Mathematical Biosciences and Engineering

MBE, 19 (3): 2240–2285.
DOI: 10.3934/mbe.2022105
Received: 27 October 2021
Revised: 02 December 2021
Accepted: 16 December 2021
Published: 04 January 2022

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

Research article

DTSMA: Dominant Swarm with Adaptive T-distribution Mutation-based Slime Mould Algorithm

Shihong Yin[1,2,3], Qifang Luo[1,2,3], Yanlian Du[4,5] and Yongquan Zhou[1,2,3]

[1] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China
[2] Key Laboratory of Guangxi High Schools Complex System and Computational Intelligence, Nanning 530006, China
[3] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China
[4] College of Information and Communication Engineering, Hainan University, Haikou 570228, China
[5] State Key Laboratory of Marine Resources Utilization in South China Sea, Hainan University, Haikou 570228, China

* **Correspondence:** Email: yongquanzhou@126.com; Tel: +8613607882594; Fax: +867713265523.

Abstract: The slime mould algorithm (SMA) is a metaheuristic algorithm recently proposed, which is inspired by the oscillations of slime mould. Similar to other algorithms, SMA also has some disadvantages such as insufficient balance between exploration and exploitation, and easy to fall into local optimum. This paper, an improved SMA based on dominant swarm with adaptive t-distribution mutation (DTSMA) is proposed. In DTSMA, the dominant swarm is used improved the SMA's convergence speed, and the adaptive t-distribution mutation balances is used enhanced the exploration and exploitation ability. In addition, a new exploitation mechanism is hybridized to increase the diversity of populations. The performances of DTSMA are verified on CEC2019 functions and eight engineering design problems. The results show that for the CEC2019 functions, the DTSMA performances are best; for the engineering problems, DTSMA obtains better results than SMA and many algorithms in the literature when the constraints are satisfied. Furthermore, DTSMA is used to solve the inverse kinematics problem for a 7-DOF robot manipulator. The overall results show that DTSMA has a strong optimization ability. Therefore, the DTSMA is a promising metaheuristic optimization for global optimization problems.

Keywords: Slime mould algorithm; t-distribution mutation; functions optimization; engineering problems; metaheuristic optimization

1. Introduction

With the development of technology and society, more and more highly complex and challenging practical optimization problems need to be solved. Most of the traditional optimization methods are based on gradient or derivative information, such as Newton's method [1,2], conjugate gradient method [3], which have the advantages of theoretical soundness and fast convergence and can be used to solve optimization problems in some engineering fields. However, these methods tend to be based on problem-specific characteristics, are difficult to meet the needs of a large number of practical problems, and it easily becomes trapped into local optima when used to solve complex, highly nonlinear, and multi-peak complex problems [4]. To overcome these problems, metaheuristic optimization algorithms were introduced, which can help to solve optimal or near-optimal solutions of complex functional and real-world problems with the iterative process of the algorithm. Unlike traditional methods, metaheuristic algorithms have a stochastic and gradient-free mechanism, require minimal mathematical analysis, and use only inputs and outputs to consider and solve the optimization problem [5]. This is one of the fundamental advantages of metaheuristic algorithms, giving them a high degree of flexibility in solving various problems.

The metaheuristic optimization methods can be divided into three categories from the principle of algorithms: evolution-based, physics-based, and swarm intelligence-based [6]. Evolution-based algorithms are proposed to simulate Darwinian biological evolution and mainly include Genetic Algorithm (GA) [7] and Differential Evolution (DE) [8]. The physics-based algorithms are inspired by the laws of physics and mainly include Simulated Annealing (SA) [9], Quantum Search Algorithm (QSA) [10], Big Bang-Big Crunch (BBBC) [11], Artificial Chemical Reaction Optimization Algorithm (ACROA) [12], Lightning Search Algorithm (LSA) [13], Multi-Verse Optimizer (MVO) [14], Heat transfer search (HTS) [15], Atom Search Optimization (ASO) [16], and Equilibrium optimizer (EO) [17]. Swarm intelligence-based algorithms are proposed to simulate the collaborative behavior of natural biological swarms. Representative algorithms include Particle Swarm Optimization (PSO) [18], Artificial Bee Colony Algorithm (ABC) [19], Teaching-Learning Based Optimization (TLBO) [20], Gray Wolf Optimizer (GWO) [21], Whale Optimization Algorithm (WOA) [22], Salp Swarm Algorithm (SSA) [23], Social Spider Optimization (SSO) [24], Seagull Optimization Algorithm (SOA) [25], Marine Predators Algorithm (MPA) [26], Harris Hawks Optimization (HHO) [27], Bald Eagle Search (BES) [28], Slime Mould Algorithm (SMA) [29], Chameleon Swarm Algorithm (CSA) [30], and so on.

It is worth noting that, according to the No Free Lunch (NFL) theorem [31], no algorithm performs well on all problems, and each algorithm has its own strengths and weaknesses, which are applied to different real-world problems to obtain better results. As a result, applying improved algorithms to specific problems has become a hot topic of current research. For example, Zhang et al. [32] proposed a state transition simulated annealing algorithm (STASA) that introduces a new elementary breakpoint operator and neighborhood search structure in SA to solve multiple traveling salesman problems, and experimental results show that the improved algorithm outperforms other state-of-the-art algorithms. Yu et al. [33] proposed a performance-guided JAYA (PGJAYA) algorithm for extracting parameters of different PV models, and the performance of PGJAYA was evaluated on a standard dataset of three PV models, and the results showed that PGJAYA has excellent performance. Fan et al. [34] proposed an improved Harris Hawk Optimization algorithm based on domain centroid opposite-based learning (NCOHHO), which was applied to feedforward neural network training and achieved good results in classification applications.

The Slime Mould Algorithm (SMA) is a metaheuristic algorithm inspired by slime mould oscillation proposed by Li et al. in 2020 [29]. It has been applied to many fields in less than two years because it simulates the unique oscillatory foraging behavior of the slime mould and has superior performance. For example, Ewees et al. [35] applied the firefly algorithm (FA) and SMA hybrid algorithm (SMAFA) to the feature selection (FS). Abdel-basset et al. [36] applied the binary SMA (BSMA) to the FS problem. Abdel-basset et al. [37] proposed a hybrid method based on threshold technology (HSMA_WOA) to overcome the image segmentation problem (ISP) of chest X-ray images of COVID-19. Zhao et al. [38] proposed a Renyi's entropy multi-threshold image segmentation method based on improved slime mold algorithm (DASMA). Naik et al. [39] applied an improved SMA (LSMA) to the ISP. Yousri et al. [40] proposed a novel hybrid algorithm of marine predator algorithm (MPA) and SMA (HMPA) to solve the ISP. Mostafa et al. [41] applied SMA to the single-diode and dual-diode models of photovoltaic cells. El-Fergany [42] studied the performance of SMA and its improved version (ImSMA) in photovoltaic parameter extraction. Liu et al. [43] proposed a SMA that integrates Nelder-Mead simplex strategy and chaotic mapping to identify photovoltaic solar cell parameters. Kumar et al. [44] applied SMA to the parameter extraction of photovoltaic cells and proved the superiority of SMA. Agarwal et al. [45] applied SMA to path planning and obstacle avoidance problem of mobile robots. Rizk-Allah et al. [46] proposed a chaos-opposition-enhanced SMA (CO-SMA) to minimize the energy costs of wind turbines at high-altitude sites. Hassan et al. [47] proposed an improved version of the SMA (ISMA) and applied it to efficiently solve economic and emission dispatch (EED) problem with single and dual objectives, and compared it with five algorithms on five test systems. Wei et al. [48] proposed an improved SMA (ISMA) for optimal reactive power dispatch (ORPD) problem in power systems, and achieved better results than the well-known algorithms on power test systems with IEEE 57 bus, IEEE 118 bus and IEEE 300 bus. Abdollahzadeh et al. [49] proposed a binary version of SMA to solve the 0-1 knapsack problem; Zubaidi et al. [50] combined SMA and artificial neural network for urban water demand prediction; Chen et al. [51] combined K-means clustering and chaotic SMA with support vector regression to obtain higher prediction accuracy. Ekinci et al. [52] applied SMA to the power system stabilizer design (PSSD); Wazery et al. [53]. Combined SMA and K-nearest neighbor for disease classification and diagnosis system. Premkumar et al. [54] proposed a multi-objective version of the SMA (MOSMA) for solving complex real-world multi-objective engineering optimization problems, which has better performance compared to other well-known multi-objective algorithms. Yu et al. [55] proposed an improved SMA (WQSMA), which used quantum rotation gate (QRG) and water cycle operator to improve the robustness of the original SMA, so as to balance the exploration and exploitation ability. The effectiveness of WQSMA on CEC2014 and three engineering problems was verified. Houssein et al. [56] proposed a hybrid SMA and adaptive guided differential evolution (AGDE) algorithm, namely SMA-AGDE, which makes a good combination of SMA's exploitation ability and AGDE's exploration ability, and verified the effectiveness of SM-AGDE through CEC2017 and three engineering design problems.

As mentioned above, many scholars have only improved SMA for specific problems, and the generalization ability of the proposed algorithms has yet to be tested. Yu et al. [55] and Houssein et al. [56] respectively used QRG and AGDE to enhance the exploration ability of SMA to address the shortcomings of SMA and achieved good results. In this paper, a novel improved slime mould algorithm DTSMA based on dominant swarm and nonlinear adaptive t-distribution mutation is proposed based on the improved experience of WQSMA and SMA-AGDE. The dominant swarm enhanced the exploitation ability of SMA, and the t-distribution mutation enhanced the exploration ability of SMA. In order to further improve the exploitation ability of SMA, a new exploitation formula is added to DTSMA. The main contributions of this paper are as follows.

