Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 19(5): 5269–5292.
DOI: 10.3934/mbe.2022247
Received: 26 January 2022
Revised: 15 March 2022
Accepted: 17 March 2022
Published: 24 March 2022

Research article

Multi-strategy improved salp swarm algorithm and its application in reliability optimization

Dongning Chen[1,2,*], Jianchang Liu[1,2], Chengyu Yao[3], Ziwei Zhang[1,2] and Xinwei Du[1,2]

[1] Hebei Provincial Key Laboratory of Heavy Machinery Fluid Power Transmission and Control, Yanshan University, Qinhuangdao 066004, China
[2] Key Laboratory of Advanced Forging & Stamping Technology and Science (Yanshan University), Ministry of Education of China, Qinhuangdao 066004, China
[3] Key Laboratory of Industrial Computer Control Engineering of Hebei Province, Yanshan University, Qinhuangdao 066004, China

* **Correspondence:** Email: dnchen@ysu.edu.cn; Tel: +8613930358806.

Abstract: To improve the convergence speed and solution precision of the standard Salp Swarm Algorithm (SSA), a hybrid Salp Swarm Algorithm based on Dimension-by-dimension Centroid Opposition-based learning strategy, Random factor and Particle Swarm Optimization's social learning strategy (DCORSSA-PSO) is proposed. Firstly, a dimension-by-dimension centroid opposition-based learning strategy is added in the food source update stage of SSA to increase the population diversity and reduce the inter-dimensional interference. Secondly, in the followers' position update equation of SSA, constant 1 is replaced by a random number between 0 and 1 to increase the randomness of the search and the ability to jump out of local optima. Finally, the social learning strategy of PSO is also added to the followers' position update equation to accelerate the population convergence. The statistical results on ten classical benchmark functions by the Wilcoxon test and Friedman test show that compared with SSA and other well-known optimization algorithms, the proposed DCORSSA-PSO has significantly improved the precision of the solution and the convergence speed, as well as its robustness. The DCORSSA-PSO is applied to system reliability optimization design based on the T-S fault tree. The simulation results show that the failure probability of the designed system under the cost constraint is less than other algorithms, which illustrates that the application of DCORSSA-PSO can effectively improve the design level of reliability optimization.

Keywords: salp swarm algorithm; social learning; centroid opposition-based learning; system reliability optimization; T-S fault tree

1. Introduction

Optimization algorithm refers to the process of finding the best combination for a set of decision variables to solve a specific problem. For the complex optimization problems emerging in various fields such as engineering, economy, and medicine, it is not easy to find the optimal global solution by using traditional methods of mathematical optimization. However, the swarm intelligence optimization algorithm, which simulates the behavior of natural organisms, has successfully solved many complex optimization problems [1–4]. With the in-depth understanding of biological organisms, researchers have successively developed a series of swarm intelligence optimization algorithms, such as Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], Artificial Bee Colony (ABC) [7], Firefly Algorithm (FA) [8], Grey Wolf Optimizer (GWO) [9], Seagull Optimization Algorithm (SOA) [10], Slime Mould Algorithm (SMA) [11] and so on.

Mirjalili et al. [12] proposed a Salp Swarm Algorithm (SSA) in 2017, which is a new swarm intelligence optimization algorithm. Its optimization idea came from the population mechanism of the salp swarm chain foraging in the ocean. Once the SSA was proposed, it has attracted the extensive attention of many scholars because of its simple principle and easy implementation. Currently, this algorithm has been widely used in the fields of feature extraction [13,14], image segmentation [15,16], dispatch optimization [17], nodes localization [18], and so on.

The essence of the SSA is a random search optimization algorithm. It has the shortcomings of low accuracy in the later stage of iteration and is easy to get stuck at local optima. As a meta-heuristic algorithm, the searching behavior of SSA is divided into two main phases: exploration and exploitation phases. In exploration phase, it can efficiently discover the search space mostly by randomization, but it may face abrupt changes. In exploitation phase, it converges toward the most promising region. But, SSA often traps into local optima due to its stochastic nature and lack of balancing between exploration and exploitation. Thus, from this point, many studies have been presented to improve the performance of SSA and to overcome these defects.

Some improvement of single strategy has applied to enhance the performance of SSA by scholars. Sayed et al. [19] used a chaotic mapping sequence to take place the random parameter, which significantly improved the convergence rate and resulting precision of SSA. Abbassi et al. [20] proposed an Opposition-based Learning Modified Salp Swarm Algorithm (OLMSSA) for the accurate identification of circuit parameters. Singh et al. [21] updated the position of the salp swarm by sine cosine to enhance the exploration and exploitation capability. Syed et al. [22] proposed a strategy based on the weighted distance position update called the Weighted Salp Swarm Algorithm (WSSA) to enhance the performance and convergence rate of the SSA. Singh et al. [23] proposed a Hybrid SSA-PSO algorithm (HSSAPSO) by adding the speed optimization method of particle swarm optimization algorithm in the position update stage of salp swarm to avoid premature convergence of the optimal solution in the search space.

Moreover, multi-strategy improvement is adopted to enhance the SSA and has achieved good results with the development of research and application of SSA. Zhang et al. [24] used the Gaussian Barebone and stochastic fractal search mechanism to balance the global search ability and local search ability of the basic SSA. Liu et al. [25] proposed a new SSA-based method named MCSSA, in which the structure of SSA is rearranged using a chaos-assisted exploitation strategy and multi-population foundation to enhance its performance. Zhang et al. [26] proposed an ensemble composite mutation strategy to boost the exploitation and exploration trends of SSA, as well as a restart strategy to assist salps in getting away from local optimum. Zhao et al. [27] made an improvement of SSA called AGSSA, in which an adaptive control parameter is introduced into the position update stage of followers to boost the local exploitative ability of the population, and the elite gray wolf domination strategy is introduced in the last stage of the population position update to help the population find the global optimal solution faster. Zhang et al. [28] presented a chaotic SSA with differential evolution (CDESSA), and in the proposed framework, chaotic initialization is utilized to produce a better initial population aim at locating a better global optimal, and the differential evolution is used to build up the search capability of each agent. Xia et al. [29] proposed a QBSSA, in which an adaptive barebones strategy help to reach both accurate convergence speed and high solution quality and a quasi-oppositional-based learning make the population away from trapping into local optimal and expand the search space. Zhang et al. [30] proposed an enhanced SSA (ESSA), which improves the performance of SSA by embedding strategies such as orthogonal learning, quadratic interpolation, and generalized oppositional learning.

2. Motivation and innovation

Although the basic SSA enriches some characteristics like fast convergence speed and simple implementation, it may trap at sub-optimal solutions easily in some cases when handling the more complex optimization problems. Some improved algorithms of SSA have been provided by scholars mentioned as above, but each algorithm has its own merits and drawbacks. Hence, there is no guarantee which algorithm is best suited for a specific problem according to the "No free lunch" theorem [31]. In practical applications, there are special requirements on the accuracy or the convergence speed of the algorithm, so it is necessary to adopt more strategies.

In basic SSA, when the whole swarm of salps falls into a sub-optimal solution, the algorithm is trapped at that local solution and eventually stagnate at that suboptimal solution. So, we proposed a strategy of using dimension-by-dimension centroid opposition-based learning to make the slap population get more wide search space. Moreover, a random factor is used to increase the randomness of the population distribution and PSO's social learning strategy is added to speed up convergence.

The main contributions of this paper are as follows:

1) An improved algorithm which combines dimension-by-dimension centroid opposition-based learning, random factor, and PSO's social learning strategy (DCORSSA-PSO) is proposed.

2) The performance of proposed DCORSSA-PSO is verified by comparing it with several well-known algorithms in benchmark functions.

3) The proposed DCORSSA-PSO is used to the design of system reliability optimization based on T-S fault tree and has achieved good result.

The remainder section of this article is structured as follows. Section 3 introduces the basic principles of SSA. Section 4 introduces the mathematical principles of dimension-by-dimension centroid opposition-based learning, the addition of random factor and PSO' social learning, and proposes DCOSSA, DCORSSA and DCORSSA-PSO. Section 5 contains simulation experiment and result analysis. In Section 6, the efficacy of the proposed DCORSSA-PSO is assessed on engineering design of system reliability optimization. Finally, conclusions and future works are summarized in Section 7.

3. Basic principles of salp swarm algorithm

The salp swarm algorithm was proposed by Mirjalili et al. in 2017 [12], which is a heuristic swarm intelligent optimization algorithm that simulates the navigating and foraging behavior of salps. The salp chain consists of two types of salps: leader and follower. The leader is the salp at the head of the salp chain, and the other salps are considered followers. To enhance the population diversity of the algorithm and enhance the ability to jump out of the local optima, half the salps are selected as the leaders.

The position update equation of the leader is as follows [32]:

$$x_j^i = \begin{cases} F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right) & 0 \le c_3 < 0.5 \\ F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right) & 0.5 \le c_3 \le 1 \end{cases} \tag{1}$$

where x_j^i is the position of the ith leader in the jth dimension; F_j is the position of food source in the jth dimension; ub_j and lb_j indicate the upper and lower bound of the jth dimension, respectively; c_2 and c_3 are random numbers in the range [0,1], which decide respectively the moving step and the moving direction (positive or negative) of the jth dimension. c_1 is the convergence factor, which is used to balance the exploration and exploitation ability of the algorithm in the iterative process. c_1 is defined as follows:

$$c_1 = 2e^{-(4t/T)^2} \tag{2}$$

where t is the current number of iterations and T is the maximum number of iterations.

