friedman's statistical test is equal or less than the significance level, here is 0.05, the null hypothesis is rejected, indicating that there are significant differences between the accuracy of the comparative algorithms. Friedman's test is typically followed by a post hoc test such as Holm's method to consider a pairwise comparison of the algorithms. In general, the lowest ranked algorithm is used as a control algorithm for the post hoc test method. A summary of the statistical test results obtained using Friedman's test on the mean results given in Tables 2, 3 and 4 is presented in Table 5.

According to the statistical results in Table 5, ESSA is the best performing algorithm among all other algorithms. Look-

Table 5 A summary of the statistical results obtained using Friedman's test on the results shown in Tables 2, 3 and 4

Algorithm	l	Rank
ESSA	**1.760869**	
SSA	3.652173	
MFO	5.195652	
MVO	4.934782	
SCA	5.695652	
GSA	5.239130	
GA	4.543478	
HS	4.978260	

Table 6 Results of Holm's method based on the statistical mean results of the unimodal, multimodal and fixed-dimension multimodal benchmark functions

i	Method	p value	$\alpha \div i$	Hypothesis
7	SCA	5.109424E−8	0.007142	Rejected
6	GSA	1.468802E−6	0.008333	Rejected
5	MFO	1.982096E−6	0.01	Rejected
4	HS	8.417653E−6	0.0125	Rejected
3	MVO	1.112409E−5	0.016666	Rejected
2	GA	1.169909E−4	0.025	Rejected
1	SSA	0.008834	0.05	Rejected

ing at the significance level of $\alpha = 0.05$, ESSA is ranked first in the test functions, followed in turn by SSA, GA, MVO, HS, MFO, GSA and SCA in the last rank.

The p value calculated by Friedman's test based on the mean results in Tables 2, 3 and 4 is 4.06632E−7. The null hypothesis of equivalent performance is rejected to affirm if there are statistically considerable differences between the performance of the comparative algorithms. Holm's method is then conducted as a posttest procedure to determine whether there are statistically significant differences between ESSA (i.e., the control algorithm) and other algorithms. The statistical results obtained using Holm's method are given in Table 6.

The results of Friedman's and Holm's test methods in Tables 5 and 6 illustrate the satisfactory performance and reliability of ESSA. These results demonstrate that ESSA will be proficient in solving complex real-world problems.

5 ESSA for engineering design problems

The efficiency of the proposed ESSA in solving real-world problems is demonstrated by experimenting with well-known examples of constrained engineering design problems. In the following subsections, ESSA was presented to solve five engineering design problems: the welded beam design problem, the tension/compression spring design problem, the three-bar truss design problem, the I-beam design problem and the cantilever beam design problem. These design problems have a range of constraints, so there is a need to equip ESSA with a constraint handling method. In this paper, to conduct a fair comparison between ESSA and algorithms reported in the literature, a death penalty function was used in a way analogous to that described in Yang (2010b). This constraint handling method has a low computational cost and does not use the information of infeasible solutions. The parameter setting of ESSA in solving the design problems below is the same as those presented in Table 1.

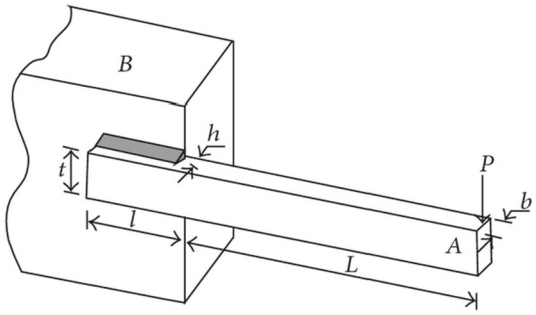

Fig. 6 A schematic structure of the welded beam

5.1 Welded beam design problem

The main goal of this classical design problem is to design a welded beam for the structure shown in Fig. 6 to achieve the minimum fabrication cost (Wang et al. 2014).

As shown in Fig. 6, the welded beam composes of a beam A and a weld required to attach it to the member B. This design problem is subject to four constraints: shear stress (τ), bending stress in the beam (θ), buckling load on the bar (P_c) and end deflection of the beam (δ). To solve this problem, there is a need to find the possible combination of the four structural parameters of the welded beam design: thickness of the weld (h), length of the clamped bar (l), height of the bar (t) and thickness of the bar (b). These structural design parameters can be represented by a vector as: $\vec{x} = [x_1, x_2, x_3, x_4]$, where x_1, x_2, x_3 and x_4 represent h, l, t and b, respectively. The mathematical formula of the cost function to be minimized for this design problem can be defined as follows:

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{23}$$

Subject to the following constraints,

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \leq 0$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \leq 0$$
$$g_3(\vec{x}) = x_1 - x_4 \leq 0$$
$$g_4(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$$

Table 7 A comparison of the results obtained by ESSA and other algorithms for the welded beam problem

Algorithm	Optimal values for variables				Optimum cost
	h	l	t	b	
ESSA	0.205730	3.470489	9.036624	0.205730	**1.724852**
SSA (Mirjalili et al. 2017)	0.2057	3.4714	9.0366	0.2057	1.72491
MVO (Mirjalili et al. 2016)	0.205611	3.472103	9.040931	0.205709	1.725472
SCA (Mirjalili 2016)	0.204695	3.536291	9.004290	0.210025	1.759173
MFO (Mirjalili 2015)	0.203567	3.443025	9.230278	0.212359	1.732541
CS (Gandomi et al. 2013)	0.2015	3.562	9.0414	0.2057	1.73121
ACO (Kaveh and Talatahari 2010)	0.205700	3.471131	9.036683	0.205731	1.724918
GSA (Rashedi et al. 2009)	0.18219	3.856979	10.0000	0.202376	1.879952
IPSO (Cagnina et al. 2008)	0.205729	3.470488	9.036624	0.205729	**1.724852**
ABC (Karaboga and Basturk 2008)	0.205730	3.470489	9.036624	0.205730	**1.724852**
ES (Mezura-Montes and Coello 2008)	0.812500	0.437500	42.098087	176.640518	6059.7456
Coevolutionary PSO (He and Wang 2007)	0.20573	3.47049	9.03662	0.20573	1.72485084
DE (Mezura-Montes et al. 2007)	0.20573	3.470489	9.0336624	0.205730	1.724852
SA (Hedar and Fukushima 2006)	0.20564426	3.472578742	9.03662391	0.2057296	1.7250022

$$g_5(\vec{x}) = 0.125 - x_1 \leq 0$$
$$g_6(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \leq 0$$
$$g_7(\vec{x}) = P - P_c(\vec{x}) \leq 0$$