(1) It is verified that the dominant swarm strategy can improve the convergence rate of SMA.

(2) The proposed nonlinear adaptive t-distribution mutation mechanism can expand the search range of SMA in the iterative process, increase the difference of search agents, improve the global search ability of SMA, and avoid falling into local optimal.

(3) The proposed new exploitation mechanism is effectively combined with that of SMA.

(4) The DTSMA is compared with other advanced metaheuristic algorithms on CEC2019, and the advantages of DTSMA in convergence speed and solution accuracy are verified.

(5) The performance of DTSMA is tested on eight classical engineering application problems and the inverse kinematics problems of a 7-DOF robot manipulator.

In this paper, the CEC2019 functions and eight constrained engineering design problems are selected as test cases and compared with twenty-two well-known algorithms on CEC2019 and with SMA and improved algorithms in the literature on engineering instances. Experimental results show that DTSMA has strong search ability and can obtain better solutions than most algorithms under the condition that constraints are satisfied.

The rest of this paper is organized as follows: Section 2 briefly describes the principle and characteristics of SMA. Section 3 describes the principle of DTSMA and its difference from SMA in detail. Section 4 presents the experimental configuration, the comparative experimental results of the CEC2019 functions, and its statistical analysis. In section 5, DTSMA is used to optimize eight engineering problems, i.e., three-bar truss, cantilever beam, pressure vessel, tension/compression spring, welded beam, speed reducer, multi-disc clutch brake, and car side crash problem. In section 6, DTSMA is used to solve the inverse kinematics problems of a 7-DOF robot manipulator. Section 7 presents the discussion, conclusions and future work.

2. Slime mould algorithm (SMA)

2.1. Inspiration

SMA is an interesting swarm-based meta-heuristic algorithm proposed by Li et al. in 2020 [29]. It simulates the behavior and morphological changes of slime mould in foraging to find the best solution. The slime mould relies mainly on propagating waves generated by biological oscillators to modify the cytoplasmic flow in the veins to approach a higher food concentration, then surrounds it and secretes enzymes to digest.

2.2. Mathematical model

During the foraging process of slime mould, individuals can approach the food based on the odor in the air. The greater the concentration of food odor, the stronger the bio-oscillator wave, the faster the cytoplasmic flow, and the thicker the vein-like tubes formed by the slime mould. The mathematical model for updating the location of slime mould is as Eq. (1).

$$\overrightarrow{X(t+1)} = \begin{cases} rand \cdot (UB - LB) + LB & rand < z \\ \overrightarrow{X_b(t)} + \overrightarrow{vb} \cdot \left(\overrightarrow{W} \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)} \right) & r < p \\ \overrightarrow{vc} \cdot \overrightarrow{X(t)} & r \geq p \end{cases} \tag{1}$$

where LB and UB denote the lower and upper bounds of the search range, $rand$ and r denote random numbers in [0,1], and z is a parameter that the original authors did a lot of experiments and suggested to take 0.03, $\overrightarrow{X_b}$ indicates the location where the highest concentration of food odor is currently found, $\overrightarrow{vb}$ and $\overrightarrow{vc}$ are parameters, $\overrightarrow{vb}$ takes values in $[-a, a]$, $\overrightarrow{vc}$ decreases linearly from 1 to 0 with the number of iterations t, $\overrightarrow{W}$ indicates the thickness of the vein-like vessels formed by the slime mould, $\overrightarrow{X_A}$ and $\overrightarrow{X_B}$ are two randomly selected agents positions in the population, $\overrightarrow{X}$ indicates the current position of the slime mould.

The value of p is calculated as Eq. (2).

$$p = \tanh |S(i) - DF| \tag{2}$$

where $i \in 1, 2, ..., n$, $S(i)$ denotes the fitness $\overrightarrow{X}$, DF denotes the best fitness obtained so far.

The value of a in the range of $\overrightarrow{vb}$ is calculated as Eq. (3).

$$a = \operatorname{arctanh}\left(1 - t/\max\_t\right) \tag{3}$$

where $\max\_t$ indicates the maximum number of iterations.

The formula of $\overrightarrow{W}$ is calculated as Eq. (4).

$$\overrightarrow{W(SmellIndex(i))} = \begin{cases} 1 + r \cdot \log\left(\dfrac{bF - S(i)}{bF - wF} + 1 \right) & condition \\ 1 - r \cdot \log\left(\dfrac{bF - S(i)}{bF - wF} + 1 \right) & others \end{cases} \tag{4}$$

$$SmellIndex = sort(S) \tag{5}$$

where $condition$ represents that $S(i)$ ranks first half of the population, r means a random number in [0,1], bF represents the optimal fitness obtained in the iterative process currently, wF represents the worst fitness obtained in the iterative process currently, $SmellIndex$ denotes the result of the ascending order of fitness values (in the minimization problem).

2.3. Characteristics of SMA

The slime mould approximation food behavior shown in Eq. (1), the individuals position $\overrightarrow{X}$ can be updated according to the best position $\overrightarrow{X_b}$ obtained so far, while the fine-tuning of parameters $\overrightarrow{vb}$, $\overrightarrow{vc}$ and $\overrightarrow{W}$ can change the individuals position and $rand$ allows the search agents to form a search vector of any angle.

Algorithm 1 Pseudo-code of SMA

```
1.  Initialize the parameters $z, n, d, max\_t$;
2.  Initialize the positions of slime mould $\overrightarrow{X_i}(i = 1, 2, ..., n)$;
3.  **While** ($t \le max\_t$)
4.      Calculation the fitness $\vec{S}$ of all slime mould;
5.      Sort the fitness $\vec{S}$;
6.      Update $bF, wF, DF, \overrightarrow{X_b}$;
7.      Calculate the $\overrightarrow{W}$ by Eq. (4);
8.      Update $p, \overrightarrow{vb}, \overrightarrow{vc}, A, B$;
9.      **For** each search agents
10.         Update positions by Eq. (1);
11.     **End For**
12.     $t = t + 1$;
13. **End While**
14. **Return** $DF, \overrightarrow{X_b}$;
```

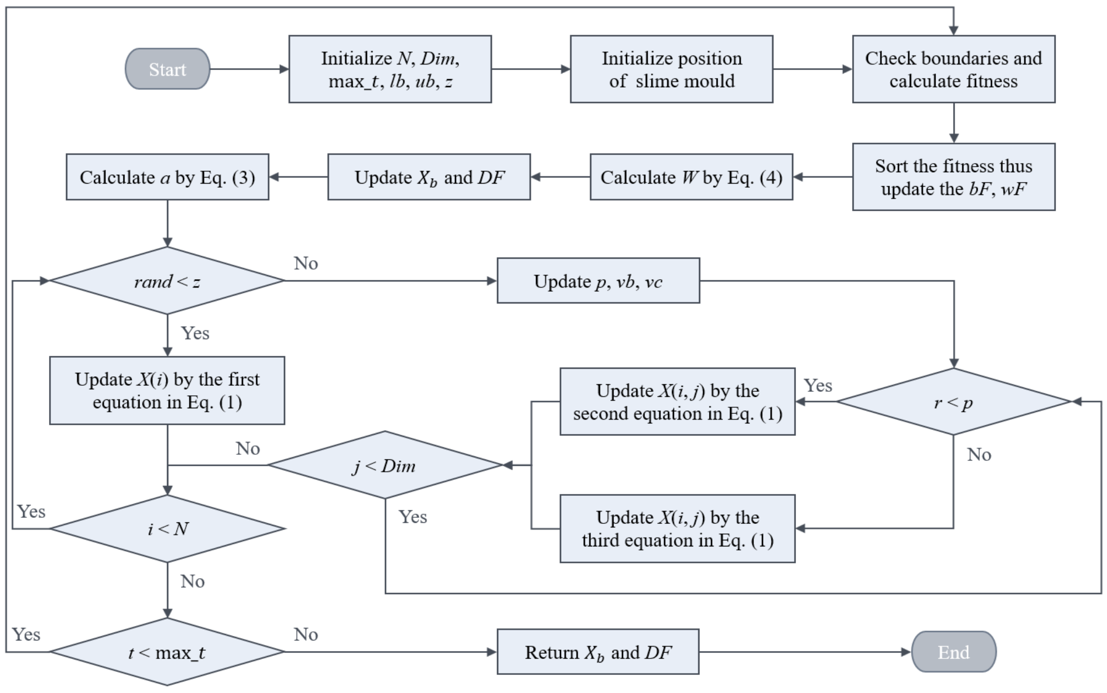
Start
Initialize N, Dim, max_t, lb, ub, z
Initialize position of slime mould
Check boundaries and calculate fitness
Sort the fitness thus update the bF, wF
Calculate W by Eq. (4)
Update X_b and DF
Calculate a by Eq. (3)
rand < z
No
Yes
Update p, vb, vc
r < p
Yes
No
Update X(i) by the first equation in Eq. (1)
Update X(i, j) by the second equation in Eq. (1)
Update X(i, j) by the third equation in Eq. (1)
j < Dim
No
Yes
i < N
Yes
No
t < max_t
Yes
No
Return X_b and DF
End

Figure 1. Flow chart of the SMA [29].