The position update equation of followers is as follows:

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}) \tag{3}$$

where $i \ge 2$ and x_j^i is the position of the ith follower in the jth dimension. Eq (3) shows that the ith follower in the jth dimension is updated according to the center of the previous generation of the ith follower and the i-1th follower in the jth dimension.

4. An improved salp swarm algorithm

Although the SSA is experienced to reach good accuracy compared with recent meta-heuristics, it may still face the shortcomings of getting trapped in local optima and is not suitable for highly complex optimization functions. To extend the search capability of SSA, a new hybrid salp swarm algorithm based on dimension-by-dimension centroid opposition-based learning strategy, random factor, and PSO's social learning strategy (DCORSSA-PSO) is proposed to solve engineering problems.

4.1. DCOSSA

4.1.1. Opposition-based learning

Opposition-based learning is a novel learning strategy proposed by Tizhoosh in 2005 [33]. The principle is that the current optimal solution and the opposition-based learning solution are searched for simultaneously during the population iteration and the better one in these two solutions is retained to the next generation according to the fitness value. This searching method improves the population diversity and enhances the ability of the algorithm to jump out of the local solutions.

Definition 1: Let $x \in R$ be a real number and $x \in [lb, ub]$. The opposite number $\tilde{x}$ is defined as follows:

$$\tilde{x} = lb + ub - x \tag{4}$$

Analogously, the opposite number in a multidimensional case is also defined.

Definition 2: Let $X = (x_1, x_2, \cdots, x_D)$ be a point in a D-dimensional coordinate system with $x_1, ..., x_D \in R$, and $x_j \in [lb_j, ub_j]$. The opposite point $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_D)$ is defined as:

$$\tilde{x}_j = lb_j + ub_j - x_j, \quad 1 \le j \le D \tag{5}$$

4.1.2. Centroid opposition-based learning

When the opposition-based learning method calculates the opposite point, two boundaries (min and max) are taken from two extreme points in the population for every dimension. The remaining points of the population are not considered, so this represents a weakness in terms of convergence speed. A Centroid Opposition-Based Computing (COBC) was proposed by Rahnamayan [34], which takes the entire population to generate the opposite points and hence improves the convergence speed and solution accuracy.

Let $(X_1, ..., X_n)$ be n points in D-dimensional search space with each point in that space carrying a unit mass. Then the centroid of the body can be defined as follows:

$$M = \frac{X_1 + X_2 + \cdots + X_n}{n} \tag{6}$$

where then we have

$$M_j = \frac{\sum_{i=1}^{n} x_{i,j}}{n} \tag{7}$$

where $x_{i,j}$ is the jth dimension of ith point, M_j is the jth dimension centroid of all n points.

The opposite-point $\tilde{X}_i$ of the point X_i is calculated as follows:

$$\begin{cases} \tilde{X}_i = 2 \times M - X_i \\ \tilde{x}_{i,j} = 2 \times M_j - x_{i,j}, \quad 1 \le j \le D \end{cases} \tag{8}$$

4.1.3. Dimension-by-dimension centroid opposition-based learning

At present, most algorithms adopt the method of variation for all dimension information of the population and select or eliminate evolution by comparing the fitness values of different individuals as a whole for all dimension, however, it is difficult to ensure that each selected dimensional information of the evolutionary individual is better than that of the eliminated individual. Therefore, there is often inter-dimensional interference in the calculation of the fitness value, which masks the information of evolution dimension and reduces the quality of solution and the convergence speed.

The dimension-by-dimension update evaluation strategy refers to evaluating the fitness value separately for each dimension, which can reduce the inter-dimensional interference between each dimension of the individual and avoid the problem of low variation efficiency. This operation for population individuals can more accurately evaluate the fitness value in each iteration. However, for the high-dimensional test functions, it will greatly increase the time complexity of the algorithm. Through the test, it is found that better results can still be achieved by updating the dimensional information of the optimal individual instead of each individual.

4.1.4. DCOSSA

This paper combines the dimension-by-dimension update strategy with the centroid opposition-based learning strategy, and proposes a Dimension-by-dimension Centroid Opposition-based learning Salp Swarm Algorithm (DCOSSA). The basic step is: Firstly, calculate the opposite point of the centroid of the current population through Eqs (4)–(8). Then, replace the information at the first dimension of the food source with the first dimension of the opposite point of the center of gravity. If the fitness value of the new food source is better than the fitness value of the original food source, the opposite solution information of the population center of gravity of this dimension is retained, otherwise, the update result of this dimension is discarded. At last, update the next food source information in dimension order until all dimensions are updated.

The pseudo-code of the strategy of dimension-by-dimension centroid opposition-based learning is as follows:

```
Calculate the center position $M$ of the iterative population according to Eq (6)
for $j$ = 1: $Dim$
    Calculate opposite solution of the food source position $\tilde{F}_j$ according to Eq (8)
    If $f(\tilde{F}_j)$ is better than $f(F_j)$ then
        $F_j = \tilde{F}_j$
    end if
end for
```

4.2. DCORSSA

According to the position update Eq (3) of followers in the salp swarm algorithm, the position of follower takes the center of gravity of the corresponding position of the salp in the previous generation and the position adjacent to the previous salp. The head follower uses the information of the leader slap besides its information of previous generation, so the whole population of followers are affected gradually by the leader slap. It can be seen from Eq (1) that the leader's position is updated near the food source. Therefore, when the food source does not fall into the local solutions, this end-to-end search mechanism can enable the followers to fully carry out local exploitation, but the gradual transmission of the global optimal information is not conducive to the rapid convergence of the algorithm. Moreover, when the food source falls into the local solutions, the followers will fall into the dilemma of invalid region search, that is, the lack of population diversity, so the algorithm is easy to fall into the local extremes.

The above searching mechanism is fixed and lack of dynamic adjustment. Therefore, this paper proposes a DCORSSA algorithm on the basis of DCOSSA, that is, a random factor is added to the update equation of its followers to enhance the update randomness, so that population can get more chance to jump out of the local optima.

A random factor c_4 between 0 and 1 replaces the constant 1 of Eq (3). So, the new position update equation of followers is as follows:

$$x_j^i = \frac{c_4}{2}(x_j^i + x_j^{i-1}) \tag{9}$$

4.3. DCORSSA-PSO

As mentioned above in Section 4.2, the one-by-one transfer mechanism of salp chain makes the convergence speed is slow. So, in this section, we introduce the social learning strategy of PSO on the basis of DCORSSA. The Particle Swarm Optimization (PSO) is a very practical swam optimization algorithm proposed by Kennedy and Eberhart [5]. Particles search for optimization through information sharing mechanism. That is, particles obtain their own historical experience (individual optimal p^i) and group experience (global optimal p^g) through information exchange between individuals to achieve the purpose of optimization. The update formula of velocity and position are as follows:

$$v_j^i(t+1) = \omega v_j^i(t) + c_1 r_1(t)[p_j^i(t) - x_j^i(t)] + c_2 r_2(t)[p_j^g(t) - x_j^i(t)] \tag{10}$$

$$x_j^i(t+1) = x_j^i(t) + v_j^i(t+1) \tag{11}$$

where v_j^i is the velocity of the ith particle in the jth dimension, x_j^i is the position of the ith particle in the jth dimension, t refers to the iteration number, w is inertia weight that aims at determining the effect of previous velocities on current velocity, c_1 represents the individual learning factor and c_2 represents the social learning factor, r_1 and r_2 are random variables used to increase the randomness of particle flight, whose values are normally distributed in [0, 1], p_j^i and p_j^g indicate the elements of individual optimal location and global optimal location in the jth dimension, respectively.

Equation (10) includes three parts: the first part is the "inertia" part, which is the motion inertia of particles, reflecting the tendency of particles to maintain their previous velocity; the second part is the "individual learning", which reflects the trend of particles moving to their previous best position in history; the third part is the "social learning", which reflects the trend of particles moving to the best position in the previous history of the population.

Finally, based on DCORSSA, this paper further proposes a DCORSSA-PSO algorithm. That is, the social learning strategy of PSO which is the third part of Eq (10) is introduced to the position update equation of the followers of DCORSSA. On the basis of increasing the random distribution of its own position, the improved algorithm makes full use of the global information and strengthens the tendency of individuals to move to the food source.

The position update equation of followers of proposed DCORSSA-PSO is as follows:

$$x_j^i = x_j^i + c_5 \times \text{rand}(0,1) \times \left(F_j - x_j^i\right) \tag{12}$$

where x_j^i is the position of the ith follower in the jth dimension updated by Eq (9), F_j is the food source position of the jth dimension, c_4 is random number between 0 and 1, c_5 is the social learning factor, taken $c_5 = 1.49$.

The flow chart of the algorithm DCORSSA-PSO is shown in Figure 1.