where the other parameters of the beam design are determined as follows:

$$\tau(\vec{x}) = \sqrt{((\tau')^2 + (\tau'')^2) + \frac{2\tau'\tau''x_2}{2R}}, \tau' = \frac{p}{\sqrt{2}x_1x_2}$$
$$\tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{(\frac{x_1 + x_3}{2})^2 + \frac{x_2^2}{4}}$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2\right]\right\}, \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}$$
$$\delta(\vec{x}) = \frac{4PL^3}{Ex_4x_3^3}, P_c(\vec{x}) = \frac{4.013\sqrt{EGx_3^2x_4^6/36}}{L^2}\left(1 - \frac{x^3}{2L}\sqrt{\frac{E}{4G}}\right)$$

where $P = 6000$ lb, $L = 14$ in., $\delta_{\max} = 0.25$ in., $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\delta_{\max} = 13{,}600$ psi, $\sigma_{\max} = 30{,}000$ psi.

In this design problem, the ranges of the variables h, l, t and b were taken as $0.1 \leq x_1 \leq 2$, $0.1 \leq x_2 \leq 10$, $0.1 \leq x_3 \leq 10$ and $0.1 \leq x_4 \leq 2$, respectively. The welded beam design problem was solved by many algorithms such as SSA (Mirjalili et al. 2017), MVO (Mirjalili et al. 2016), SCA (Mirjalili 2016), MFO (Mirjalili 2015), CS algorithm (Gandomi et al. 2013), ACO (Kaveh and Talatahari 2010), GSA (Rashedi et al. 2009), improved PSO (IPSO) (Cagnina et al. 2008), ABC (Karaboga and Basturk 2008), evolutionary strategy (ES) (Mezura-Montes and Coello 2008), coevolutionary PSO (He and Wang 2007), differential evolution (DE) algorithm (Mezura-Montes et al. 2007) and simulated annealing (SA) (Hedar and Fukushima 2006). The comparative results for the best solution obtained by ESSA and other algorithms for this design problem are shown in Table 7. The results in Table 7 indicate that ESSA produced an optimal design for the welded beam problem by finding the optimal cost of approximately 1.72485, which is similar to the cost obtained by IPSO (Cagnina et al. 2008) and ABC (Karaboga and Basturk 2008).

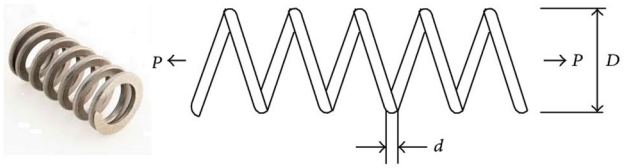

Fig. 7 A schematic of the tension/compression spring problem

5.2 Tension–compression spring design problem

The second engineering problem is a tension/compression spring with three parameters as shown in Fig. 7.

The goal of this design problem is to reduce the weight of a tension/compression spring design. This problem is subject to a number of constraints, including shear stress, surge frequency and minimum deflection. The parameters in this design problem can be defined as follows: wire diameter (d), mean coil diameter (D) and number of active coils (N), which are, respectively, represented by vector parameters as $\vec{x} = [x_1, x_2, x_3]$. The function $f(\vec{x}) = (x_3 + 2)x_2x_1^2$ to be optimized is subject to the following constraints:

Table 8 A comparison of the results obtained by ESSA and other algorithms for the tension/compression spring design problem

Algorithm	Optimum variables			Optimum weight
	d	D	N	
ESSA	0.051592	0.354396	11.426390	**0.0126654**
SHO (Dhiman and Kumar 2017)	0.051144	0.343751	12.0955	0.012674000
SSA (Mirjalili et al. 2017)	0.051207	0.345215	12.004032	0.0126763
GWO (Mirjalili et al. 2014)	0.05169	0.356737	11.28885	**0.0126665**
MVO (Mirjalili et al. 2016)	0.05000	0.315956	14.22623	0.012816930
SCA (Mirjalili 2016)	0.050780	0.334779	12.72269	0.012709667
MFO (Mirjalili 2015)	0.05000	0.313501	14.03279	0.012753902
GSA (Rashedi et al. 2009)	0.050276	0.323680	13.525410	0.0127022
ES (Mezura-Montes and Coello 2008)	0.051989	0.363965	10.890522	0.0126810
Coevolutionary PSO (He and Wang 2007)	0.051728	0.357644	11.244543	0.0126747
DE (Huang et al. 2007)	0.051609	0.354714	11.410831	0.0126702
IHS (Mahdavi et al. 2007)	0.05025	0.316351	15.23960	0.012776352

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

The ranges of the parameters, d, D and N, were taken as follows: $0.05 \leq x_1 \leq 2.0$, $0.25 \leq x_2 \leq 1.3$ and $2.0 \leq x_3 \leq 15.0$, respectively. This problem was widely addressed in the literature by many algorithms such as spotted hyena optimizer (SHO) (Dhiman and Kumar 2017), SSA (Mirjalili et al. 2017), GWO (Mirjalili et al. 2014), MVO (Mirjalili et al. 2016), SCA (Mirjalili 2016), MFO (Mirjalili 2015), GSA (Rashedi et al. 2009), ES (Mezura-Montes and Coello 2008), coevolutionary PSO (He and Wang 2007), DE (Huang et al. 2007) and improved HS (IHS) (Mahdavi et al. 2007). A comparison of the best solution obtained by ESSA and other algorithms for this design problem is shown in Table 8. The results in Table 8 reveal the merits of ESSA in solving this problem by producing an optimal design. The proposed ESSA obtained a design for this problem with a cost of about 0.0126654, which is slightly better than the design obtained by similar algorithms such as GWO (Mirjalili et al. 2014).

5.3 Three-bar truss design problem

The three-bar truss engineering design problem (Gandomi et al. 2013) consists of two design variables, where the structural parameters in this problem are shown in Fig. 8.