At the beginning of the SMA, the individual positions are scattered, the value of p tends to 1, and the slime mould is mainly explored by the second equation in Eq. (1). As the number of iterations increases, the individual positions are gradually close together, the vein-like vessels of the slime population are gradually formed, the individual fitness value $S(i)$ is gradually approached with the current optimal fitness value DF, the value of p tends to 0, and the slime mould are mainly exploited by the third equation in Eq. (1). In addition, a stochastic strategy was introduced into the search process of SMA so that the algorithm maintains some exploration ability even during the exploitation phase. In SMA, there are no velocity settings for the agents of the slime mould and the population is not divided into hierarchies or subpopulations. All search agents are simply and equally selected close to or far from the current best location $\overrightarrow{X_b}$. Furthermore, the position is updated using only the best positions obtained so far and not using the historical best position information of individuals. The pseudo-code of the SMA is shown in Algorithm 1 [29], and the flow chart is expressed in Figure 1.

3. Improved slime mould algorithm (DTSMA)

3.1. Dominant swarm

In the process of solving the optimization problem, SMA does not use the information of the individual optimal position of slime mould to update the solution, and may miss a good opportunity to find the global optimal. In DTSMA, in order to record the individual historical optimal position information, the dominant swarm $\overrightarrow{X_{\text{good}}}$ and its fitness value $\overrightarrow{S_{\text{good}}}$ are defined to store the historical optimal information. After the position is updated, the updated position $\vec{X}$ is compared with the position in the dominant swarm $\overrightarrow{X_{\text{good}}}$, and the greedy selection strategy is used to reserve the better position to the dominant swarm. In the exploration phase, DTSMA uses the individual historical optimal $\overrightarrow{X_{\text{good}A}}$, $\overrightarrow{X_{\text{good}B}}$ and the population historical optimal $\overrightarrow{X_{\text{good}b}}$ found so far to jointly update the search individual position $\vec{X}$. The formula for updating the position of slime mould is as Eq. (6).

$$\overrightarrow{X(t+1)} = \overrightarrow{X_{\text{good}b}(t)} + \overrightarrow{vb} \cdot (\vec{W} \cdot \overrightarrow{X_{\text{good}A}(t)} - \overrightarrow{X_{\text{good}B}(t)}) \tag{6}$$

where $\overrightarrow{X_{\text{good}b}}$ is the best solution for the fitness value in the dominant swarm, $\overrightarrow{X_{\text{good}A}}$ and $\overrightarrow{X_{\text{good}B}}$ are two randomly selected position vectors from the dominant swarm, $\overrightarrow{vb}$ is the random number vector with the value in $[-a, a]$, a is calculated by Eq. (3), $\vec{W}$ represents the adaptive weight of the slime mould individual.

SMA sorts the individual fitness value in each iteration in order to find the optimal and the worst fitness. The sorting process is time-consuming, and to make better use of the sorted individual positions and fitness values, DTSMA divides the sorted population into two subpopulations, $\overrightarrow{X_{\text{good}A}}$ from the population ranked in the top half of fitness values and $\overrightarrow{X_{\text{good}B}}$ from the other population. The values of A and B are taken as Eq. (7) and Eq. (8).

$$A = round\left(\frac{N}{2} \cdot rand\right) \tag{7}$$

$$B = round\left(\frac{N}{2} + \frac{N}{2} \cdot rand\right) \tag{8}$$

where N denotes the population size, $rand$ denotes a random number in [0,1], $round$ indicates the rounding function.

After adding the dominant swarm, the convergence speed and solution accuracy of SMA have been greatly improved, but the problem of easily falling into local optimum is still severe.

3.2. Mutation mechanism

SMA has strong exploitation ability, but weak exploration ability. The algorithm is easy to fall into local optimum and appear premature convergence phenomenon. To balance exploration and exploitation, mutation mechanism is added after the regeneration of dominant swarm. There are many probabilistic mutation mechanisms, such as Levy flight [57,58], Gaussian mutation [49,59,60] and Cauchy mutation [61], all of which can enhance the search ability of the algorithm. Levy flight can enhance the exploration and exploitation ability of the algorithm at the same time, but mainly enhance the exploitation ability. SMA needs to improve the exploration ability, so it is not suitable to use Levy flight mechanism. For algorithms with strong exploitation ability, Gaussian mutation can enhance its exploration ability, while for algorithms with strong exploration ability, Gaussian mutation can enhance its exploitation ability. In literature [49], SMA based on Gaussian mutation is used to solve the 0-1 knapsack problem. Since knapsack problem is NP hard discrete optimization problem, it is necessary to improve the exploration ability of SMA. But for more general optimization problems, the later exploitation ability of the algorithm needs to be concerned. Cauchy mutation also enhanced SMA's exploration ability, but not as much as Gaussian mutation. Therefore, inspired by the above literature, this paper applies the t-distribution mutation switching between Gaussian mutation and Cauchy mutation to SMA. The degree of freedom of t-distribution mutation adaptively changes with the number of iterations, which can well balance the exploration and exploitation of SMA. When the degree of freedom is large, the t-distribution is close to the Gaussian distribution, and when the degree of freedom is equal to 1, it is the Cauchy distribution, as clearly shown in Eq. (9) and Figure 2.

$$trnd(tn) = \begin{cases} Norm(0,1) & tn \to \infty \\ Cauchy(0,1) & tn = 1 \end{cases} \tag{9}$$

where $trnd(tn)$ denotes the t-distribution with degrees of freedom tn.

In DTSMA, the position of each slime mould of the dominant swarm $\overrightarrow{X_{\text{good}}}$ is perturbed using t-distribution mutation with adaptive parameters. t-distribution mutation operator is mathematically formulated as Eq. (10).

$$\overrightarrow{TX} = \overrightarrow{X_{\text{good}}} + \overrightarrow{X_{\text{good}}} \cdot trnd(tn) \tag{10}$$

where $\overrightarrow{TX}$ denotes the position vector of slime mould after t-distribution mutation, and tn denotes the degree of freedom parameter of the t-distribution.

In DTSMA, the degree of freedom parameter tn grows nonlinearly with the number of iterations t. The value of tn is calculated as Eq. (11).

$$tn = \exp\left(4 \cdot \left(t/\max\_t\right)^2\right) \tag{11}$$

The degree of freedom parameter tn enables DTSMA to approximate the use of the Cauchy mutation in the early iteration to enhance the exploration ability, and to approximate the use of the Gaussian mutation in the late iteration to focus on the exploitation ability. During the iteration of DTSMA, with the increase of the degree of freedom tn, the algorithm gradually transforms from focusing on the global exploration ability to the local exploitation ability. The t-distribution mutational operator combines the advantages of Gaussian mutational and Cauchy mutational operators, allowing DTSMA to achieve an excellent balance between exploration and exploitation.

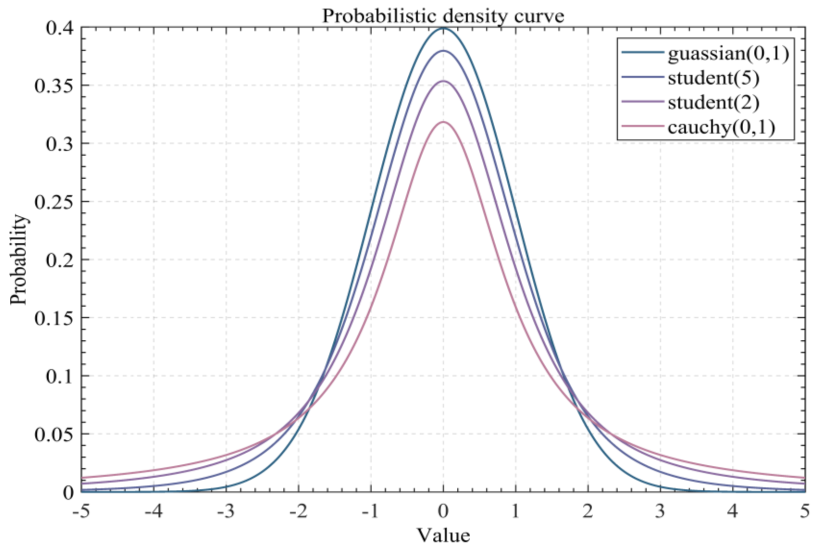

Figure 2. Probability density curves for Gaussian, Cauchy, and T-distribution.

3.3. Greedy strategy

SMA does not use greedy selection, and a greedy strategy is utilized in DTSMA to retain search agents of slime mould with better fitness than the current ones and eliminate those with worse fitness in each iteration, expressed in the mathematical formula as Eq. (12).

$$\overrightarrow{X_{\text{good}}(t+1)} = \begin{cases} \overrightarrow{X(t+1)} & S(\overrightarrow{X(t+1)}) < S(\overrightarrow{X_{\text{good}}(t)}) \\ \overrightarrow{X_{\text{good}}(t)} & \textit{others} \end{cases} \tag{12}$$

where $S(\vec{X})$ denotes the fitness of $\vec{X}$, $\overrightarrow{X_{\text{good}}}$ represents the position in the dominant swarm.

The use of a greedy strategy seems to weaken the exploration performance of the algorithm, but the mutation mechanism incorporated in each iteration of DTSMA constantly performs exploration, and greedy selection simply discards the fraction of individuals that fail in exploration and prepares them more adequately for the next exploration.