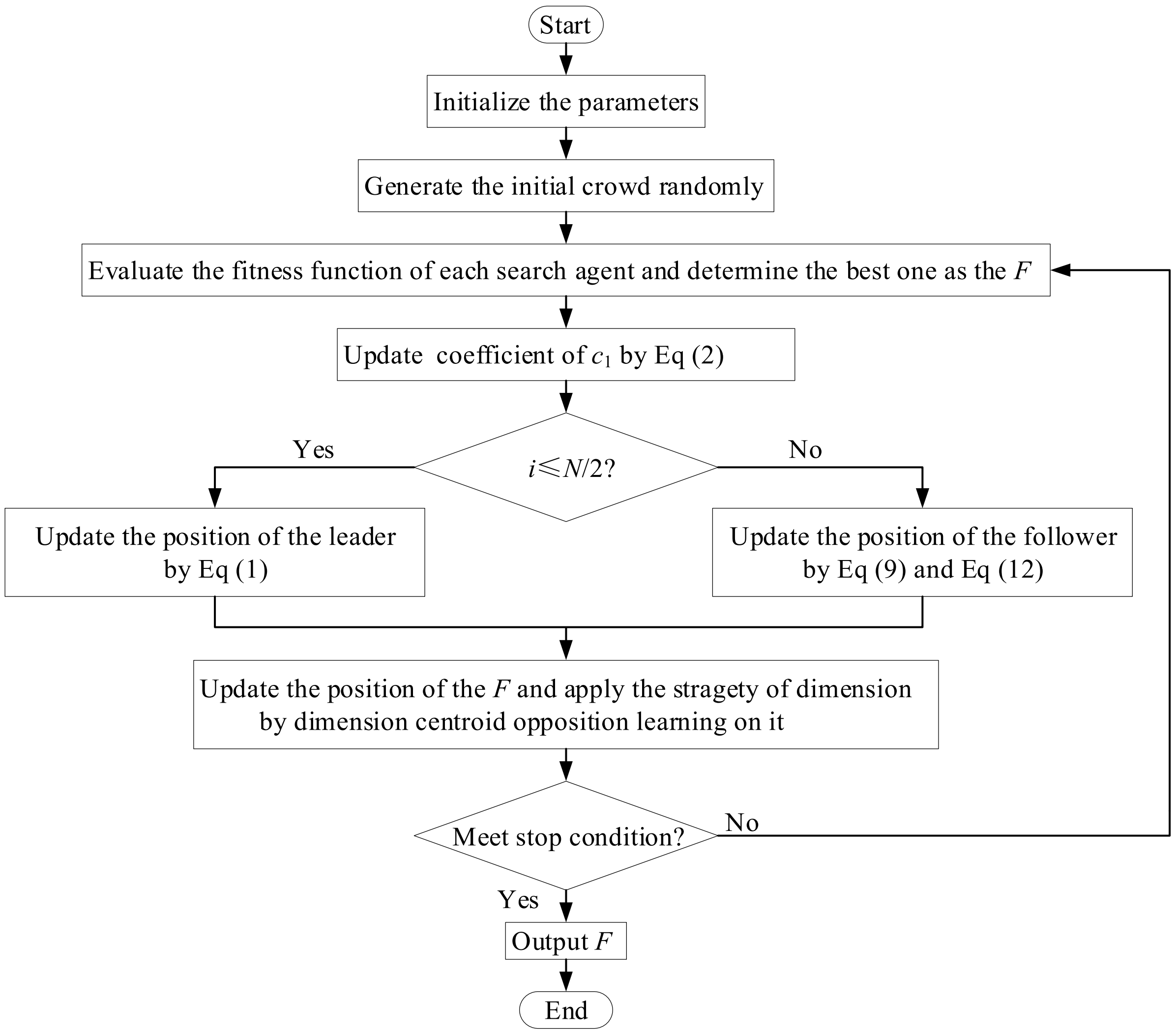

Start
Initialize the parameters
Generate the initial crowd randomly
Evaluate the fitness function of each search agent and determine the best one as the F
Update coefficient of c1 by Eq (2)
Yes
i≤N/2?
No
Update the position of the leader by Eq (1)
Update the position of the follower by Eq (9) and Eq (12)
Update the position of the F and apply the stragety of dimension by dimension centroid opposition learning on it
Meet stop condition?
No
Yes
Output F
End

Figure 1. Flow chart of DCORSSA-PSO.

The pseudo-code of DCORSSA-PSO is as follows:

Set the initial parameters of the algorithm: population numbers N, population dimensions Dim, population iteration times T, search upper bound ub and search lower bound lb;

Initialize the population randomly, calculate the fitness value of each individual, and take the position with the optimal fitness value as the food source position F;

for $t = 1$: T

```
    for $i$ = 1: $N$
       if $i$ <= $N$/2
          Update the position of leader by Eq (1)
       else
          Update the position of followers by Eqs (9) and (12)
       end if
    end for
    Update the food source position $F$ and its fitness value
    Calculate the center position $M$ of the iterative population according to Eq (6)
       for $j$ = 1: $Dim$
          Calculate opposite solution of the food source position $\tilde{F}_j$ according to Eq (8)
          If $f(\tilde{F}_j)$ is better than $f(F_j)$ then
             $F_j = \tilde{F}_j$
          end if
       end for
end for
```

4.4. Computational complexity of DCORSSA-PSO

According to the literature [12], the computational complexity of the SSA algorithm is $O(t(d*n + Cof*n))$ where t shows the number of iterations, d is the number of variables (dimension), n is the number of solutions, and Cof indicates the cost of the objective function. In the DCORSSA-PSO, the fitness function is recalculated for each dimension of the food source with the dimension-by-dimension centroid opposition-based learning, so the amount of operation of $O(Cof * d)$ is increased; simultaneously, to increase the search vitality of the algorithm and improve the search speed of the algorithm, random factor and PSO's social learning strategy are introduced. Still, the number of code execution is not increased. So the computational complexity of DCORSSA-PSO algorithm is $O(t(d*n + Cof*n + Cof*d))$. It can be seen that the computational complexity of the DCORSSA-PSO algorithm is higher than that of standard SSA, and it increases with the increase of population dimension.

5. Simulation experiment and result analysis

5.1. Testing environment

Test environment: the hard disk running environment is CPUi5-7200U, the memory is 12GB, the software running environment is windows10 system, and the running software is MATLAB 2019b. These parameters of different algorithms are set the same: population sizes $N = 30$, population dimensions $Dim = 30$, and the maximum number of iterations $T = 500$. Respective parameter settings for involved algorithms are shown in Table 1.

Table 1. Parameter settings for involved algorithms.

Algorithm	Parameters
SSA	$c_1 \in [2 \times e^{-16}, 2 \times e]$; $c_2 \in [0,1]$; $c_3 \in [0,1]$;
DCOSSA	$c_1 \in [2 \times e^{-16}, 2 \times e]$; $c_2 \in [0,1]$; $c_3 \in [0,1]$;
DCORSSA	$c_1 \in [2 \times e^{-16}, 2 \times e]$; $c_2 \in [0,1]$; $c_3 \in [0,1]$; $c_4 \in [0,1]$;
DCORSSA-PSO	$c_1 \in [2 \times e^{-16}, 2 \times e]$; $c_2 \in [0,1]$; $c_3 \in [0,1]$; $c_4 \in [0,1]$; $c_5 = 1.49$;
PSO	$w \in [0.4, 0.9]$; $c_1 = c_2 = 1.49$;
GWO	$r_1 \in [0,1]$; $r_2 \in [0,1]$

5.2. Test function

Ten classical benchmark functions, as shown in Table 2, are used to evaluate the performance of DCORSSA-PSO. SSA, PSO, DCOSSA and GWO are the algorithms for comparison. In the ten functions, f_1~f_4 are unimodal test functions that can test the optimization accuracy of the algorithms, f_5~f_8 are multimodal test functions that can test the global optimization ability and convergence speed of the algorithms, f_9~f_{10} are ill-conditioned test functions that can test the exploration and exploitation capabilities of the algorithms.

Table 2. Benchmark functions used in the study.

Benchmark function	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]	0
$f_3(x) = \max_{i=1}^{n} \{\lvert x_i \rvert\}$	[-100, 100]	0
$f_4(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	[-100, 100]	0
$f_5(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12, 5.12]	0
$f_6(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$	[-600, 600]	0
$f_7(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	[-32, 32]	0
$f_8(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-100, 100]	0
$f_9(x) = \sum_{i=1}^{n} i x_i^4 + rand$	[-1.28, 1.28]	0
$f_{10}(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	[-30, 30]	0

Table 3. Test results of DCORSSA-PSO and compared algorithms.