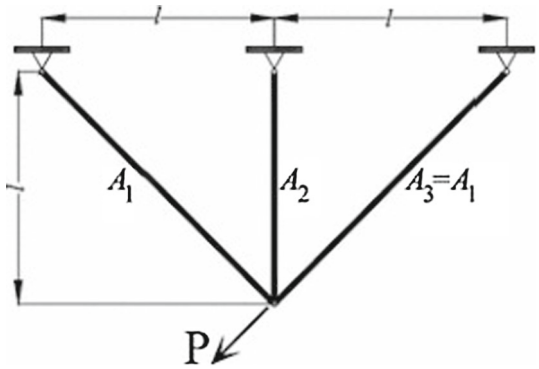

Fig. 8 Three-bar truss design problem

The aim of this problem is to design a truss with three bars to minimize the volume of the truss structure for an objective function setup as:

$$f(\vec{x}) = \left(2\sqrt{2}x_1 + x_2\right) L \tag{24}$$

This problem is subject to the following constraints:

$$g_1(\vec{x}) = \left(\frac{2\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1 x_2}\right) P \leq 2$$

$$g_2(\vec{x}) = \left(\frac{1}{x_1 + \sqrt{2}x_2}\right) P \leq 2$$

$$g_3(\vec{x}) = \left(\frac{2}{\sqrt{2}x_1^2 + 2x_1 x_2}\right) P \leq 2$$

where $0 \leq x_1 \leq 1.0$ and $0 \leq x_2 \leq 1.0$. The other constants are $L = 100$ cm and $P = 2$ kN/cm^2. The three-bar truss design problem was solved by many researchers using different algorithms such as SSA (Mirjalili et al. 2017),

Table 9 A comparison of the results obtained by ESSA and other algorithms for the three-bar truss design problem

Algorithm	Optimal values for variables		Optimal weight
	x_1	x_2	
ESSA	0.788604	0.408450	**263.8958433**
SSA (Mirjalili et al. 2017)	0.78866541	0.40827578	263.8958434
CS (Gandomi et al. 2013)	0.78867	0.40902	263.9716
MBA (Sadollah et al. 2013)	0.7885650	0.4085597	263.8958522
PSO-DE (Liu et al. 2010)	0.7886751	0.4082482	**263.8958433**
DEDS (Zhang et al. 2008)	0.78867513	0.40824828	263.8958434

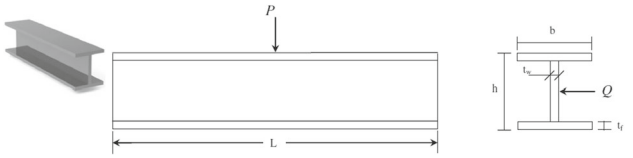

Fig. 9 I beam design problem

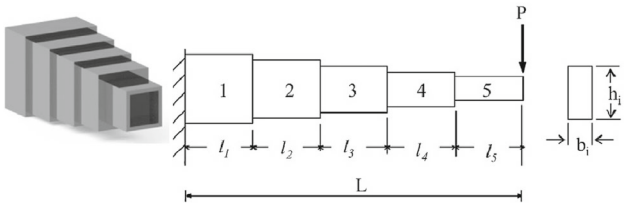

Fig. 10 A cantilever beam design problem

CS algorithm (Gandomi et al. 2013), mine blast algorithm (MBA) (Sadollah et al. 2013), PSO with DE (PSO-DE) (Liu et al. 2010) and DE with dynamic stochastic selection (DEDS) (Zhang et al. 2008). Table 9 shows a comparison of the best solutions obtained by ESSA and other algorithms previously reported in the literature for the three-bar truss design problem. According to Table 9, ESSA can find an optimal design for the three-bar truss problem with a cost of about 263.8958433.

5.4 I-beam design problem

The capability of ESSA was also used to solve the real I-beam engineering design problem that deals with four structural parameters (Gandomi et al. 2013). The main objective of this problem is to minimize the vertical deflection of the I-beam design shown in Fig. 9.

This design problem simultaneously meets the cross-sectional area and stress limitations under given loads. The four parameters of this problem are listed as: b, h, t_w and t_f, where the length of the beam (L) and modulus of elasticity (E) are 5.200 cm and 523.104 kN/cm^2, respectively. The objective function defined to minimize the vertical deflection $f(x) = PL^3/48EI$ was considered as given in the following formula:

$$\text{Minimize: } f(\vec{x}) = \frac{5000}{\frac{x_3(x_2-2x_4)^3}{12} + \frac{x_1 x_4^3}{6} + 2x_1x_4\frac{(x_2-x_4)^2}{4}}$$

The problem is subject to cross-sectional area less which would be than 300 cm^2, as defined below:

$$g_1(\vec{x}) = 2x_1x_3 + x_3(x_2 - 2x_4) \leq 300$$

where the design spaces of the variables are defined as: $10 \leq x_1 \leq 50, 10 \leq x_2 \leq 80, 0.9 \leq x_3 \leq 5.0$ and $0.9 \leq x_4 \leq 5.0$. There are several optimization algorithms in the literature previously presented to solve this design problem. These algorithms respected the aforementioned constraint and used the same range of design variables. The results of ESSA for the I-beam design problem design are

Table 10 A comparison of the results obtained by ESSA and other algorithms for the I-beam design problem

Algorithm	Optimal values for variables				Optimum vertical deflection
	b	h	t_w	t_f	
ESSA	50.0	80.0	1.764706	5.0	**0.0066259581655**
SSA (Mirjalili et al. 2017)	50.0	80.0	1.76470587	5.0	0.0066259581689
SOS (Cheng and Prayogo 2014)	50	80	0.9	2.32179	0.0130741
CS (Gandomi et al. 2013)	50	80	0.9	2.321675	0.0130747
ARSM (Wang 2003)	37.05	80.0	1.71	2.31	0.0157
IARSM (Wang 2003)	48.42	79.99	0.90	2.40	0.131

Table 11 A comparison of the results obtained by ESSA and other algorithms for the cantilever beam design problem

Algorithm	Optimal values for variables					Optimum weight
	x_1	x_2	x_3	x_4	x_5	
ESSA	6.016243	5.308456	4.494628	3.501176	2.153158	**1.33995634**
SSA (Mirjalili et al. 2017)	6.015134	5.309304	4.495006	3.501426	2.152787	1.33995639
SOS (Cheng and Prayogo 2014)	6.01878	5.30344	4.49587	3.49896	2.15564	1.33996
GCA$_I$ (Gandomi et al. 2013)	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
GCA$_I I$ (Gandomi et al. 2013)	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
CS (Gandomi et al. 2013)	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999

compared with the standard SSA (Mirjalili et al. 2017) and many other algorithms such as symbiotic organism search (SOS) (Cheng and Prayogo 2014), CS algorithm (Gandomi et al. 2013), adaptive response surface method (ARSM) (Wang 2003) and inherited ARSM (IARSM) (Wang 2003). A comparison of the best solutions obtained by ESSA and other algorithms for the I-beam design problem is presented in Table 9. Table 10 shows that ESSA is capable of finding an optimal design for the I-beam design problem with a minimum cost of 0.0066259581655.