3.4. Exploitation operator

Finally, a search operator was added in the exploitation phase of DTSMA to increase the population diversity of slime mould, and the exploitation operator was formulated as Eq. (13).

$$\overrightarrow{X(t+1)} = \overrightarrow{X_{\text{good}}(t)} + \overrightarrow{vc} \cdot \overrightarrow{X_{\text{good}}(t)} \tag{13}$$

where $\overrightarrow{X_{\text{good}}}$ represents the position in the dominant swarm, $\overrightarrow{vc}$ is a random number vector with the value in $[-b, b]$, and b decreases linearly from 1 to 0 with the number of iterations.

This operator donates that the search agents of the slime mould will eventually stop at the optimal position it currently finds, and in some cases, the individual optimal may converge beyond the current global optimal position $\overrightarrow{X_{\text{good}b}}$. Based on the above principles, the mathematical formula for the position update can be organized as Eq. (14).

$$\overrightarrow{X(t+1)} = \begin{cases} rand \cdot (UB - LB) + LB & rand < z \\ \overrightarrow{X_{\text{good}b}(t)} + \overrightarrow{vb} \cdot (\overrightarrow{W} \cdot \overrightarrow{X_{\text{good}A}(t)} - \overrightarrow{X_{\text{good}B}(t)}) & r < p \\ \overrightarrow{vc} \cdot \overrightarrow{X_{\text{good}}(t)} & r \geq p \text{ and } r < q \\ \overrightarrow{X_{\text{good}}(t)} + \overrightarrow{vc} \cdot \overrightarrow{X_{\text{good}}(t)} & others \end{cases} \tag{14}$$

where q is a parameter that can be adjusted to the specific problem and takes values in [0,1].

3.5. Sensitivity analysis of parameters

When using DTSMA, it is necessary to determine two adjustable parameters z and q, among which the adjustment method of parameter z is consistent with that of SMA, which can be referred to [29]. To illustrate the impact of q on solving optimization problems and to facilitate users to adjust on specific problems, the value of q was compared on the CEC2019 functions, and the interval between 0 and 1 is 0.1. The test results are shown in Table 1. The data presented in the table are the average optimal fitness obtained by the algorithm running 30 times on each function and their rank among the other values taken by q. As can be seen from Table 1, the Friedman mean rank best when q is 0.9 and obtained the best results on the five functions. It shows that the searching ability of DTSMA is improved significantly when q is taken as 0.9. Therefore, considering the generalization ability of the DTSMA algorithm, q is taken as 0.9 for the next test. In addition, for most optimization problems, the value of q should be taken in [0.7,0.9].

Table 1. Comparison of parameter *q* of DTSMA on CEC2019 functions.

Functions	*q*										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
F1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F2	4.2512	4.2500	**4.2475**	4.2559	4.2751	4.2583	4.3347	4.3351	4.3725	4.5199	4.5303
F3	2.6640	2.5785	2.5607	3.0614	2.3024	2.0508	2.3376	2.3145	2.0807	**1.9740**	2.6250
F4	13.734	**13.502**	14.127	14.663	13.933	14.331	14.398	14.265	14.133	14.100	14.663
F5	1.3057	1.2905	1.2724	1.2592	1.2849	1.2519	1.2704	1.2395	1.2635	**1.2385**	1.3039
F6	2.5919	2.6585	2.6508	2.8353	2.6663	2.6223	2.6432	2.6485	2.4341	**2.3836**	2.6009
F7	606.12	589.35	600.74	606.12	606.12	617.14	588.10	580.54	577.53	**576.87**	579.22
F8	3.4369	3.5183	3.5882	3.2536	3.2650	3.4407	3.4934	3.3356	3.4767	**3.2166**	3.4208
F9	1.1659	1.1824	1.1650	1.1700	1.1550	1.1569	1.1706	1.1651	1.1734	1.1460	**1.1388**
F10	19.041	20.556	19.545	19.937	20.647	20.127	21.235	20.052	18.799	18.835	**18.718**
Mean rank	5.64	6.45	5.64	6.55	5.55	5.36	6.73	5.09	4.82	**2.73**	5.45
Ranking	7.5	9	7.5	10	6	4	11	3	2	1	5

The optimal values are shown in bold.

The pseudo-code of DTSMA is presented in Algorithm 2, and the flow chart is shown in Figure 3.

Algorithm 2 Pseudo-code of DTSMA

```
1.  Initialize the parameters $z, q, n, d, max\_t$;
2.  Initialize the positions of slime mould $\overrightarrow{X_i}(i = 1, 2, ..., n)$;
3.  While ($t \le max\_t$)
4.      Calculation the fitness $\vec{S}$ of all slime mould;
5.      Update $\overrightarrow{X_{\text{good}}}, \overrightarrow{S_{\text{good}}}$ by Eq. (12);
6.      Perturbation $\overrightarrow{X_{\text{good}}}$ by Eq. (10);
7.      Update $\overrightarrow{X_{\text{good}}}, \overrightarrow{S_{\text{good}}}$ by Eq. (12);
8.      Sort the fitness $\overrightarrow{S_{\text{good}}}$;
9.      Update $bF, wF, DF, \overrightarrow{X_{\text{good}b}}$;
10.     Calculate the $\vec{W}$ by Eq. (4);
11.     Update $p, \vec{vb}, \vec{vc}, A, B$;
12.     For each search agents
13.         Update positions by Eq. (14);
14.     End For
15.     $t = t + 1$;
16. End While
17. Return $DF, \overrightarrow{X_{\text{good}b}}$;
```

3.6. Computational complexity analysis

DTSMA mainly consists of the subsequent components: initialization, fitness evaluation, dominant swarm update, t-distribution mutation, sorting, weight update, and location update. Among them, N donates the number of agents of slime mould, Dim donates the dimension of the variable, and $\max\_t$ donates the maximum number of iterations. The computation complexity of initialization is $O(N * Dim)$, the computation complexity of dominant swarm update and t-distribution mutation are $O(N)$, the computation complexity of sorting is $O(N * \log N)$, the computation complexity of weight update is $O(N * Dim)$, the computation complexity of location update is $O(N * Dim)$. Therefore, assuming that the time complexity of fitness evaluation is $O(F)$, the total computation complexity is $O\left(\max\_t * \left(N * Dim + N * \log N + F\right)\right)$, which is the same as SMA. The space complexity of DTSMA is $O(N * Dim)$.

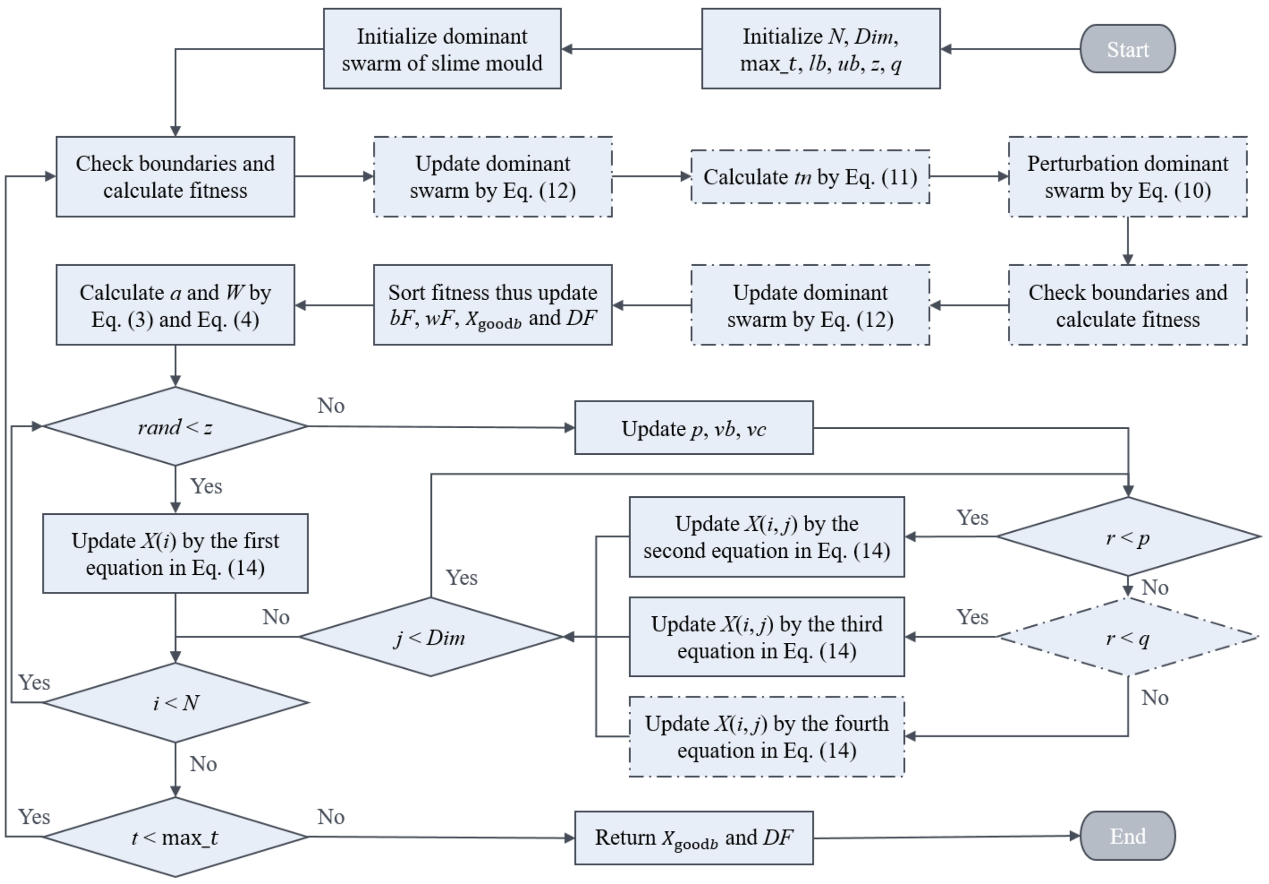
Start
Initialize N, Dim, max_t, lb, ub, z, q
Initialize dominant swarm of slime mould
Check boundaries and calculate fitness
Update dominant swarm by Eq. (12)
Calculate tn by Eq. (11)
Perturbation dominant swarm by Eq. (10)
Check boundaries and calculate fitness
Update dominant swarm by Eq. (12)
Sort fitness thus update bF, wF, X_{goodb} and DF
Calculate a and W by Eq. (3) and Eq. (4)
$rand < z$
No
Update p, vb, vc
Yes
Update $X(i)$ by the first equation in Eq. (14)
$r < p$
Yes
Update $X(i, j)$ by the second equation in Eq. (14)
No
$r < q$
Yes
Update $X(i, j)$ by the third equation in Eq. (14)
No
Update $X(i, j)$ by the fourth equation in Eq. (14)
$j < Dim$
Yes
No
$i < N$
Yes
No
$t <$ max_t
Yes
No
Return X_{goodb} and DF
End

Figure 3. Flow chart of the DTSMA.