Functions	Measure	SSA	DCOSSA	DCORSSA	DCORSSA-PSO	PSO	GWO
f_1	Mean	2.40×10^{-7}	2.28×10^{-9}	2.70×10^{-27}	5.80×10^{-44}*	2.08×10^{2}	1.42×10^{-27}
	Best	3.70×10^{-8}	9.29×10^{-12}	6.45×10^{-32}	1.42×10^{-46}*	5.82×10^{1}	3.07×10^{-29}
	Std	4.09×10^{-7}	5.16×10^{-9}	5.86×10^{-27}	1.17×10^{-43}*	1.08×10^{2}	1.82×10^{-27}
	Time/s	1.01×10^{-1}*	1.99×10^{-1}	2.01×10^{-1}	2.11×10^{-1}	2.71×10^{-1}	1.70×10^{-1}
f_2	Mean	1.96	1.36×10^{-5}	1.00×10^{-14}	7.80×10^{-23}*	6.30	7.96×10^{-17}
	Best	1.71×10^{-1}	1.03×10^{-6}	6.24×10^{-16}	2.12×10^{-24}*	3.57	4.54×10^{-18}
	Std	1.46	3.95×10^{-5}	1.18×10^{-14}	9.73×10^{-23}*	1.75	4.69×10^{-17}
	Time/s	9.37×10^{-2}*	1.83×10^{-1}	1.80×10^{-1}	1.93×10^{-1}	2.41×10^{-1}	1.47×10^{-1}
f_3	Mean	1.20×10^{1}	1.08	9.05×10^{-15}	7.16×10^{-23}*	1.24×10^{1}	8.01×10^{-7}
	Best	5.19	1.36×10^{-1}	6.79×10^{-17}	1.98×10^{-24}*	4.76	1.16×10^{-7}
	Std	3.35	6.96×10^{-1}	1.29×10^{-14}	8.06×10^{-23}*	3.19	7.47×10^{-7}
	Time/s	9.07×10^{-2}*	1.75×10^{-1}	1.72×10^{-1}	1.85×10^{-1}	2.36×10^{-1}	1.41×10^{-1}
f_4	Mean	1.08×10^{-7}	9.70×10^{-10}*	3.39×10^{-7}	6.50×10^{-8}	1.67×10^{2}	7.97×10^{-1}
	Best	2.54×10^{-8}	9.62×10^{-12}*	2.56×10^{-8}	2.41×10^{-8}	5.07×10^{1}	2.57×10^{-1}
	Std	8.12×10^{-8}	2.23×10^{-9}*	8.51×10^{-7}	2.66×10^{-8}	8.92×10^{1}	2.57×10^{-1}
	Time/s	9.29×10^{-2}*	1.79×10^{-1}	1.76×10^{-1}	1.91×10^{-1}	2.42×10^{-1}	1.44×10^{-1}
f_5	Mean	6.16×10^{1}	4.64×10^{-1}	0*	0*	1.31×10^{2}	3.42
	Best	2.79×10^{1}	2.80×10^{-12}	0*	0*	8.73×10^{1}	0*
	Std	1.80×10^{1}	5.68×10^{-1}	0*	0*	2.41×10^{1}	4.36
	Time/s	1.05×10^{-1}*	1.98×10^{-1}	1.92×10^{-1}	2.00×10^{-1}	2.59×10^{-1}	1.52×10^{-1}
f_6	Mean	9.54×10^{-1}	1.00×10^{-1}	0*	0*	2.65	4.12×10^{-13}
	Best	8.49×10^{-1}	4.44×10^{-2}	0*	0*	1.57	5.00×10^{-15}
	Std	4.30×10^{-2}	3.49×10^{-2}	0*	0*	1.04	4.65×10^{-13}
	Time/s	1.22×10^{-1}*	2.42×10^{-1}	2.36×10^{-1}	2.46×10^{-1}	2.77×10^{-1}	1.70×10^{-1}
f_7	Mean	1.96	1.04	2.96×10^{-15}	2.25×10^{-23}*	2.75	1.83×10^{-1}
	Best	1.00	7.00×10^{-1}	1.22×10^{-16}	1.86×10^{-25}*	1.80	9.99×10^{-2}
	Std	4.18×10^{-1}	2.40×10^{-1}	2.99×10^{-15}	2.90×10^{-23}*	4.88×10^{-1}	3.79×10^{-2}
	Time/s	9.56×10^{-2}*	1.94×10^{-1}	1.91×10^{-1}	2.00×10^{-1}	2.46×10^{-1}	1.51×10^{-1}
f_8	Mean	2.46	7.58×10^{-6}	8.23×10^{-15}	8.88×10^{-16}*	5.84	9.80×10^{-14}
	Best	9.31×10^{-1}	7.50×10^{-7}	8.88×10^{-16}*	8.88×10^{-16}*	4.45	7.55×10^{-14}
	Std	7.42×10^{-1}	1.01×10^{-5}	1.23×10^{-14}	0*	8.80×10^{-1}	1.49×10^{-14}
	Time/s	1.08×10^{-1}*	2.04×10^{-1}	1.98×10^{-1}	2.10×10^{-1}	2.62×10^{-1}	1.52×10^{-1}
f_9	Mean	1.51	5.48×10^{-2}	6.72×10^{-5}*	7.24×10^{-4}	1.22	1.72×10^{-2}
	Best	5.85×10^{-1}	1.73×10^{-2}	1.32×10^{-7}*	7.12×10^{-6}	3.09×10^{-2}	8.50×10^{-4}
	Std	6.72×10^{-1}	1.91×10^{-2}	6.83×10^{-5}*	1.55×10^{-3}	1.23	1.55×10^{-2}
	Time/s	1.59×10^{-1}*	3.16×10^{-1}	3.18×10^{-1}	3.27×10^{-1}	3.06×10^{-1}	2.09×10^{-1}
f_{10}	Mean	2.77×10^{2}	5.65×10^{1}	2.79×10^{1}	2.64×10^{1}*	5.39×10^{3}	2.69×10^{1}
	Best	2.49×10^{1}*	3.96×10^{-2}	2.74×10^{1}	2.62×10^{1}	5.89×10^{2}	2.57×10^{1}
	Std	4.33×10^{2}	4.31×10^{1}	2.05×10^{-1}	1.33×10^{-1}*	5.25×10^{3}	7.83×10^{-1}
	Time/s	1.24×10^{-1}*	2.47×10^{-1}	2.47×10^{-1}	2.60×10^{-1}	2.81×10^{-1}	1.78×10^{-1}

Note: the mark "*" at the top right of the data indicates the best result obtained by all algorithms.

5.3. Experimental results and analysis

To objectively test the optimization performance of the algorithm of DCORSSA-PSO, the same initial population is selected for all algorithms, and the average fitness value, optimal fitness value, standard deviation of fitness value, and the average running time of each algorithm for 30 times independently are counted to evaluate the algorithm comprehensively. The comparison results of each algorithm are shown in Table 3.

From the experimental results of 30 independent runs in Table 3, it can be seen that the optimization performance of each algorithm in the standard test functions is different. The optimal value and average value can measure the accuracy of the optimization algorithm. In multimodal functions f_5 and f_6, DCORSSA-PSO and DCORSSA can search the theory optimal value 0, showing excellent optimization ability. In the test functions f_1~f_3, f_7, the DCORSSA-PSO algorithm is superior to other algorithms in terms of both the average value and the optimal value. In the test functions f_8 and f_{10}, DCORSSA-PSO is superior to other algorithms in terms of the average value. In the unimodal test function f_1, DCORSSA-PSO is more than 10 orders of magnitude higher than other algorithms in the accuracy of the optimal value. Compared with SSA, the convergence accuracy of DCORSSA and DCORSSA-PSO is also greatly improved, which indicates that adding different optimization strategies is very helpful to improve the optimization of SSA. At the same time, in the ill-conditioned function f_9, the optimization accuracy of DCORSSA-PSO reaches 7.12×10^{-6}; although its accuracy is improved compared with SSA, it is worse than DCORSSA. In the ill-conditioned function f_{10}, although the DCORSSA-PSO algorithm improves the mean optimization accuracy compared with the SSA algorithm, the optimal value search is still insufficient compared with the SSA algorithm. In unimodal function f_4, DCORSSA-PSO inferiors to DCOSSA in terms of average value and optimal value. These cases which DCORSSA-PSO does not get the best performance indicate that the DCORSSA-PSO algorithm is still insufficient in search of some functions.

The standard deviation can measure the optimization stability of the optimization algorithm. Except for f_4 and f_9, the standard deviation of the DCORSSA-PSO algorithm calculated 30 times independently is always less than that of other algorithms, which shows that the improved DCORSSA-PSO algorithm can ensure the optimization stability of the algorithm when dealing with unimodal, multimodal, even ill-conditioned functions.

In terms of average running time, the SSA algorithm has a shorter running time than PSO algorithm and GWO algorithm, which shows that the improved DCOSSA, DCORSSA and DCORSSA-PSO have inherent advantages in operation speed. The average running time of DCORSSA-PSO algorithm is slightly longer than that of the DCOSSA, which does not cause a significant increase in running time, indicating that the addition of random factor and PSO's social learning strategy have little impact on the time complexity of the algorithm. The average running time of DCORSSA-PSO algorithm and DCORSSA is longer than that of the SSA algorithm, mainly due to the addition of dimension-by-dimension centroid opposition-based learning strategy.

The Wilcoxon signed-rank test [35] with a significance level of 0.05 was used to judge the statistical difference between the improved algorithm DCORSSA-PSO and the comparative algorithms such as SSA. The statistical results are shown in Table 4, in which: "+" indicates that the test result of DCORSSA-PSO is superior to the corresponding comparison algorithm. "=" indicates that the performance of the DCORSSA-PSO test result is similar to the corresponding comparison algorithm, and there is no statistically significant difference. "-" indicates that the DCORSSA-PSO test result is inferior to that of the corresponding comparison algorithm.

Table 4. Wilcoxon signed-rank test of DCORSSA-PSO and other algorithms.