5.5 Cantilever beam design problem

The fifth engineering design problem is a cantilever beam that consists of five elements, each with a hollow cross section with a constant thickness (Gandomi et al. 2013). The beam is strictly supported as shown in Fig. 10, and there is an external vertical force acting at the free end of the cantilever.

The goal of this problem is to minimize the weight of a cantilever beam while setting an upper limit on the vertical displacement of the free end. The design variables are the heights (or widths) of the cross section of each element. The lower limits of these variables are too small and the upper limits are too large so they do not become active in the problem. To solve this optimization problem, there is a need to find a possible combination of the five parameters of this design. The cost function of this design problem can be formulated as follows:

$$f(\vec{x}) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5)$$

where this problem is subject to the following constraint,

$$g_1(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3}$$

The ranges of the variables in this design problem were considered as follows: $1 \leq x_i \leq 10$, where $i \in 1, 2, 3, 4, 5$. Several algorithms in the literature were used to solve the cantilever beam design problem such as SSA (Mirjalili et al. 2017), SOS (Cheng and Prayogo 2014), generalized convex approximation (GCA$_I$, GCA$_I I$) (Gandomi et al. 2013) and CS (Gandomi et al. 2013). A comparison of the solutions obtained by ESSA and other algorithms for the cantilever beam design problem is presented in Table 11. The results in Table 11 affirm that ESSA produced an optimal design for the cantilever beam problem with an optimal cost of 1.33995634, which is the lowest cost compared to the others.

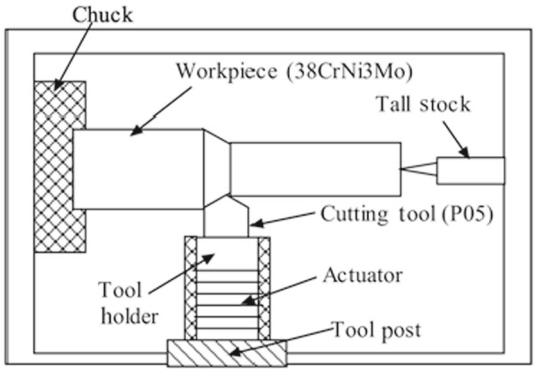

Fig. 11 A schematic diagram of the metal cutting system tool

6 Modeling of complex nonlinear systems

This section shows the applicability of ESSA to model two real industrial problems. These systems are a metal cutting system and an industrial winding machine system.

6.1 Metal cutting system

The system modeling in this problem aims to design a cutting temperature experiment for the metal cutting system presented in Fig. 11 (Zhou et al. 1998).

The metal cutting tool used here is P05 horny alloy steel and the workpiece metal is 38CrNi3Mo. A commonly cutting temperature model used for P05 is presented in Eq. (25) (Zhou et al. 1998; Faris and Sheta 2016).

$$T_r = k a^x f^y v^z \tag{25}$$

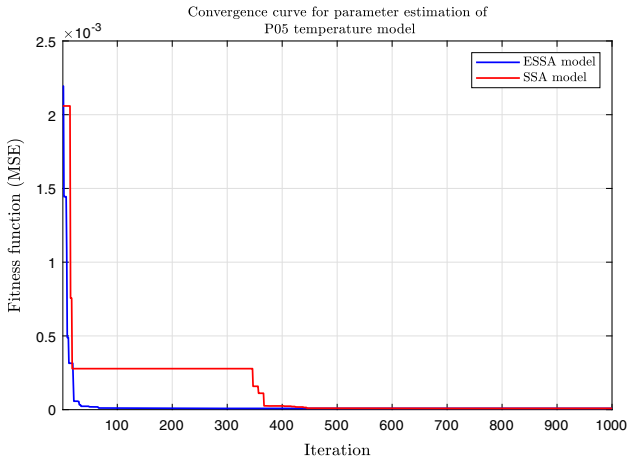

Fig. 12 Convergence curves over thirty experiments for parameter estimation of the cutting tool P05

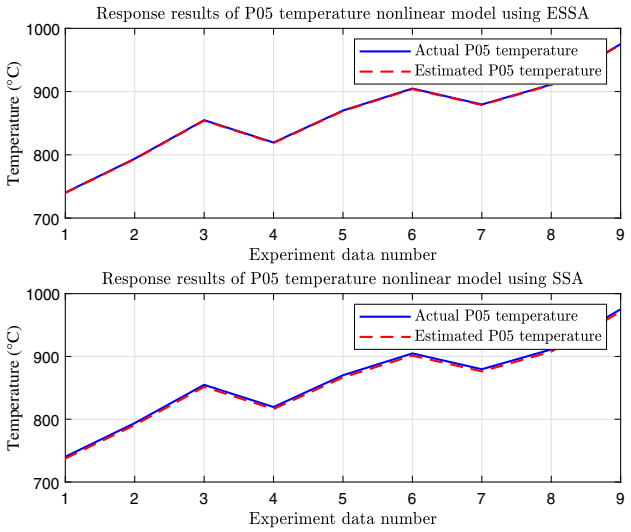

Fig. 13 Evaluation responses of the models developed by ESSA and SSA for the cutting tool P05

where T_r identifies the output of the temperature model, k is a parameter where its value depends on the machined material, a represents the depth of cut section in millimeters (mm), f represents the cutting feed rate in mm/rev, v identifies the speed of cutting in meter/minute (m/min) and x, y and z are three parameters whose values depend on the material of the cutting tool.

In this study, ESSA was used to develop a model for a cutting temperature system using publicly available data (Yi-jian et al. 2005). More specifically, ESSA was used to estimate k, x, y, z parameters of the P05 model shown in Eq. (25). The main goal of the developed model is to minimize the error between the actual metal cutting temperature system (T) and the counterpart estimated model (T_r). The fitness function is defined in Eq. (26).

$$\text{fitness} = \sum_{i=1}^{n} [T(i) - T_r(i)]^2 \tag{26}$$

where n represents the number of data samples. The variance-accounted-for (VAF) measure defined in Eq. (27) was used to assess the performance of the developed models.

$$\text{VAF} = \left[1 - \frac{\text{var}(T - T_r)}{var(T)}\right] \times 100\% \tag{27}$$

where var(y) is the variance of the actual data of the metal cutting temperature system (T) and var($T - T_r$) is the variance of the difference between the actual data, T, and the estimated data (T_r). The convergence curve obtained by ESSA compared to that obtained by SSA in the modeling of P05 is shown in Fig. 12. Figure 12 shows that ESSA presented faster convergence better than the convergence provided by the standard SSA. The model responses of P05 along with the true P05 obtained by ESSA and SSA are presented in Fig. 13. Table 12 compares the VAF values and the estimated values of k, x, y, z, obtained by ESSA with those obtained by other algorithms such as SSA, PSO and GA. The findings in Table 12 confirm that ESSA has remarkably reported a better VAF rate than those reported by the standard SSA, PSO and GA.