4. Experimental results on benchmark functions

To verify the improvement, the performance of DTSMA was evaluated using the average best fitness value and its standard deviation, the results of the test functions were ranked, and the Friedman rank of each algorithm on the different test functions was counted. Then, the Wilcoxon rank-sum test was used to evaluate the differences between DTSMA and comparison algorithms. For a fair comparison, all algorithms were set with the same common parameters, the population size to 30, and the maximum number of iterations to 1000. All experiments were executed on Windows 10 OS and all algorithm codes were run in MATLAB R2019a with hardware details: Intel(R) Core (TM) i7-9700 CPU (3.00GHz) and 16GB RAM.

4.1. Benchmark functions

In this study, the test functions for the DTSMA comparison experiment are the CEC2019 functions. The search ranges and minimum values are shown in Table 2, and the 3-D map for 2-D function are shown in Figure 4.

Table 2. Characteristics of CEC2019 benchmark functions.

Functions	Dim	Range	Optimal
F1: Storn's Chebyshev Polynomial Fitting Problem	9	[-8192,8192]	1
F2: Inverse Hilbert Matrix	16	[-16384,16384]	1
F3: Lennard-Jones Minimum Energy Cluster	18	[-4,4]	1
F4: Rastrigin's Function	10	[-100,100]	1
F5: Griewank's Function	10	[-100,100]	1
F6: Weierstrass Function	10	[-100,100]	1
F7: Modified Schwefel's Function	10	[-100,100]	1
F8: Expanded Schaffer's F6 Function	10	[-100,100]	1
F9: Happy Cat Function	10	[-100,100]	1
F10: Ackley Function	10	[-100,100]	1

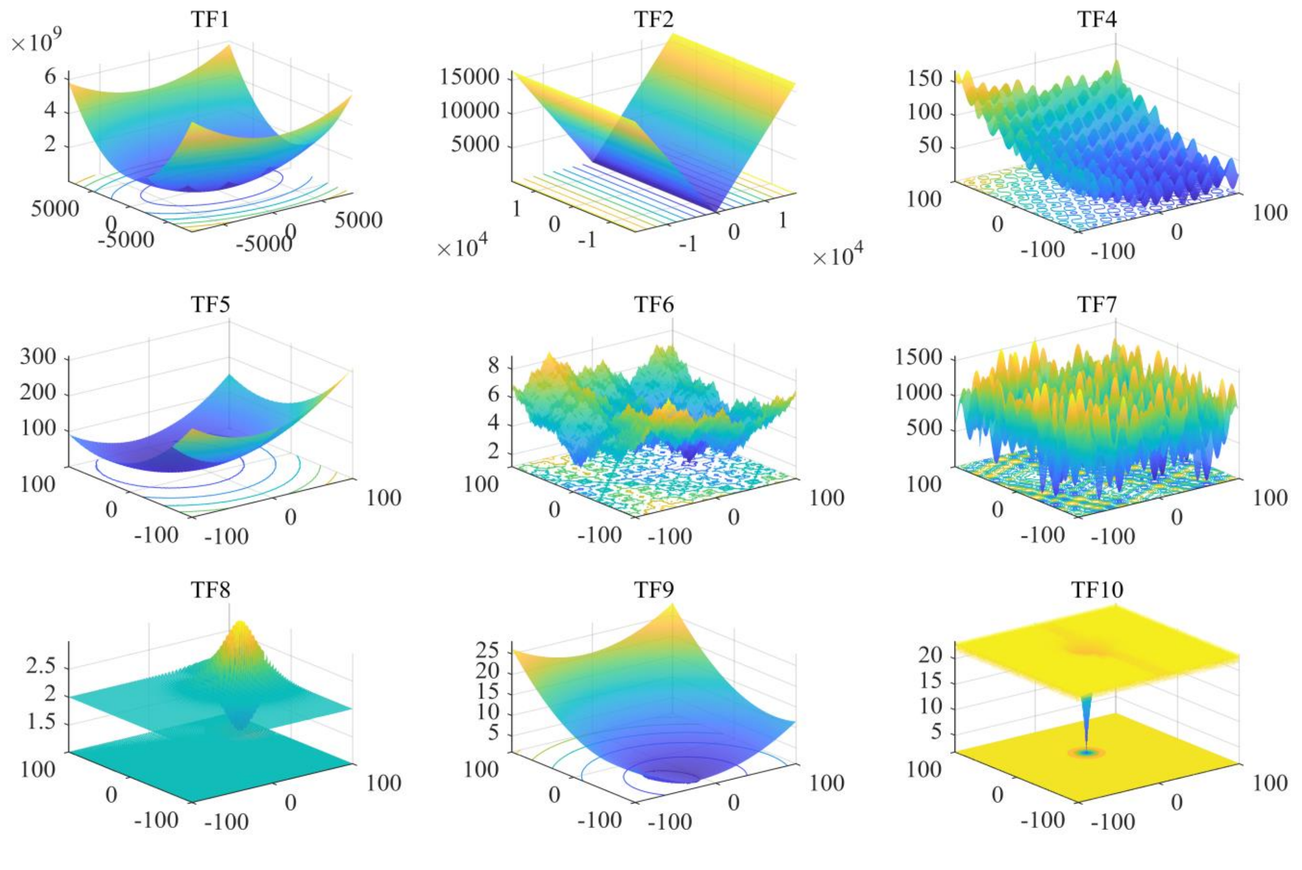

Figure 4. Two-dimensional perspective view of CEC2019 benchmark functions.

4.2. Comparison algorithm parameter setting

To test the effectiveness and efficiency, DTSMA was compared with twenty-two algorithms, including the original SMA [29], classical algorithms (i.e., PSO [18], DE [8], TLBO [20], GWO [21], WOA [22], SSA [23], MVO [14], MFO [62], ALO [63], DA [64], SCA [65]), novel algorithms (i.e., Equilibrium Optimizer (EO) [17], Bald Eagle Search (BES) [28], Harris Hawks Optimization (HHO) [27], Pathfinder Algorithm (PFA) [66], Seagull Optimization Algorithm (SOA) [25]), improved algorithms (i.e., Autonomous Groups Particle Swarm Optimization (AGPSO) [67], Gaussian Quantum-behaved Particle Swarm Optimization (GQPSO) [68], hybrid Particle Swarm Optimization and Gravitational Search Algorithm (PSOGSA) [69], Centroid Opposition-based Differential Evolution (CODE) [70]), and superior performance algorithms (i.e., Multi-trial Vector-based Differential Evolution (MTDE) [71]). The adjustable parameter settings of comparison algorithms are shown in Table 3.

Table 3. Parameter settings of the optimization algorithms.

Algorithms	Parameters	Values	Algorithms	Parameters	Values
DTSMA	Constant z	0.03	TLBO	Teaching factor TF	{1, 2}
	Constant q	0.9	PFA	Parameter less	NA
SMA	Constant z	0.03	PSO	Inertia weight w	1
HHO	Constant β	1.5		Cognitive coefficient c_1	2
GWO	Convergence factor a	[2, 0]		Social coefficient c_2	2
WOA	Convergence factor a	[2, 0]		Maximum velocity v	6
	Logarithmic spiral b	1	AGPSO	Inertia weight w	[0.9, 0.4]
	Random number l	[-1, 1]		Cognitive coefficient c_1	2
MVO	Wormhole existence probability	[0.2, 1]		Social coefficient c_2	2
	Traveling distance rate TDR	[0.6, 1]	GQPSO	Inertia weight w	[1, 0.5]
MFO	Convergence factor a	[-1, -2]		Cognitive coefficient c_1	1.5
	Logarithmic spiral b	1		Social coefficient c_2	1.5
	Random number t	[-1, 1]	PSOGSA	Inertia weight w	[1, 0]
ALO	Parameter less	NA		Cognitive coefficient c_1	0.5
DA	Convergence factor w	[0.9, 0.4]		Social coefficient c_2	1.5
	Constant s	0.1		Gravitational constant G_0	1
	Constant a	0.1		Constant α	23
	Constant c	0.7	DE	Mutation factor F	0.5
SCA	Constant a	2		Crossover rate Cr	0.9
SOA	Convergence factor f_c	[2, 0]	CODE	Mutation factor F	0.5
SSA	Convergence factor c_1	[2, 0]		Crossover rate Cr	0.9
	Random number c_2	[0, 1]		Generation jumping rate Jr	0.3
	Random number c_3	[0, 1]	MTDE	Constant $WinIter$	20
EO	Control volume V	1		Constant H	5
	Generation probability GP	0.5		Constant $initial$	0.001
	Constant a_1	2		Constant $final$	2
	Constant a_2	1		Parameter Mu	log(Dim)
BES	Constant α	2		Constant μf	0.5
	Spiral parameter a	10		Constant σ	0.2
	Spiral parameter R	1.5			

For all algorithms, N=30, Max_t=1000.