Comparison group	+/=/-	Comparison group	+/=/-
DCORSSA-PSO VS SSA	9/1/0	DCORSSA-PSO VS PSO	10/0/0
DCORSSA-PSO VS DCOSSA	9/0/1	DCORSSA-PSO VS GWO	10/0/0
DCORSSA-PSO VS DCORSSA	7/2/1	---	

According to the Wilcoxon signed-rank test results described in Table 4, it can be learned that DCORSSA-PSO wins in 45(= 9 + 9 + 7 + 10 + 10) cases, loses in 2 cases and shows a tie in the other cases in the total 50 (=5*10) cases. In general, the DCORSSA-PSO algorithm is better than other algorithms such as SSA algorithm in most functions, which proves the effectiveness of the proposed improved method.

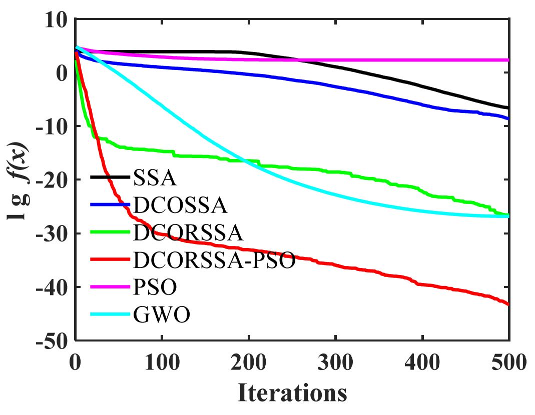
In addition, in order to further evaluate the statistical comparison of the optimization performance of each algorithm, Friedman test [36] is used to study the difference between each algorithm as is shown in Table 5. The average ranking value (ARV) represents the average ranking value of the Friedman test of an algorithm that runs 30 times of all test functions independently. The smaller the ARV, the higher the optimization performance of the algorithm.

Table 5. Friedman test of DCORSSA-PSO and other algorithms.

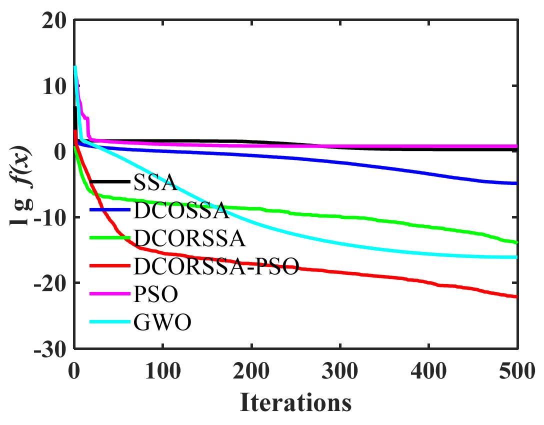
Algorithm	SSA	DCOSSA	DCORSSA	DCORSSA-PSO	PSO	GWO
ARV	4.8533	3.5700	2.2567	1.4300	5.8700	3.0200
rank	5	4	2	1	6	3

From Table 5, we can clearly see the statistical results of the Friedman test. The ARV of DCORSSA-PSO integrating the three learning strategies is 1.4300, and the rank is No.1, which indicates that DCORSSA-PSO is significantly better than other comparison algorithms in solving these test functions. In addition, the rank of DCORSSA and DCOSSA combining the other strategies are No.2 and No.4 respectively, indicating that the above-mentioned optimization strategies are of great help in improving the optimization accuracy of the SSA algorithm.

Figure 2 shows the average convergence curve of each algorithm in 10 standard test functions. To better observe the optimization effect of the algorithm, the logarithm based on 10 is taken for the optimization fitness values of f_1~f_{10}.

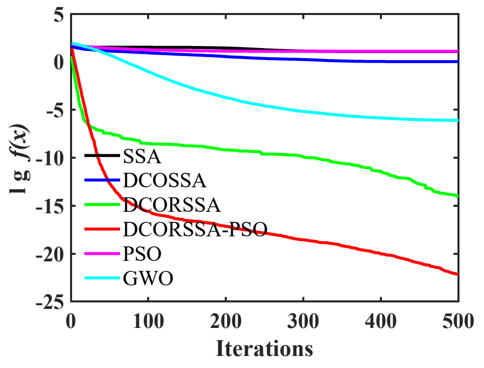


(a) Performance comparison on the f_1 function.

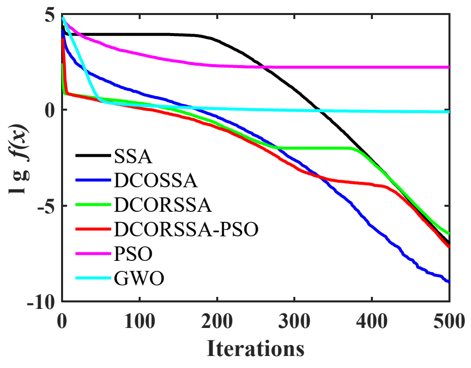


(b) Performance comparison on the f_2 function.

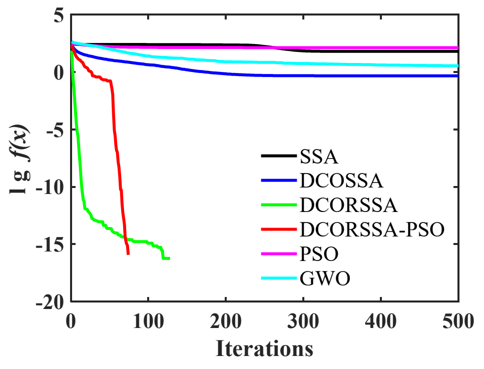
continued on next page



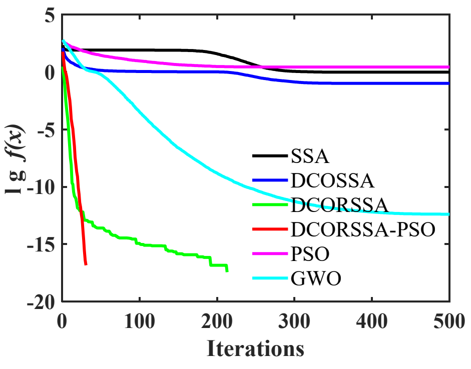
(c) Performance comparison on the f_3 function.



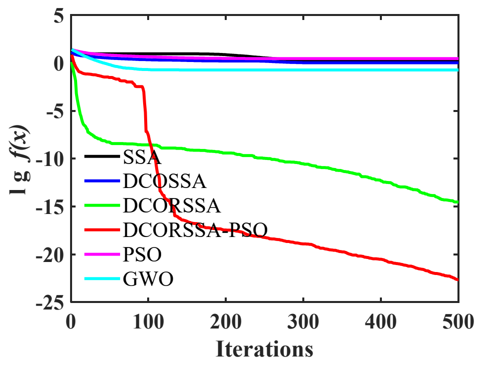
(d) Performance comparison on the f_4 function.



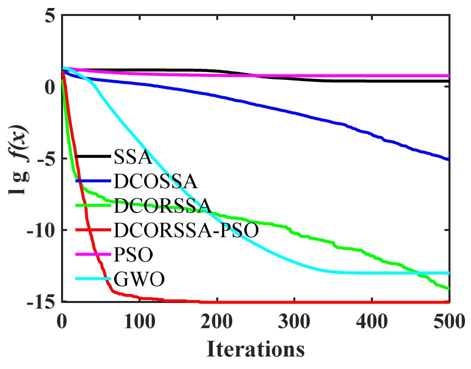
(e) Performance comparison on the f_5 function.



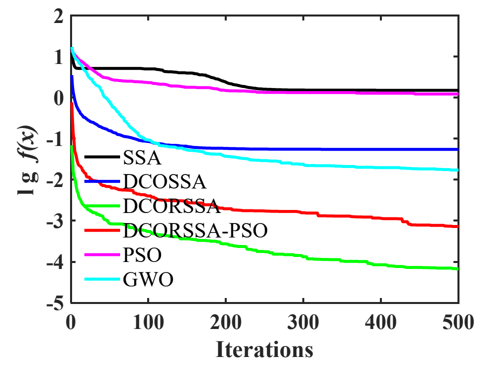
(f) Performance comparison on the f_6 function.



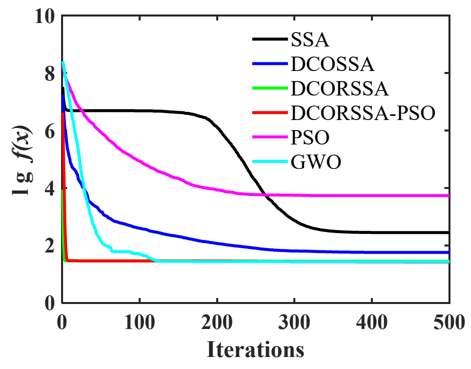
(g) Performance comparison on the f_7 function.



(h) Performance comparison on the f_8 function.



(i) Performance comparison on the f_9 function.



(j) Performance comparison on the f_{10} function.

Figure 2. Average convergence curve of the standard test functions.