6.2 An industrial winding process

The industrial winding process under consideration is usually encountered in real web conveyance systems (Bastogne et al. 1998; Rodan et al. 2017). The structural diagram of this process is shown in Fig. 14.

As shown in Fig. 14, the winding process targeted in this study consists of three reels, referred to as reel 1, reel 2 and reel 3. These reels are controlled by three DC motors known as M_1, M_2 and M_3, respectively. Reels 1 and 3 are coupled to DC motors driven by set-point currents I_1 at motor 1 and I_3 at motor 2. Moreover, tension meters are placed to measure strip tensions in the web between reel 1 and reel 2, referred to as T_1, and between reel 2 and reel 3, referred to as T_3. In this process, there is a dynamo tachometer to measure the angular speeds of reels 1, 2 and 3, referred to as S_1, S_2 and S_3), respectively. The angular velocity of motor M_2, or denoted as Ω_2, is measured using a dynamo tachometer. The main input variables of this process are: S_1, S_2, S_3, I_1 and I_3, and the outputs are: T_1 and T_3.

The training dataset for each model developed for T_1 and T_3 consisted of 1250 data samples for each input variable. The dataset in the testing process consisted of 1250 samples for each input variable. These datasets are described in Nozari et al. (2012). The goal of this modeling problem is to capture the main characteristics of the correct output responses for

Table 12 A comparison of the results obtained by ESSA and other algorithms for P05 of a metal cutting temperature system

Algorithm	Optimal values for variables				Optimal VAF value
	k	x	y	z	
ESSA	325.9	0.0415	0.0324	0.0215	**81.2314**
SSA	448.0	0.0362	0.0422	0.0457	66.0578
PSO	470.3	0.0324	0.0828	0.159	56.6956
GA	471.0	0.0301	0.083	0.159	56.9459

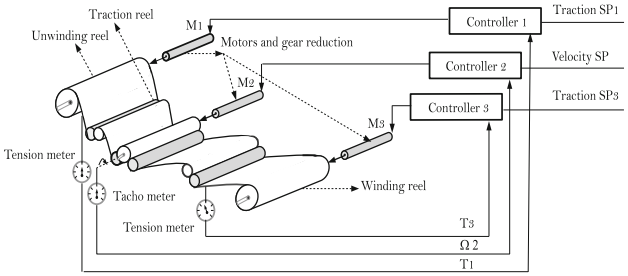

Fig. 14 A schematic diagram of the winding process

this process given its input and output parameters.To solve this problem using linear estimates, there is a need to create linear models for T_1 and T_3. These models are defined in Equations 28 and 28, respectively.

$$\begin{aligned} T_1(t) &= T_1(t-1) + \alpha_1 S_1(t) + \alpha_2 S_2(t) \\ &\quad + \alpha_3 S_3(t) + \alpha_4 I_1(t) + \alpha_5 I_3(t) \\ T_3(t) &= T_3(t-1) + \beta_1 S_1(t) + \beta_2 S_2(t) \\ &\quad + \beta_3 S_3(t) + \beta_4 I_1(t) + \beta_5 I_3(t) \end{aligned}$$

where $T_1(t-1)$ and $T_3(t-1)$, respectively, are the preceding values for T_1 and T_3 at time $t-1$, and α_1–α_5 and β_1–β_5 are the weights of the linear models developed for $T_1(t)$ and $T_3(t)$, respectively.

The mean absolute percentage error (MAPE) criterion, defined in Eq. (28), was used as a fitness function for this modeling problem.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{28}$$

where n represents the number of experimental data values, y identifies the actual values of a real experiment and $\hat{y}$ represents the estimated values generated by the models. In the development of T_1 and T_3 models, the maximum number of iterations and the number of search agents for ESSA and SSA were set to 1000 and 30, respectively. The convergence curves obtained by ESSA compared to the corresponding curves obtained by SSA in the modeling of T_1 and T_3 are shown in Fig. 15.

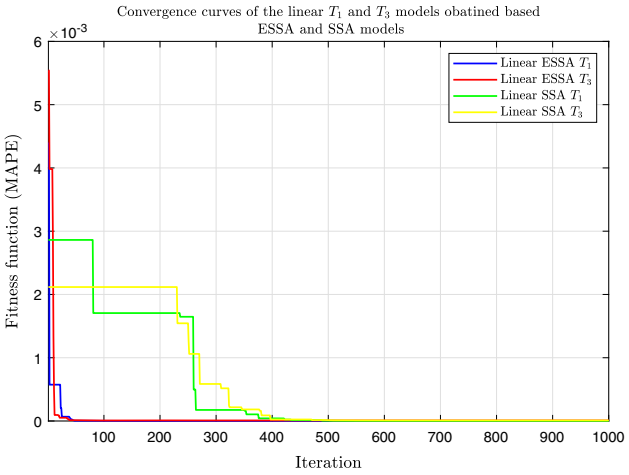

Fig. 15 Convergence curves over thirty experiments for linear T_1 and T_3 model-based ESSA and SSA

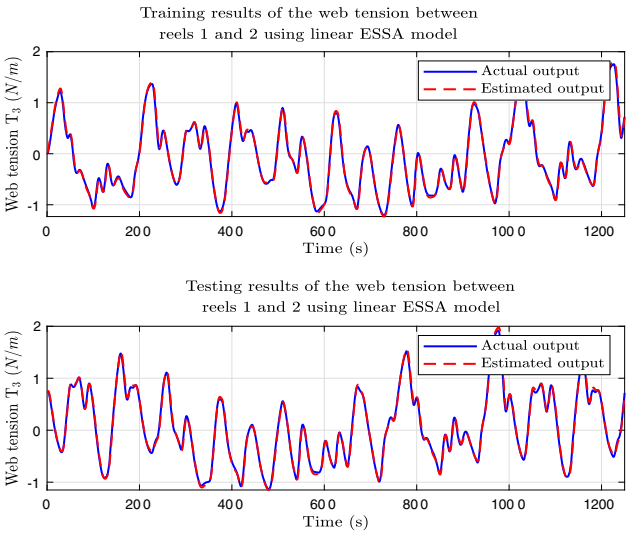

Fig. 16 Training and testing responses of the linear T_1 model developed by ESSA

The convergence curves in Fig. 15 confirm the reliability of the proposed ESSA in reaching the minimum errors over those obtained by SSA. The performance of ESSA in tracking the output data of T_1 and T_3 is shown in Fig. 16 and 17, respectively, along with the actual responses of T_1 and T_3, for both training and test cases.