4.3. Experimental results and analysis

The results were reported in Table 4 and Table 5, where Table 4 exhibits the average best fitness obtained by running the algorithm for 30 times, and Table 5 exhibits the standard deviation of the 30 best fitness values. As can be seen from Table 4, DTSMA achieves the best results on F1-2 and F10, and is significantly superior to other comparison algorithms in terms of convergence accuracy. In addition, MTDE obtained the best solution on F5-7, EO showed a clear advantage on F8-9, and PFA performed best on F3. But in general, DTSMA ranks first in average performance among 23 comparison algorithms, and can obtain better solutions, and is far better than SMA, which indicates that the performance of proposed DTSMA is significant.

Table 4. Comparison of the fitness values of the optimized results on the CEC2019 functions.

Algorithms	Functions										Mean	Rank
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	rank	
DTSMA	**1.00E+00**	**4.52E+00**	1.9740	14.1003	1.2385	2.3836	5.77E+02	3.2166	1.1460	**18.8351**	**1.78**	**1**
SMA	1.00E+00	4.98E+00	4.3983	15.5960	1.2864	4.2854	7.06E+02	3.7392	1.2237	20.4424	3.57	8
HHO	1.00E+00	4.99E+00	4.4340	47.2977	1.9456	7.1446	1.21E+03	4.8260	1.4232	21.1185	6.30	15
GWO	2.21E+04	3.63E+02	2.3769	16.1127	1.6048	2.6925	8.18E+02	3.6554	1.1865	21.4486	4.61	10
WOA	8.20E+06	6.70E+03	4.6371	57.7068	2.1331	8.9500	1.38E+03	4.6022	1.3726	20.8007	8.00	20
MVO	1.34E+06	4.68E+02	7.6377	19.5270	1.2984	3.2522	7.38E+02	3.8942	1.2143	21.0478	5.13	12
MFO	1.35E+07	1.21E+03	7.4254	28.2234	2.3360	5.0370	1.04E+03	4.3095	1.3662	21.1609	7.09	17
ALO	1.72E+06	1.88E+03	3.4462	26.6147	1.2077	5.0212	1.11E+03	4.3382	1.3086	20.3639	5.52	14
DA	1.80E+07	5.43E+03	9.6797	54.6155	2.4107	7.3037	1.32E+03	4.5528	1.3616	21.3550	8.74	22
SCA	1.77E+06	3.06E+03	9.2544	45.4784	8.3053	7.6132	1.46E+03	4.4275	1.5651	21.4495	8.91	23
SOA	3.89E+03	1.48E+02	9.2135	28.4609	3.5890	7.3025	1.05E+03	4.3820	1.3520	21.4066	6.87	16
SSA	1.71E+06	9.87E+02	3.6506	25.6377	1.2731	4.0937	9.10E+02	4.1329	1.3158	21.0355	5.35	13
EO	2.46E+02	8.42E+01	1.6587	12.5181	1.0457	1.4470	5.84E+02	**3.2097**	**1.0714**	21.2241	2.17	3
BES	1.49E+00	7.10E+00	3.4173	14.5653	1.2038	2.3826	8.28E+02	3.2594	1.1128	20.3191	2.48	4
TLBO	1.13E+04	3.09E+02	1.7333	10.8442	1.0972	1.9704	6.96E+02	3.4395	1.1487	20.8363	2.61	5
PFA	1.61E+05	6.37E+02	**1.5436**	27.9839	1.2161	5.0069	1.02E+03	3.9178	1.2428	21.1618	4.96	11
PSO	8.00E+07	1.82E+04	9.4677	42.7568	1.9978	4.8771	1.14E+03	4.0256	1.2283	21.4468	7.65	19
AGPSO	1.91E+05	3.94E+02	3.9437	16.3787	1.4541	2.6212	6.65E+02	3.6565	1.2064	21.0567	4.35	9
GQPSO	1.00E+00	4.93E+00	6.6126	60.9991	27.3784	7.6309	1.66E+03	4.6306	1.7226	21.2928	7.57	18
PSOGSA	1.47E+07	2.87E+03	6.7206	51.0280	5.9787	6.1109	1.15E+03	4.8314	1.5481	21.0608	8.17	21
DE	4.59E+04	1.52E+02	3.6217	9.0607	1.0330	1.5816	4.58E+02	3.5361	1.1434	21.3024	3.00	6
CODE	4.02E+05	7.12E+02	4.2384	**5.3583**	1.1457	1.4039	2.95E+02	4.3878	1.1045	19.3490	3.13	7
MTDE	1.00E+00	8.34E+01	2.1171	6.7722	**1.0126**	**1.1680**	**1.03E+02**	3.2895	1.1538	21.1915	2.04	2

The optimal values are shown in bold.

It can be summarized from Table 5 that the stability of MTDE is better than DTSMA on the CEC2019 functions, and it is also inferior to EO and GQPSO in terms of robustness, but the robustness of DTSMA is much better than the original SMA. Therefore, the proposed DTSMA is superior to SMA in convergence accuracy and robustness, which verifies the effectiveness and efficiency of DTSMA. In conclusion, the Friedman mean rank shows DTSMA as a powerful optimization algorithm with good performance not only in the search ability of the optimal solution but also in most functions, which is very competitive with MTDE and EO. Therefore, DTSMA can provide a high-level candidate solution for complex function optimization problems with strong generalization ability.

Table 5. Comparison of the standard deviation of the fitness values of the optimized results.

Algorithm	Functions										Mean	Rank
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	rank	
DTSMA	**0.00E+00**	3.72E-01	1.3323	6.6519	0.1015	1.3082	272.906	0.4676	0.0460	5.8242	3.91	4
SMA	0.00E+00	9.75E-02	2.4525	7.9251	0.1117	1.6918	239.462	0.4720	0.0877	3.6532	5.22	13.5
HHO	0.00E+00	**3.46E-02**	1.3755	17.7759	0.2214	1.6247	378.685	0.2044	0.1461	0.1002	5.00	10
GWO	5.68E+04	2.58E+02	1.1658	7.2606	0.4893	1.0755	328.711	0.4717	0.0894	0.1019	5.13	12
WOA	9.31E+06	3.59E+03	1.6751	19.4379	0.4902	1.6476	348.194	0.3436	0.2181	2.2460	7.65	21
MVO	1.23E+06	1.25E+02	2.1286	7.9015	0.1325	1.3980	291.478	0.5071	0.0850	**0.0483**	5.26	15
MFO	1.19E+07	1.90E+03	2.5007	12.2410	3.1002	2.0649	267.218	0.4505	0.1932	0.1549	7.78	22
ALO	1.79E+06	1.18E+03	1.7404	10.9068	0.0970	1.7245	300.774	0.3597	0.1399	3.6584	6.70	19
DA	1.64E+07	3.12E+03	1.2699	22.4050	2.3606	1.8155	348.975	0.3223	0.1403	0.1316	7.43	20
SCA	3.09E+06	1.17E+03	1.5328	7.2385	3.5911	1.3884	212.128	0.1744	0.1134	0.0838	5.09	11
SOA	2.09E+04	2.08E+02	1.4258	10.5959	1.2344	1.2177	298.096	0.3580	0.1012	0.1173	5.22	13.5
SSA	1.83E+06	6.19E+02	2.0938	9.4445	0.1589	1.3869	253.190	0.4795	0.1403	0.0747	5.83	18
EO	9.02E+02	9.89E+01	0.6993	4.6354	0.0278	0.6651	262.494	0.5480	0.0347	0.1044	3.04	2
BES	2.62E+00	7.76E+00	1.4614	7.0637	0.1427	1.1426	297.048	0.5116	0.0359	3.8863	4.74	8
TLBO	2.57E+04	1.33E+02	0.4779	3.9135	0.0620	0.9021	346.690	0.4583	0.0625	2.9572	4.09	5
PFA	2.64E+05	7.46E+02	**0.4472**	10.8316	0.1061	1.6349	276.130	0.3301	0.0868	1.3630	4.91	9
PSO	5.00E+07	6.79E+03	0.8635	9.2252	0.0900	1.6462	307.838	0.4587	0.0877	0.0750	5.78	17
AGPSO	2.54E+05	1.10E+02	2.2718	7.1966	1.6605	1.4331	267.269	0.4819	0.0838	0.0909	5.43	16
GQPSO	1.12E-08	1.03E-01	0.7154	6.7416	4.3359	**0.2477**	166.489	**0.1493**	0.1034	0.3075	3.22	3
PSOGSA	2.53E+07	3.02E+03	3.1301	22.3074	10.7717	1.7938	335.219	0.3501	0.3823	0.1051	8.26	23
DE	9.02E+04	8.17E+01	2.2019	4.7660	0.0234	0.7731	331.394	0.4103	0.0588	0.1179	4.17	6
CODE	4.73E+05	2.08E+02	2.0383	**1.7357**	0.1720	0.5976	236.436	0.4352	**0.0340**	6.1610	4.35	7
MTDE	7.09E-03	5.58E+01	1.2621	2.4686	**0.0138**	0.4680	**126.141**	0.4102	0.0479	0.0543	**1.78**	**1**

The optimal values are shown in bold.