In Figure 2, it can be seen that the optimization speed and accuracy of DCOSSA added with the dimension-by-dimension centroid opposition-based learning strategy are greatly improved compared with SSA, which shows that the dimension-by-dimension centroid opposition-based learning strategy is of great benefit to improve the population diversity and the ability to jump out of the local solutions. Compared with DCOSSA and the other three algorithms, DCORSSA-PSO which adds a random factor and integrates the social learning strategy of PSO, declines rapidly in the middle of the iteration, and its optimization speed is significantly ahead. Especially in the middle and early iterations of the function, the DCORSSA-PSO algorithm can almost quickly search for the optimal value, and continues to show high search activity in the later iterations. Even in multimodal functions f_5 and f_6, the curves are interrupted because the DCORSSA-PSO algorithm searches the theoretical optimal value 0 (the independent variable of lg cannot be 0).

All the above show that the DCORSSA-PSO algorithm is effective in dealing with unimodal, multimodal, and ill-conditioned test functions, and it has better optimization accuracy and speed, which is very helpful to solve the problems to be optimized in engineering practice.

6. System reliability optimization model

Nowadays, more and more engineering problems are adopting optimization methods to get optimal performance [37], while system reliability optimization is one of the most useful engineering fields. System reliability optimization refers to finding an optimal design under certain resource constraints to obtain the highest reliability of the system or minimizing the investment while meeting specific reliability index requirements, thus obtaining the maximum economic benefits. At present, practice shows that the optimal redundancy allocation design is one of the most usually used methods to reduce system failure probability and improve system reliability. Redundancy design means that when a part of the system fails, the redundant part is activated through the monitoring and switching mechanism to complete the same function instead of the failed part, to reduce the failure probability of the system.

Many scholars have used intelligent optimization algorithms to solve reliability optimization problems. In literature [38], an enhanced nest cuckoo optimization algorithm was used to study the system reliability redundancy allocation with a cold-standby strategy. Literature [39] carried out reliability optimization of a fuzzy multi-objective system based on genetic algorithm and cluster analysis. Literature [40] proposed a new particle swarm optimization algorithm based on fuzzy adaptive inertia weight to solve the reliability redundancy allocation problem.

6.1. T-S fault tree construction

Fault tree analysis is one of the commonly used reliability analysis methods, which is oriented by system failure and unit failure. A fault tree is composed of events and gates. It is named fault tree because its fault logic relationship is graphically represented like a tree with the top event as root, event logic causality represented by the gate as a branch, and bottom event as a leaf. T-S model [41] was proposed by Takagi and Sugeno in 1985. Through if-then rules, a series of local linear subsystems and membership functions were used to accurately describe nonlinear systems. Song et al. [42] constructed T-S gates to describe event relations based on the T-S model, proposing the T-S fault tree analysis method. Yao et al. [43] proposed a new reliability optimization method based on the T-S fault tree and EPSO (Extended PSO).

Hypothetically, a mechanical system consists of two subsystems, each of which can improve the system reliability by adding a redundant design. In this paper, the T-S fault tree analysis method is used to construct the reliability allocation optimization model of the system, and the DCORSSA-PSO algorithm is used to optimize its reliability allocation. The T-S fault tree of the mechanical system is shown in Figure 3.

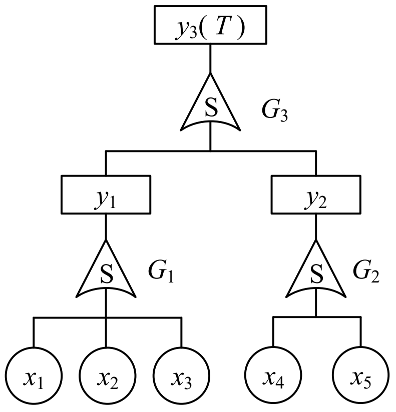


Figure 3. T-S fault tree of a mechanical system.

In Figure 3, x_1~x_5 are the bottom events, y_1~y_2 are the intermediate events, and y_3 is the top event. G_1~G_3 are T-S gates. Fuzzy numbers 0, 0.5 and 1 represent the three states of normal, semi failure, and complete failure of each part, respectively. The fault states of each part are independent of each other. According to expert experience and historical data, the rule tables of the T-S gate are defined as shown in Tables 6–8.

Table 6. Rule table of T-S gate 1.

rules	x_1	x_2	x_3	y_1			rules	x_1	x_2	x_3	y_1		
				0	0.5	1					0	0.5	1
1	0	0	0	1	0	0	15	0.5	0.5	1	0	0	1
2	0	0	0.5	0.2	0.5	0.3	16	0.5	1	0	0	0	1
3	0	0	1	0	0	1	17	0.5	1	0.5	0	0	1
4	0	0.5	0	0.3	0.5	0.2	18	0.5	1	1	0	0	1
5	0	0.5	0.5	0.2	0.3	0.5	19	1	0	0	0	0	1
6	0	0.5	1	0	0	1	20	1	0	0.5	0	0	1
7	0	1	0	0	0	1	21	1	0	1	0	0	1
8	0	1	0.5	0	0	1	22	1	0.5	0	0	0	1
9	0	1	1	0	0	1	23	1	0.5	0.5	0	0	1
10	0.5	0	0	0.2	0.5	0.3	24	1	0.5	1	0	0	1
11	0.5	0	0.5	0.1	0.4	0.5	25	1	1	0	0	0	1
12	0.5	0	1	0	0	1	26	1	1	0.5	0	0	1
13	0.5	0.5	0	0.1	0.5	0.4	27	1	1	1	0	0	1
14	0.5	0.5	0.5	0.1	0.4	0.5	-	-	-	-	-	-	-

Table 7. Rule table of T-S gate 2.

rules	x_4	x_5	y_2			rules	x_4	x_5	y_2		
			0	0.5	1				0	0.5	1
1	0	0	1	0	0	6	0.5	1	0	0	1
2	0	0.5	0.4	0.4	0.2	7	1	0	0.1	0.2	0.7
3	0	1	0.1	0.1	0.8	8	1	0.5	0	0	1
4	0.5	0	0.8	0.1	0.1	9	1	1	0	0	1
5	0.5	0.5	0.1	0.5	0.4	-	-	-	-	-	-

Table 8. Rule table of T-S gate 3.

rules	y_1	y_2	y_3			rules	y_1	y_2	y_3		
			0	0.5	1				0	0.5	1
1	0	0	1	0	0	6	0.5	1	0	0	1
2	0	0.5	0.4	0.5	0.1	7	1	0	0.1	0.2	0.7
3	0	1	0.1	0.1	0.8	8	1	0.5	0	0	1
4	0.5	0	0.8	0.1	0.1	9	1	1	0	0	1
5	0.5	0.5	0.1	0.5	0.4	-	-	-	-	-	-

Taking Table 6 as an example, each row in rules1~27 represents a G_1 gate rule. For example, in rule1, the fault states of bottom events x_1, x_2, and x_3 are 0, 0, and 0 respectively, then the occurrence probability of fault state 0 of y_1 is $P^1(y_1 = 0) = 1$, the occurrence probability of fault state 0.5 of y_1 is $P^1(y_1 = 0.5) = 0$, the occurrence probability of fault state 1 of y_1 is $P^1(y_1 = 1) = 0$. Under the same rule, the sum of the occurrence possibilities of each fault state of the superior event y_1 is 1, that is, $P^1(y_1 = 0) + P^1(y_1 = 0.5) + P^1(y_1 = 1) = 1$.

6.2. System reliability optimization model

According to the T-S fault tree and the corresponding rule gate of the system, a system reliability optimization model is constructed with the lowest system fault probability as the objective function and the overall cost of the system as the constraint. Among them, the system cost is the sum of the expenses of each component unit, its connectors, and switching equipment. The unit cost increases nonlinearly with the improvement of its reliability. The objective function and cost constraint expression are as follows:

$$\min \sum_{q=2}^{3} P\left(T = T_q\right) = \sum_{l=1}^{9} P_o^l P^l\left(T = 0.5\right) + \sum_{l=1}^{9} P_o^l P^l\left(T = 1\right) \tag{13}$$

$$\text{s.t} \sum_{i=1}^{5} \alpha_i \left\{ \frac{-\mu}{\ln\left[1 - \sum_{\alpha_i = 2}^{3} P\left(x_i^{\alpha_i}\right)\right]} \right\}^{\beta_i} \times \left[n_i + \exp\left(\frac{n_i}{4}\right) \right] \le C_0 \tag{14}$$

where $P(T = T_q)$ is the fault probability of top event T when its fault state is T_q; $P^l(T = 0.5)$ and $P^l(T = 1)$

respectively represent the probability when the fault state of top event T is 0.5 and 1 in the rule l; P_0^l is the execution degree of T-S rule; $P(x_i^{a_i})$ is the failure probability of bottom event x_i when its failure state is $x_i^{a_i}$; n_i is the redundancy number of the element; μ is the fault-free operation time, taken $\mu = 1000$ h; C_0 is the constraint value of system cost, taken $C_0 = 175$. α_i and β_i can be seen in Table 9.