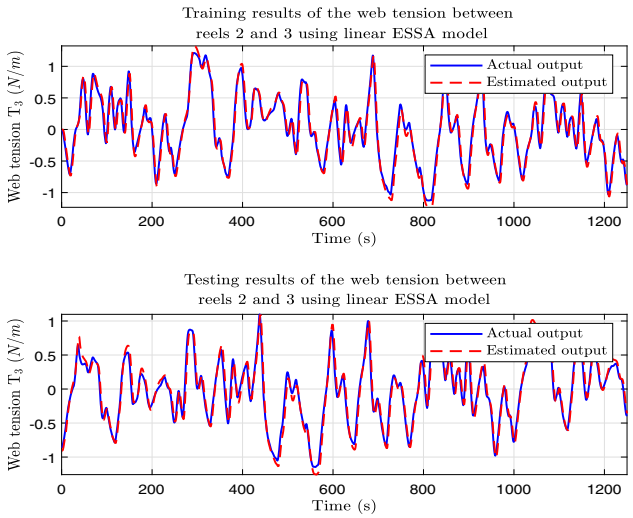

Fig. 17 Training and testing responses of the linear T_3 model developed by ESSA

The simulation results in Figs. 16 and 17 demonstrate the validity and appropriateness of ESSA in reliably modeling the web tension, T_1 and T_3, of the winding process. Table 13 presents a comparative study of the performance of the proposed ESSA and the standard SSA. According to results in Table 13, ESSA achieved VAF rates for T_1 and T_3 models significantly better than those obtained by the standard SSA.

7 Analysis of the results

The preceding sections used a set of benchmark functions, a set of five engineering problems and two real industrial problems with various characteristics to assertively measure and confirm the performance of ESSA. In this work, ESSA is different from SSA in that it uses the current positions of the search agents in the position updating mechanism. Also, the dominant parameter c_1 of ESSA does not depend on the maximum number of iterations as that of SSA. Moreover, in ESSA, we have successfully developed a novel lifetime convergence framework by self-tuning the parameter c_1 using ESSA itself. This is to find the best setting for c_1 so that ESSA can perform the best for a broad range of problems of varying complexity.

Inspecting the results of the unimodal, multimodal and fixed-dimension multimodal functions in Tables 2, 3 and 4, respectively, we may conclude that ESSA performs better than other algorithms in the majority of test functions. The higher mean values indicate that ESSA carries out better than others on average, and the standard deviation values confirm that this supremacy is stable. These results denote how excellently and robust ESSA is when solving these functions. Due to the existence of a single optimum in the unimodal functions, they can only measure exploitation and convergence of algorithms. Therefore, the results in Table 3 showed that ESSA takes advantage of the high exploitation and convergence speed of these functions.

As aforementioned, multimodal functions confront the presence of many local optimums in addition to more than one global optimum, so these functions are highly efficient to benchmark the ability of the algorithms to eschew local optimum and test their exploration capability. Therefore, the results in Tables 3 and 4 showed that ESSA benefits from large exploration.

This arises from the fact that the search agents of ESSA tend to communicate with each other, so they are not easily attracted to a local solution. The connection between the search agents of ESSA also allows it to explore the search space and gradually moves the salp chain toward the global optimal. The superior convergence of ESSA is attributed to the position updating process of salps around the best solutions obtained so far.

The entire salp chain converges toward the global optimum solution proportionate to the iteration number due to the proposed lifetime tuning mechanism for the parameter c_1 of ESSA based on ESSA itself. On the other hand, the results of the p values in Table 6 generated from Friedman's and Holm's tests show that the superiority of ESSA is statistically significant. The p values of Holm's test also evidence how important the proposed algorithm is on the functions F_1–F_{23}. The outcomes of Friedman's test also prove that the results of ESSA are statistically significant because the majority of the p values are less than the significance level (i.e., 0.05).

Despite the success of ESSA in solving basic benchmark functions of varying complexity, we conducted a more extensive test to assess its ability to solve engineering design problems and real industrial problems. This is to explore its reliability and verify the robustness of the proposed lifetime convergence scheme that is performed using the proposed self-tuning method. The results of ESSA on engineering design problems in Tables 7, 8, 9, 10 and 11, and industrial problems in Tables 12 and 13 show that ESSA produced stable performance and outperformed other algorithms. These engineering and industrial problems present very challenging test problems for any meta-heuristics search algorithm. This makes them very analogous to the actual search space that ESSA might encounter when solving real-world challenging and practical problems. Therefore, researchers always view meta-heuristic search algorithms as generally effective means to solve challenging problems of growing size. We know that each algorithm has its advantages such as flexibility, simplicity and limitations such as complexity and parameter setting. As analyzed and discussed above, ESSA has many key features; a natural question is: What are the limitations of the proposed ESSA? Based upon the position

Table 13 A comparison of the results obtained by ESSA and SSA in developing linear models for T_1 and T_3 over 30 runs

Algorithm	Weight parameters for T_1					Optimal VAF
	α_1	α_2	α_3	α_4	α_5	
ESSA	0.00342	0.00415	−0.05645	−0.01247	0.96597	**99.86**
SSA	0.02542	−0.01300	−0.09185	0.01184	−2.03375	98.03
Algorithm	Weight parameters for T_3					Optimal VAF
	β_1	β_2	β_3	β_4	β_5	
ESSA	0.00674	0.01365	0.01741	0.03116	−2.11876	**99.09**
SSA	−0.00367	0.03000	−0.02392	0.04320	−0.34492	97.39

updating process, the lifetime convergence scheme and the obtained results of ESSA, we can sum up the limitations of ESSA as given in the following subsection.