The convergence curves of algorithms on CEC2019 functions are given in Figure 5 and Figure 6. The results show that DTSMA outperforms most of the compared algorithms, especially the classical metaheuristic, in terms of convergence speed and solution accuracy. In Figure 5, DTSMA achieves the best performance on all tested functions. In Figure 6, DTSMA achieves optimal performance on F1–2 and F10, and is less competitive on F4–7 and F9, especially on F7, where MTDE shows its superiority. Because F7 has many locally optimal solutions, making the algorithm easily fall into local optima and premature convergence, which indicates that MTDE outperforms DTSMA in terms of exploration ability. On F1–2, DTSMA still has the fastest convergence speed and best solution accuracy, which indicates that DTSMA is obviously superior to MTDE in terms of exploitation ability. Therefore, DTSMA and MTDE can be considered as complementary algorithms, which can be applied to different real-world optimization problems to obtain more satisfactory results.

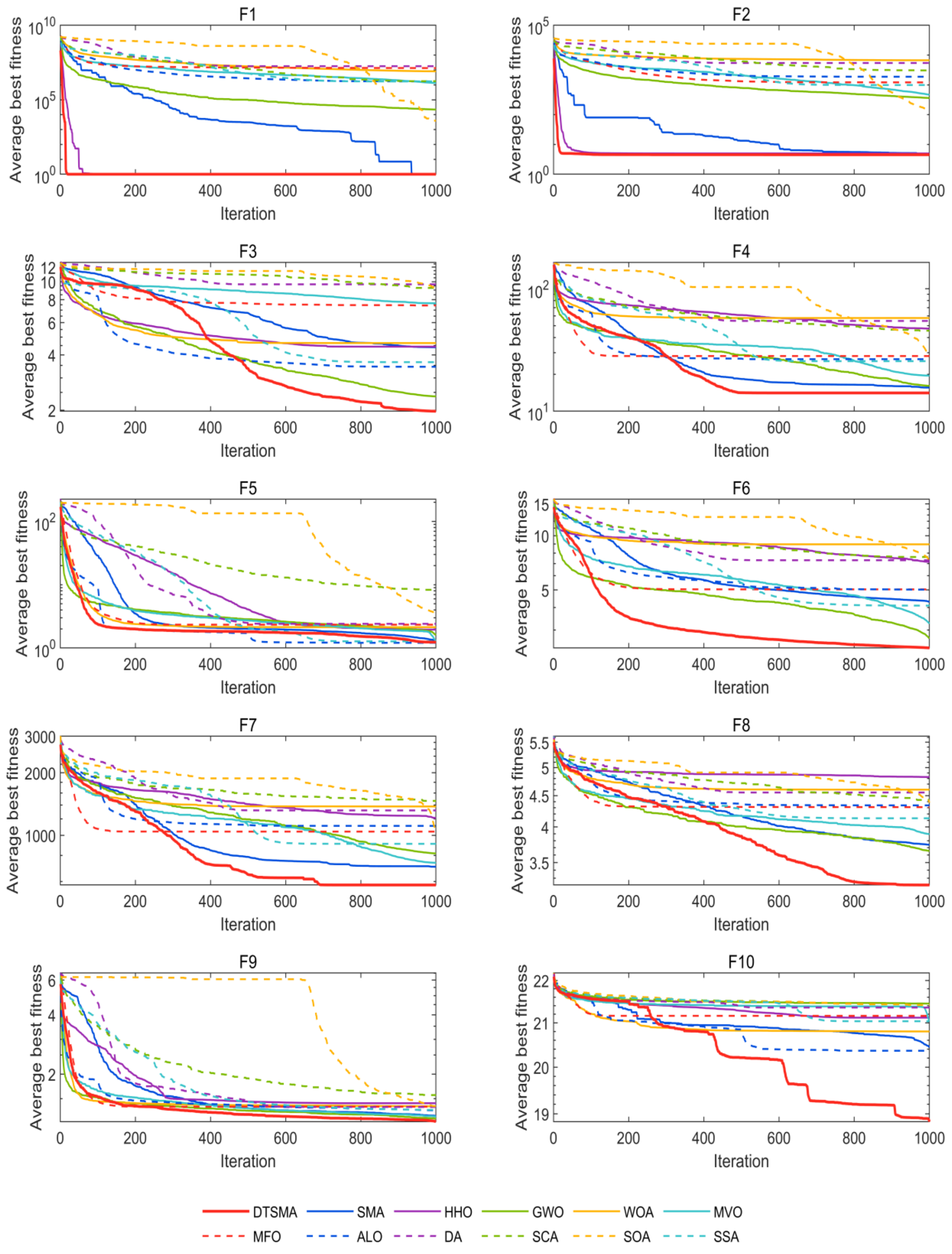

Figure 5. Convergence curve of classical algorithms on the CEC2019 functions.

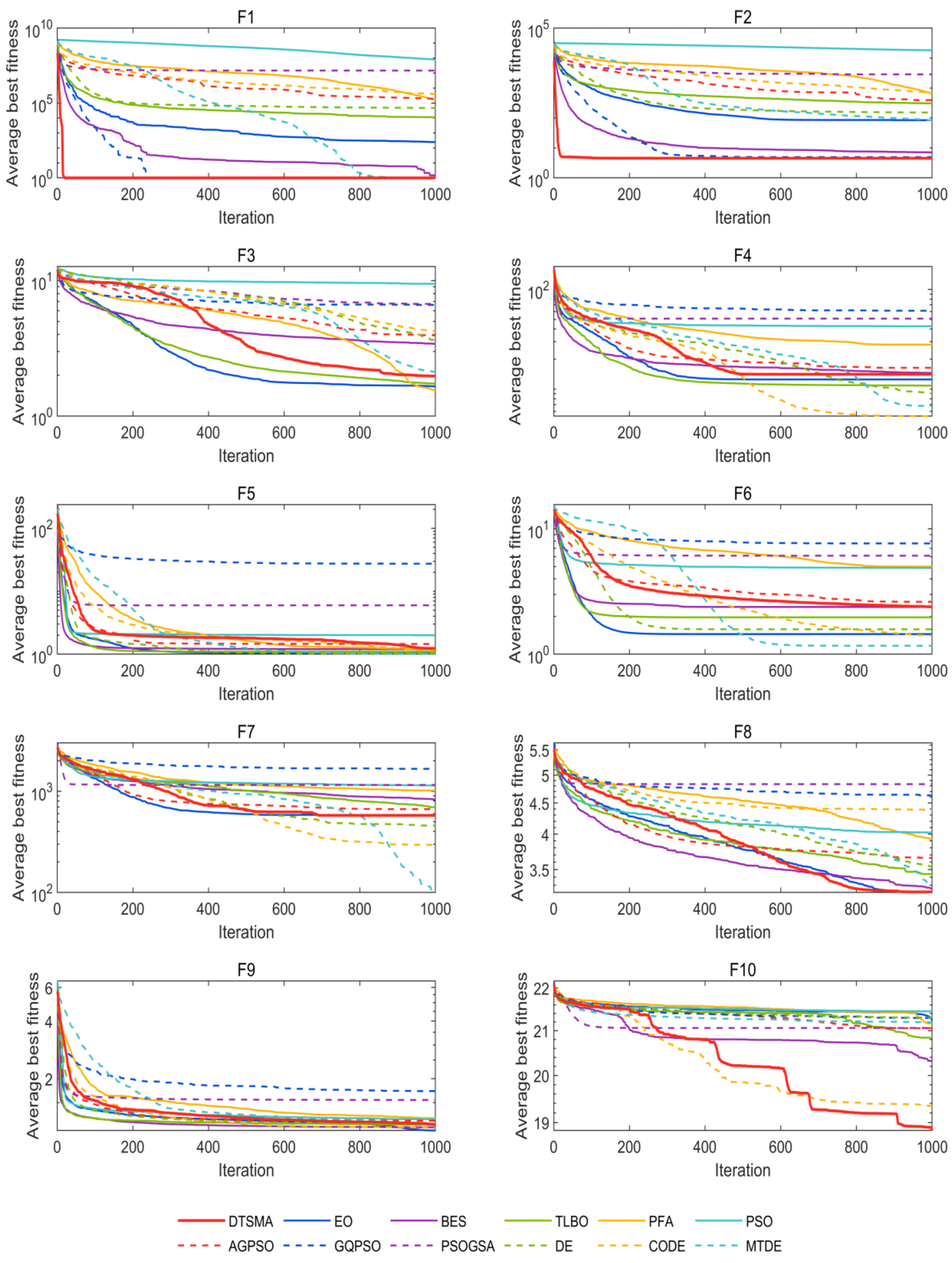

Figure 6. Convergence curves of advanced algorithms on the CEC2019 functions.

Since boxplots illustrate the data distribution, they are excellent graphs for describing the consistency between data. To further compare the distribution states of the optimization results of DTSMA and other algorithms, the best fitness values obtained by 23 algorithms run 30 times independently on each test function are presented in the form of box plots in Figure 7. The results show that DTSMA has the smallest median, upper quartile and lower quartile, the fewest outliers, and the narrowest distribution frame in the comparison of classical algorithms.