Table 9. Constraint parameter values of α_i and β_i.

i	1	2	3	4	5
$10^5\alpha_i$	2.540×10^{-5}	2.483×10^{-5}	6.420×10^{-5}	7.160×10^{-5}	2.417×10^{-5}
β_i	1.500	1.500	1.500	1.500	1.500

6.3. Fitness function

The penalty function is one of the main constraint optimization methods available at present, whose core idea is to transform the original constrained optimization problem into an unconstrained problem by constructing auxiliary functions. In this paper, the cost constraint in the system reliability optimization model is transformed into an unconstrained optimization problem by introducing a penalty function. That is, a penalty factor is added to the fitness value of the salps that does not satisfy the cost constraints, so that the infeasible solution can be eliminated in the process of evolution. In this paper, the maximum probability value of system fault N (N = 1) is used as the penalty factor, and the failure probability fitness function is constructed as follows:

$$fitness = \begin{cases} \sum_{q=2}^{3} P\left(T = T_q\right) & C \le C_0 \\ \sum_{q=2}^{3} P\left(T = T_q\right) + N & Otherwise \end{cases} \tag{15}$$

6.4. Results comparison and analysis

The DCORSSA-PSO algorithm is compared with SSA, PSO, and GWO algorithms. Set the maximum number of iterations of the above five algorithms T = 500. And the reliability optimization results of the algorithms are shown in Table 10.

Table 10. Optimization results of four algorithms.

Optimized	SSA		DCORSSA-PSO		PSO		GWO	
parameters	$P(x_i)$	n_i	$P(x_i)$	n_i	$P(x_i)$	n_i	$P(x_i)$	n_i
x_1	1.28×10^{-1}	3	1.09×10^{-1}	3	1.58×10^{-1}	3	1.37×10^{-1}	3
x_2	9.22×10^{-2}	3	1.08×10^{-1}	3	1.58×10^{-1}	3	1.68×10^{-1}	3
x_3	1.38×10^{-1}	3	9.00×10^{-2}	3	1.07×10^{-1}	3	1.88×10^{-1}	3
x_4	1.26×10^{-1}	3	1.41×10^{-1}	3	1.70×10^{-1}	3	1.57×10^{-1}	3
x_5	1.16×10^{-1}	3	9.66×10^{-2}	3	1.11×10^{-1}	3	1.59×10^{-1}	3
P	5.03×10^{-3}		3.71×10^{-3}*		8.03×10^{-3}		1.22×10^{-2}	
C	175.00		175.00		115.76		105.89*	
Time/s	4.91×10^{-1}*		6.05×10^{-1}		8.17×10^{-1}		5.09×10^{-1}	

According to the optimization results in Table 10, when taking the minimum failure probability of the system as the objective function, the system failure probability optimized by the DCORSSA-PSO algorithm is lower than that of other algorithms including SSA, PSO, GWO, which proves the feasibility and superiority of the improved algorithm. In terms of running time, SSA has the shortest one, while DCORSSA-PSO has the longest one. This indicates that the multi-strategy improvement of DCORSSA-PSO spend more time, but the running time of DCORSSA-PSO is less than one second which can fulfill the needs of practical engineering.

In addition, in order to more intuitively show the reliability optimization process of the four algorithms, the iterative curve is shown in Figure 4.

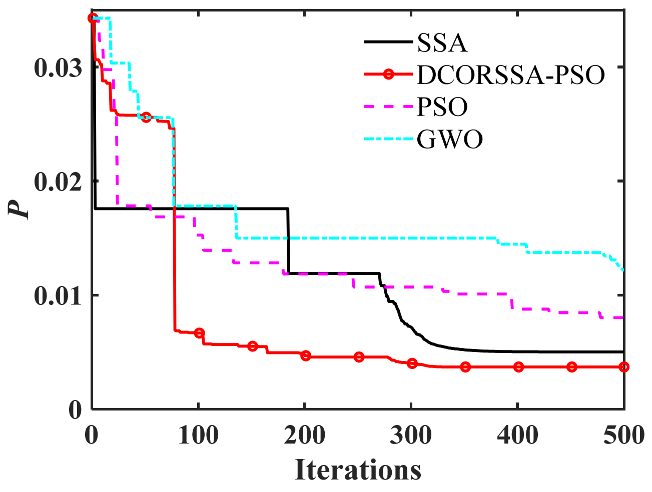


Figure 4. Optimization comparison curves.

To test the stableness of the DCORSSA-PSO algorithm, let all the algorithms run 30 times at the same initial condition. Table 11 shows the statistical results of failure probability.

Table 11. Statistical results of five algorithms.

Measure	SSA	DCORSSA-PSO	PSO	GWO
Mean	6.956×10^{-3}	3.746×10^{-3}*	5.196×10^{-3}	8.304×10^{-3}
Best	3.710×10^{-3}	3.709×10^{-3}*	4.387×10^{-3}	4.404×10^{-3}
Std	3.703×10^{-3}	3.886×10^{-5}*	8.625×10^{-4}	2.518×10^{-3}
Time/s	5.466×10^{-1}*	6.795×10^{-1}	8.975×10^{-1}	5.617×10^{-1}

From Table 11, we can find that in a statistical sense, DCORSSA-PSO compared with SSA, PSO and GWO still get the best result of failure probability including mean value, best value and standard deviation except for running time. The average failure probability obtained by DCORSSA-PSO algorithm relatively reduced by 46.14% compared to SSA, which shows that DCORSSA-PSO greatly improves the optimization performance of SSA by integrating the multi-strategy improvement.

In this paper, a box plot is used to analyze the data distribution of the system failure probability. The box plot consists of five parts: upper limit, upper quartile, median, lower quartile and lower limit. The upper limit is connected to the upper quartile with a dashed line, and same to the lower limit and the lower quartile. The center mark indicates the median. Figure 5 shows the boxplot of different algorithms.

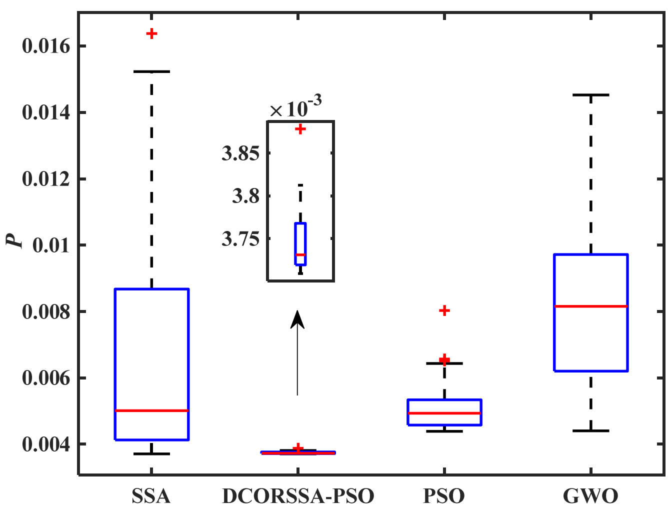


Figure 5. Boxplot of the different algorithms.

In the statistical result of Figure 5, y-axis *P* means the failure probability, and it can be found that DCORSSA-PSO has the lowest median of *P* value, which means that the reliability calculated by DCORSSA-PSO is the highest. In addition, the box obtained by DCORSSA-PSO is very compact, that is, the range of the box formed between the upper quartile and the lower quartile is the smallest, indicating that DCORSSA-PSO has less volatility compared to the datasets of other algorithms. Therefore, DCORSSA-PSO outperforms other algorithms. On the other hand, outliers (+) appear in the failure probability optimized by SSA, DCORSSA-PSO and PSO, indicating that further research in performance improvement is needed for DCORSSA-PSO.

7. Conclusions and future works

This paper proposes a DCORSSA-PSO algorithm that hybridizes dimension-by-dimension centroid opposition-based learning strategy, random factor and PSO's social learning strategy based on standard SSA. The improved algorithm mainly improves the standard SSA algorithm in three parts: a) a dimension-by-dimension centroid opposition-based learning strategy is added to the food source update, which can expand the population search range, strengthen the dimension evolution information, and enhance the ability to jump out of the local solutions; b) random factor is added in the update equation of followers to enhance the diversity of population distribution; c) drawing on the experience of PSO's social learning strategy, in the update equation of followers, the food source is added to directly guide the followers to improve the convergence speed of the algorithm. The comparison results in the synthesis of ten standard test functions and the reliability optimization example show that the DCORSSA-PSO algorithm is superior to other algorithms in optimization, which proves that the above improvement strategy has good feasibility and superiority to improve the optimization performance of the SSA algorithm. As a future plan, the method of increasing the diversity of the population will be introduced into the research of DCORSSA-PSO such as the levy-flight theory, chaos mapping. At the same time, DCORSSA-PSO can be employed to optimize pattern classification, fuzzy control, machine learning, etc.

Acknowledgments

This project is supported by the National Natural Science Foundation of China (Grant No. 51975508), Natural Science Foundation of Hebei Province (Grant No. E2021203061).

Conflict of interest

All authors declare that there is no conflict of interests in this paper.