7.1 Limitations of the proposed ESSA

It is worth mentioning here that the exploitation of ESSA attends to be better than its exploration. This can be deduced from the results produced from the unimodal functions compared to those of the multimodal functions. This is related to the position updating proposed and the proposed self-tuning method of this algorithm, where sudden changes in the solutions are small at the beginning of the execution of this algorithm. Despite this fact, the results in Tables 2, 3 and 4 confirm that this is not a serious concern, as the explorative behavior of ESSA is also very plausible. Mere exploration does not ensure locating the global optimum solution, where an appropriate balance is needed between exploration and exploitation. The computation time of ESSA is not a critical issue as it can vary according to the nature and complexity of the problem. Several meta-heuristic algorithms use fixed pre-tuned parameters. Contrarily, ESSA uses one main control parameter, which is tuned using ESSA itself as iterations proceed. This offers a way for ESSA to automatically turn from exploration to exploitation as the optimal solution approaches. This known limitation of meta-heuristic algorithms was addressed and provides an advantage to ESSA over other algorithms.

8 Conclusion and future works

This paper has proposed an improved variant of the salp swarm algorithm (SSA) or referred to as enhanced SSA (ESSA). ESSA is generally distinguished based on its strategy to update the salps position and the proposed self-tuning convergence method. The major elements that contribute to the exploitation and convergence abilities of ESSA are the proposed self-tuning method for the main parameter of ESSA and the position updating mechanism that works on the best positions acquired so far in the search space. These featured elements in ESSA allow its salps to effectively explore the search space and identify the global optimum. To verify the efficacy of ESSA, many experiments were conducted on basic benchmark test functions. The outstanding performance of ESSA on these functions evinces the exploitation and exploration capabilities of ESSA in addition to its convergence behavior. The results of ESSA were compared with SSA and other well-known algorithms in terms of the average and standard deviation values. ESSA has proved to be an effective candidate that outperforms many other algorithms. The convergence behavior of ESSA was analyzed and found to be converging rapidly compared to others. The ability of ESSA in solving challenging problems is manifested by solving five well-known engineering design problems and two real industrial processes. The proposed self-tuning method-based ESSA could be effective for parameter control, so a more general framework for both parameter setting and control could be expanded for a broad range of real-world applications. Future work could be directed to substantiate the robustness of ESSA to address multi-objective, continuous and discrete optimization problems.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Appendix A: Objective test functions

A detailed description of the unimodal benchmark test functions (F_1–F_7), multimodal benchmark test functions (F_8–F_{13}) and fixed-dimension multimodal benchmark test functions (F_{14}–F_{23}) is given in Table 14.

Table 14 Characteristics of benchmark functions

Key	Function formulation	$f(x^*)$	Category	Dimension	Range
f_1	$\sum_{i=1}^{n} x_i^2$	0	U	10,30,50,100	$x_i \in [-100,100]$
f_2	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	0	U	10,30,50,100	$x_i \in [-10,10]$
f_3	$\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	0	U	10,30,50,100	$x_i \in [-100,100]$
f_4	$\max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	0	U	10,30,50,100	$x_i \in [-100,100]$
f_5	$\sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	0	U	10,30,50,100	$x_i \in [-30,30]$
f_6	$\sum_{i=1}^{n} ([x_i + 0.5])^2$	0	U	10,30,50,100	$x_i \in [-100,100]$
f_7	$\sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	0	U	10,30,50,100	$x_i \in [-128,128]$
f_8	$\sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	-418.9829×5	M	10,30,50,100	$x_i \in [-500,500]$
f_9	$\sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	0	M	10,30,50,100	$x_i \in [-5.12,5.12]$
f_{10}	$-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	0	M	10,30,50,100	$x_i \in [-32,32]$
f_{11}	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0	M	10,30,50,100	$x_i \in [-600,600]$
f_{12}	$\frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 - a & < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	0	M	10,30,50,100	$x_i \in [-50,50]$
f_{13}	$0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	0	M	10,30,50,100	$x_i \in [-50,50]$
f_{14}	$\left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	1	F	2	$x_i \in [-65,65]$
f_{15}	$\sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	0.0003	F	4	$x_i \in [-5,5]$
f_{16}	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	-1.0316	F	2	$x_i \in [-5,5]$
f_{17}	$\left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	0.398	F	2	$x_i \in [-5,5]$
f_{18}	$\left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	3	F	2	$x_i \in [-2,2]$
f_{19}	$-\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	-3.86	F	3	$x_i \in [1,3]$
f_{20}	$-\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	-3.32	F	6	$x_i \in [0,1]$
f_{21}	$-\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	-10.1532	F	4	$x_i \in [0,10]$
f_{22}	$-\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	-10.4028	F	4	$x_i \in [0,10]$
f_{23}	$-\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	-10.5363	F	4	$x_i \in [0.10]$

U: unimodal; *M*: multimodal; *F*: fixed-dimension multimodal

References

Abbassi R, Abbassi A, Heidari AA, Mirjalili S (2019) An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. Energy Convers Manage 179:362–372

Ali TAA, Xiao Z, Sun J, Mirjalili S, Havyarimana V, Jiang H (2019) Optimal design of IIR wideband digital differentiators and integrators using salp swarm algorithm. Knowl-Based Syst 182(104):834. https://doi.org/10.1016/j.knosys.2019.07.005

AlRashidi MR, El-Hawary ME (2008) A survey of particle swarm optimization applications in electric power systems. IEEE Trans Evol Comput 13(4):913–918

Ateya AA, Muthanna A, Vybornova A, Algarni AD, Abuarqoub A, Koucheryavy Y, Koucheryavy A (2019) Chaotic salp swarm algorithm for SDN multi-controller networks. Eng Sci Technol Int J 22(4):1001–1012

Bairathi D, Gopalani D (2019) Salp swarm algorithm (SSA) for training feed-forward neural networks. In: Bansal J, Das K, Nagar A, Deep K, Ojha A (eds) Soft computing for problem solving. Springer, Berlin, pp 521–534

Bastogne T, Noura H, Sibille P, Richard A (1998) Multivariable identification of a winding process by subspace methods for tension control. Control Eng Pract 6(9):1077–1088

Bonabeau E, Marco DdRDF, Dorigo M, Theraulaz G et al (1999) Swarm intelligence: from natural to artificial systems. 1. Oxford University Press, Oxford

Braik M, Sheta A, Aljahdali S (2019) Diagnosis of brain tumors in MR images using metaheuristic optimization algorithms. In: International conference Europe Middle East & North Africa information systems and technologies to support learning. Springer, pp 603–614

Cagnina LC, Esquivel SC, Coello CAC (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 32(3):319–326

Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112

Crawford B, Valenzuela C, Soto R, Monfroy E, Paredes F (2013) Parameter tuning of metaheuristics using metaheuristics. Adv Sci Lett 19(12):3556–3559

Dhiman G, Kumar V (2017) Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 114:48–70