References

1. B. Nautiyal, R. Prakash, V. Vimal, G. Liang, H. Chen, Improved salp swarm algorithm with mutation schemes for solving global optimization and engineering problems, *Eng. Comput. Ger.*, **80** (2021), 35415–35439, https://doi.org/10.1007/s00366-020-01252-z
2. E. H. Houssein, M. A. Mahdy, D. Shebl, A. Manzoor, R. Sarkar, W. M. Mohamed, An efficient slime mould algorithm for solving multi-objective optimization problems, *Expert Syst. Appl.*, **187** (2022), 115870. https://doi.org/10.1016/j.eswa.2021.115870
3. G. Dhiman, K. K. Singh, M. Soni, A. Nagar, M. Dehghani, A. Slowik, et al., MOSOA: A new multi-objective seagull optimization algorithm, *Expert Syst. Appl.*, **167** (2021), 114150. https://doi.org/10.1016/j.eswa.2020.114150
4. J. Xue, B. Shen, A novel swarm intelligence optimization approach: sparrow search algorithm, *Syst. Sci. Control Eng.*, **8** (2020), 22–34. https://doi.org/10.1080/21642583.2019.1708830
5. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Icnn95-international Conference on Neural Networks*, 1995. https://doi.org/10.1109/ICNN.1995.488968
6. A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in *Proceedings of the first European conference on artificial life*, (1991), 134–142
7. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Glob. Optim.*, **39** (2007), 459–471. https://doi.org/10.1007/s10898-007-9149-x
8. X. Yang, *Engineering Optimization*: *An Introduction with Metaheuristic Application*, 2010. https://doi.org/10.1002/9780470640425.ch2
9. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
10. G. Dhiman, V. Kumar, Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems, *Knowl.-based Syst.*, **165** (2019), 169–196. https://doi.org/10.1016/j.knosys.2018.11.024
11. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: a new method for stochastic optimization, *Future* G*ener.* C*omp. Sy.*, **111** (2020), 300–323. https://doi.org/10.1016/j.future.2020.03.055
12. S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Software*, **114** (2017), 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002

13. R. A. Ibrahim, A. A. Ewees, D. Oliva, M. A. Elaziz, S. F. Lu, Improved salp swarm algorithm based on particle swarm optimization for feature selection, *J. Amb. Intel. Hum. Comp.*, **10** (2019), 3155–3169. https://doi.org/10.1007/s12652-018-1031-9
14. A. G. Hussien, A. E. Hassanien, E. H. Houssein, Swarming behaviour of salps algorithm for predicting chemical compound activities, in *the 8th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS)*, (2017), 315–320. https://doi.org/10.1109/intelcis.2017.8260072
15. S. Wang, H. Jia, X. Peng, Modified salp swarm algorithm based multilevel thresholding for color image segmentation, *Math. Biosci. Eng.*, **17** (2020), 700–724. https://doi.org/10.3934/mbe.2020036
16. S. Zhao, P. Wang, A. A. Heidari, Chen, W. He, S. Xu, Performance optimization of salp swarm algorithm for multi-threshold image segmentation: Comprehensive study of breast cancer microscopy, *Comput. Biol. Med.*, **139** (2021), 105015. https://doi.org/10.1016/j.compbiomed.2021.105015
17. A. M. Tudose, I. I. Picioroaga, D. O. Sidea, C. Bulac, Solving single-and multi-objective optimal reactive power dispatch problems using an improved salp swarm algorithm, *Energies*, **14** (2021), 1222–1222. https://doi.org/10.3390/en14051222
18. H. M. Kanoosh, E. H. Houssein, M. M. Selim, Salp swarm algorithm for node localization in wireless sensor networks, *J. Comput. Netw. Commun.*, **2019** (2019), 1–12. https://doi.org/10.1155/2019/1028723
19. G. I. Sayed, G. Khoriba, M. H. Haggag, A novel chaotic salp swarm algorithm for global optimization and feature selection, *Appl. Intel.*, **48** (2018), 3462–3481. https://doi.org/10.1007/s10489-018-1158-6
20. A. Abbassi, R. Abbassi, A. A. Heidari, D. Oliva, H. L. Chen, A. Habib, et al., Parameters identification of photovoltaic cell models using enhanced exploratory salp chains-based approach, *Energy*, **198** (2020), 117333. https://doi.org/10.1016/j.energy.2020.117333
21. N. Singh, L. H. Son, F. Chiclana, J. P. Magnot, A new fusion of salp swarm with sine cosine for optimization of non-linear functions, *Eng. Comput. Ger.*, **36** (2020), 185–212. https://doi.org/10.1007/s00366-018-00696-8
22. M. A. Syed, R. Syed, Weighted salp swarm algorithm and its applications towards optimal sensor deployment, *J. King Saud. Univ.-Com.*, (2019). https://doi.org/10.1016/j.jksuci.2019.07.005
23. N. Singh, S. B. Singh, E. H. Houssein, Hybridizing salp swarm algorithm with particle swarm optimization algorithm for recent optimization functions, *Evol. Intell.*, (2020), 1–34. https://doi.org/10.1007/s12065-020-00486-6
24. Q. Zhang, Z. Wang, A. A. Heidari, W. Gui, Q. Shao, H. Chen, et al. Gaussian Barebone salp swarm algorithm with stochastic fractal search for medical image segmentation: a COVID-19 case study, *Comput. Biol. Med.*, **139** (2021), 104941. https://doi.org/10.1016/j.compbiomed.2021.104941
25. Y. Liu, Y. Shi, H. Chen, A. Asghar Heidari, W. Gui, M. Wang, et al., Chaos-assisted multi-population salp swarm algorithms: framework and case studies, *Expert Syst. Appl.*, **168** (2021), 114369. https://doi.org/10.1016/j.eswa.2020.114369
26. H. Zhang, Z. Wang, W. Chen, A. A. Heidari, M. Wang, X. Zhao, et al., Ensemble mutation-driven salp swarm algorithm with restart mechanism: Framework and fundamental analysis, *Expert Syst. Appl.*, **165** (2021), 113897. https://doi.org/10.1016/j.eswa.2020.113897

27. S. Zhao, P. Wang, X. Zhao, H. Turabieh, M. Mafarja, H. Chen, Elite dominance scheme ingrained adaptive salp swarm algorithm: a comprehensive study, *Eng. Comput. Ger.*, **165** (2021), 113897. https://doi.org/10.1007/s00366-021-01464-x
28. H. Zhang, T. Liu, X. Ye, A. A. Heidari, G. Liang, H. Chen, et al, Differential evolution-assisted salp swarm algorithm with chaotic structure for real-world problems, *Eng. Comput. Ger.*, (2022), 1–35. https://doi.org/10.1007/s00366-021-01545-x
29. J. Xia, H. Zhang, R. Li, Z. Wang, Z. Cai, Z. Gu, et al, Adaptive barebones salp swarm algorithm with quasi-oppositional learning for medical diagnosis systems: a comprehensive analysis, *J. Bionic. Eng.*, **19** (2022), 1–17. https://doi.org/10.1007/s42235-021-00114-8
30. H. Zhang, Z. Cai, X. Ye, M. Wang, F. Kuang,·H. Chen, et al, A multi-strategy enhanced salp swarm algorithm for global optimization, *Eng. Comput. Ger.*, **1** (2020). https://doi.org/10.1007/s00366-020-01099-4
31. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893
32. M. M. Saafan, E. M. El-Gendy, IWOSSA: An improved whale optimization salp swarm algorithm for solving optimization problems, *Expert Syst. Appl.*, **176** (2021), 114901. https://doi.org/10.1016/j.eswa.2021.114901
33. H. R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce*, **1** (2005), 695–701. https://doi.org/10.1109/CIMCA.2005.1631345
34. S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad, G. F. Naterer, Computing opposition by involving entire population, in *2014 IEEE congress on evolutionary computation* (*CEC*), (2014), 1800–1807. https://doi.org/10.1109/CEC.2014.6900329
35. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol. Comput.*, **1** (2011), 3–18. https://doi.org/10.1016/j.swevo.2011.02.002
36. S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, *Inf. Sci.*, **180** (2010), 2044–2064. https://doi.org/10.1016/j.ins.2009.12.010
37. P. Jiang, J. Cheng, Q. Zhou, L. Shu, J. Hu, Variable-fidelity lower confidence bounding approach for engineering optimization problems with expensive simulations, *AIAA J.*, **57** (2019), 5416–5430. https://doi.org/10.2514/1.J058283
38. M. A. Mellal, E. Zio, System reliability-redundancy optimization with cold-standby strategy by an enhanced nest cuckoo optimization algorithm, *Reliab. Eng. Syst. Safe*, **201** (2020), 106973. https://doi.org/10.1016/j.ress.2020.106973
39. B. N. Chebouba, M. A. Mellal, S. Adjerid, Fuzzy multiobjective system reliability optimization by genetic algorithms and clustering analysis, *Qual. Reliab. Eng. Int.*, **37** (2020), 1484–1503. https://doi.org/10.1002/qre.2809
40. A. Samanta, K. Basu, A novel particle swarm optimization with fuzzy adaptive inertia weight for reliability redundancy allocation problems, *Intell. Decis. Technol.*, **13** (2019), 91–99. https://doi.org/10.3233/IDT-190357

41. T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE T. Syst. Man CY.*, **15** (1985), 116–132. https://doi.org/10.1109/tsmc.1985.6313399
42. H. Song, H. Zhang, C. Chan, Fuzzy fault tree analysis based on T-S model with application to INS/GPS navigation system, *Soft Comput.*, **13** (2009): 31–40. https://doi.org/10.1007/s00500-008-0290-3
43. C. Yao, B Wang, D. Chen, Reliability optimization of multi-state hydraulic system based on T-S fault tree and extended PSO algorithm, *IFAC Proceed. Vol.*, **46** (2013), 463–468. https://doi.org/10.3182/20130410-3-CN-2034.00012