Dobslaw F (2010) A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In: International conference on computer mathematics and natural computing. WASET

dos Santos Coelho L, Mariani VC (2012) Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Comput Math Appl 64(8):2371–2382

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, 1995. MHS'95. IEEE, pp 39–43

El Aziz MA, Ewees AA, Hassanien AE (2017) Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Syst Appl 83:242–256

Fallahi M, Amiri S, Yaghini M (2014) A parameter tuning methodology for metaheuristics based on design of experiments. Int J Eng Technol Sci 2(6):497–521

Faris H, Sheta A (2016) A comparison between parametric and non-parametric soft computing approaches to model the temperature of a metal cutting tool. Int J Comput Integr Manuf 29(1):64–75

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

Geem ZW, Sim KB (2010) Parameter-setting-free harmony search algorithm. Appl Math Comput 217(8):3881–3889

Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68

Greene CS, White BC, Moore JH (2008) Ant colony optimization for genome-wide genetic analysis. In: International conference on ant colony optimization and swarm intelligence. Springer, pp 37–47

He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99

Hedar AR, Fukushima M (2006) Derivative-free filter simulated annealing method for constrained continuous global optimization. J Global Optim 35(4):521–549

Hegazy AE, Makhlouf M, El-Tawel GS (2018) Improved salp swarm algorithm for feature selection. J King Saud Univ - Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2018.06.003

Huang F, Wang L, He Q (2007) An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 186(1):340–356

Ibrahim RA, Ewees AA, Oliva D, Abd Elaziz M, Lu S (2019) Improved salp swarm algorithm based on particle swarm optimization for feature selection. J Ambient Intell Humaniz Comput 10(8):3155–3169. https://doi.org/10.1007/s12652-018-1031-9

Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175

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

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697

Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput 27(1):155–182

Khadwilard A, Chansombat S, Thepphakorn T, Chainate W, Pongcharoen P (2012) Application of firefly algorithm and its parameter setting for job shop scheduling. J Ind Technol 8(1):49–58

KS SR, Murugan S (2017) Memory based hybrid dragonfly algorithm for numerical optimization problems. Expert Syst Appl 83:63–78

Kumar V, Chhabra JK, Kumar D (2015) Optimal choice of parameters for fireworks algorithm. Procedia Comput Sci 70:334–340

Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10(2):629–640

Luh GC, Lin CY (2009) Structural topology optimization using ant colony optimization algorithm. Appl Soft Comput 9(4):1343–1353

Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188(2):1567–1579

Maniezzo ACMDV (1992) Distributed optimization by ant colonies. In: Toward a practice of autonomous systems: proceedings of the First European conference on artificial life. MIT Press, p 134

Mavrovouniotis M, Li C, Yang S (2017) A survey of swarm intelligence for dynamic optimization: algorithms and applications. Swarm Evol Comput 33:1–17

Mezura-Montes E, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37(4):443–473

Mezura-Montes E, Coello Coello C, Velázquez-Reyes J, Muñoz-Dávila L (2007) Multiple trial vectors in differential evolution for engineering design. Eng Optim 39(5):567–589

Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249

Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
Nozari HA, Banadaki HD, Mokhtare M, Vahed SH (2012) Intelligent non-linear modelling of an industrial winding process using recurrent local linear neuro-fuzzy networks. J Zhejiang Univ Sci C 13(6):403–412
Omran MG, Engelbrecht AP, Salman A (2006) Particle swarm optimization for pattern recognition and image processing. In: Abraham A, Grosan C, Ramos V (eds) Swarm intelligence in data mining. Springer, Berlin, pp 125–151
Pan WT (2012) A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl-Based Syst 26:69–74
Pereira DG, Afonso A, Medeiros FM (2015) Overview of Friedman's test and post-hoc analysis. Commun Stat-Simul Comput 44(10):2636–2653
Rashaideh H, Sawaie A, Al-Betar MA, Abualigah LM, Al-Laham MM, Ra'ed M, Braik M (2018) A grey wolf optimizer for text document clustering. J Intell Syst 29:814–830
Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
Rodan A, Sheta AF, Faris H (2017) Bidirectional reservoir networks trained using SVM+ privileged information for manufacturing process modeling. Soft Comput 21(22):6811–6824
Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13(5):2592–2612
Sattari MRJ, Malakooti H, Jalooli A, Noor RM (2014) A dynamic vehicular traffic control using ant colony and traffic light optimization. In: Swiatek J, Grzech A, Swiatek P, Tomczak J (eds) Advances in systems science. Springer, Cham, pp 57–66
Sayed GI, Khoriba G, Haggag MH (2018) A novel chaotic salp swarm algorithm for global optimization and feature selection. Appl Intell 48(10):3462–3481. https://doi.org/10.1007/s10489-018-1158-6
Sheta A, Braik M, Al-Hiary H (2019) Modeling the Tennessee Eastman chemical process reactor using bio-inspired feedforward neural network (BI–FF–NN). Int J Adv Manuf Technol 103:1–22
Wang GG (2003) Adaptive response surface method using inherited Latin hypercube design points. J Mech Des 125(2):210–220
Wang GG (2018) Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput 10(2):151–164
Wang GG, Guo L, Gandomi AH, Hao GS, Wang H (2014) Chaotic krill herd algorithm. Inf Sci 274:17–34
Wang M, Chen H, Yang B, Zhao X, Hu L, Cai Z, Huang H, Tong C (2017) Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. Neurocomputing 267:69–84
Wang X, Qiu X (2013) Application of particle swarm optimization for enhanced cyclic steam stimulation in a offshore heavy oil reservoir. arXiv preprint arXiv:1306.4092
Xing Z, Jia H (2019) Multilevel color image segmentation based on GLCM and improved salp swarm algorithm. IEEE Access 7:37672–37690
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: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, pp 65–74
Yang XS (2010b) Nature-inspired metaheuristic algorithms. Luniver Press, London
Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: World congress on nature & biologically inspired computing, 2009. NaBIC 2009. IEEE, pp 210–214
Yang XS, Deb S, Loomes M, Karamanoglu M (2013) A framework for self-tuning optimization algorithm. Neural Comput Appl 23(7–8):2051–2057
Yi-jian L, Jian-ming Z, Shu-qing W (2005) Parameter estimation of cutting tool temperature nonlinear model using PSO algorithm. J Zhejiang Univ-Sci A 6(10):1026–1029
Zhang M, Luo W, Wang X (2008) Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 178(15):3043–3074
Zhou J, Liu Y, Yu Q (1998) GA algorithm for cutting experiment data drawing. J Southwest Pet Inst 29(3):62–63