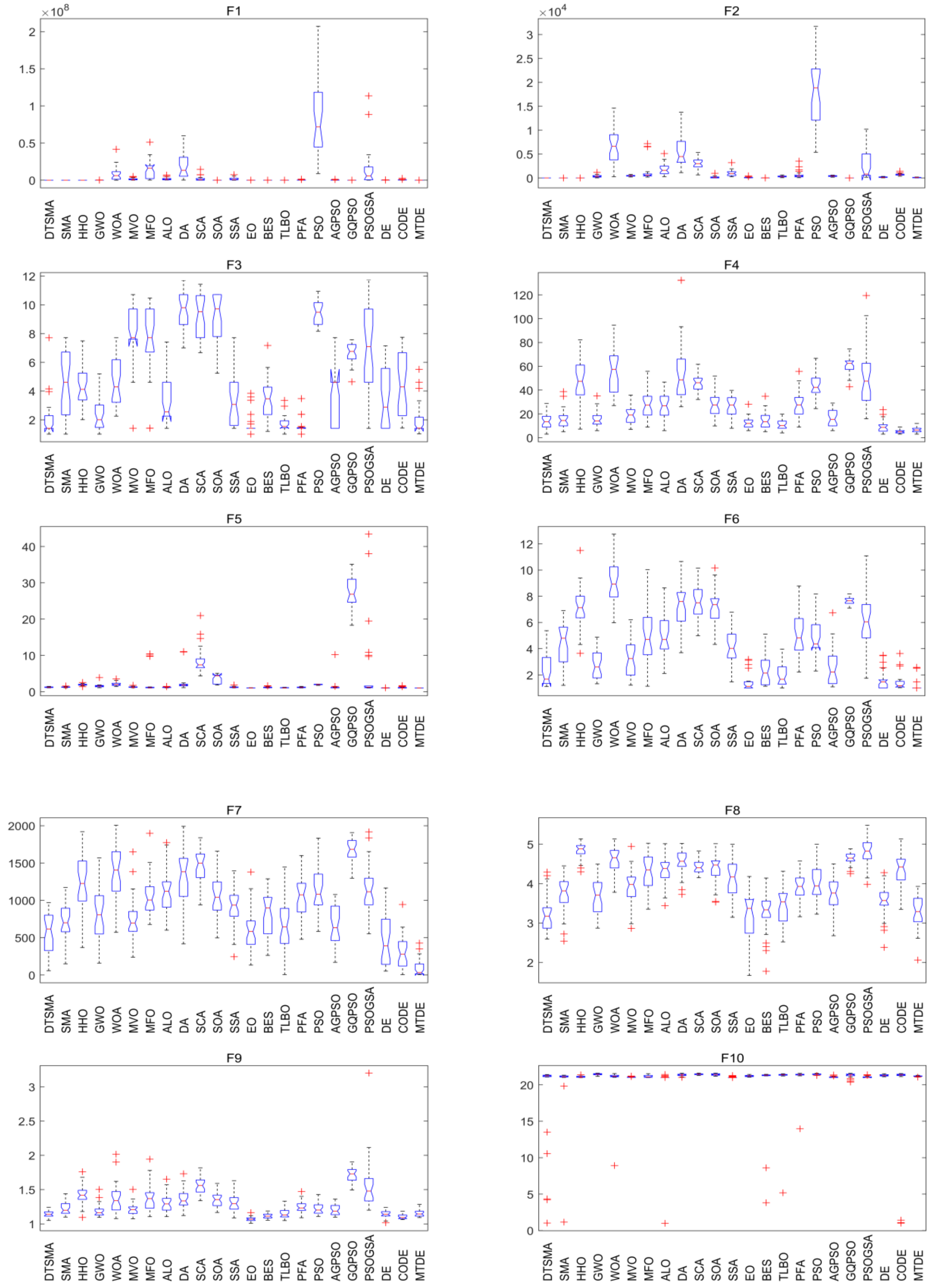

Figure 7. Comparison results of algorithms executed 30 times on CEC2019 functions.

In the comparison of advanced algorithms, DTSMA outperforms most algorithms and has strong robustness. In general, the performance of DTSMA and MTDE is the best, and the two algorithms have their own advantages for different functions respectively, which are far better than the other algorithms. Therefore, DTSMA is a good optimization algorithm in the terms of convergence accuracy and robustness.

The Wilcoxon rank-sum test [72] is used to verify whether there is a significant difference between the two data sets, i.e., the test evaluates whether the obtained performance is not random. Due to the random nature of the metaheuristic algorithm, a similar comparison of statistical experiments is necessary to ensure the validity of the data. The *p*-value is an indicator of decreasing confidence that there is a significant difference between the two data sets, the smaller the *p*-value, the higher the confidence level. When $p<0.05$, it indicates that there is a significant difference between the data considered for the two algorithms at a confidence interval of 95%. The results of the Wilcoxon *p*-value test of DTSMA and well-known algorithms are shown in Table 6.

The results of the Wilcoxon *p*-value test show that there are fewer cases (shown in bold) without significant differences and that DTSMA significantly outperforms the original SMA on six functions. DTSMA has a strong competitive performance with EO, BES and TLBO, demonstrating the algorithm's advantages on different functions for different optimization problems. In conclusion DTSMA is significantly different from and outperforms SMA, and the results are statistically meaningful, verifying that the performance of DTSMA is not random.

Table 6. Wilcoxon *p*-value test results (two-tailed).

Paired algorithms		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
DTSMA	SMA	**NA**	3.28E-05	1.02E-05	**4.55E-01**	**7.98E-02**	2.43E-05	**1.15E-01**	2.01E-04	2.13E-04	2.71E-02
	HHO	**NA**	1.78E-05	1.31E-08	1.29E-09	3.69E-11	8.99E-11	3.96E-08	3.02E-11	1.07E-09	8.31E-03
	GWO	1.21E-12	2.63E-11	2.62E-03	**2.23E-01**	1.49E-06	**8.50E-02**	6.38E-03	1.30E-03	**7.98E-02**	4.18E-09
	WOA	1.21E-12	2.63E-11	1.85E-08	3.69E-11	3.69E-11	3.02E-11	1.07E-09	1.33E-10	6.05E-07	**5.69E-01**
	MVO	1.21E-12	2.63E-11	5.07E-10	5.83E-03	4.68E-02	1.99E-02	**9.33E-02**	1.87E-05	2.39E-04	3.16E-05
	MFO	1.21E-12	2.63E-11	1.85E-08	3.57E-06	1.22E-02	1.61E-06	1.07E-07	3.82E-09	2.20E-07	**1.96E-01**
	ALO	1.21E-12	2.63E-11	4.71E-04	1.17E-05	**2.46E-01**	4.11E-07	4.31E-08	8.89E-10	5.19E-07	1.11E-04
	DA	1.21E-12	2.63E-11	4.08E-11	3.69E-11	7.12E-09	1.33E-10	2.23E-09	1.46E-10	3.50E-09	1.41E-04
	SCA	1.21E-12	2.63E-11	6.07E-11	3.02E-11	3.02E-11	3.34E-11	3.69E-11	7.39E-11	3.02E-11	1.29E-09
	SOA	1.21E-12	3.43E-09	3.34E-11	4.44E-07	3.02E-11	4.50E-11	8.20E-07	4.20E-10	1.96E-10	7.60E-07
	SSA	1.21E-12	2.63E-11	2.75E-03	1.25E-05	**6.00E-01**	1.53E-05	1.75E-05	7.69E-08	1.87E-07	8.15E-05
	EO	2.93E-05	6.55E-10	**8.42E-01**	**3.33E-01**	5.49E-11	2.25E-04	**8.30E-01**	**3.71E-01**	3.65E-08	**9.59E-01**
	BES	3.45E-07	2.47E-02	6.74E-06	**9.47E-01**	**6.79E-02**	**9.82E-01**	2.62E-03	**1.71E-01**	7.96E-03	1.95E-03
	TLBO	1.21E-12	2.63E-11	4.06E-02	**7.48E-02**	1.36E-07	**3.18E-01**	**2.46E-01**	3.78E-02	**7.62E-01**	3.09E-06
	PFA	1.21E-12	2.63E-11	**5.30E-01**	7.60E-07	**3.48E-01**	2.38E-07	1.39E-06	4.80E-07	4.74E-06	2.57E-07
	PSO	1.21E-12	2.63E-11	3.02E-11	4.50E-11	3.02E-11	5.19E-07	9.26E-09	2.03E-07	6.77E-05	8.89E-10
	AGPSO	1.21E-12	2.63E-11	2.27E-03	**2.58E-01**	1.77E-03	**6.52E-01**	**2.34E-01**	1.52E-03	6.97E-03	2.84E-04
	GQPSO	1.21E-12	**7.64E-02**	5.57E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.34E-11	3.02E-11	1.41E-04
	PSOGSA	1.21E-12	2.63E-11	1.87E-07	1.09E-10	**4.92E-01**	1.55E-09	5.53E-08	6.07E-11	1.61E-10	7.30E-04
	DE	1.21E-12	2.63E-11	1.78E-04	1.17E-03	3.34E-11	4.43E-03	**1.19E-01**	2.75E-03	**9.35E-01**	3.27E-02
	CODE	1.21E-12	2.63E-11	4.44E-07	4.69E-08	3.99E-04	4.94E-05	1.78E-04	1.41E-09	2.68E-04	9.03E-04
	MTDE	1.21E-12	2.63E-11	4.68E-02	5.46E-06	2.98E-11	1.56E-08	2.23E-09	**1.81E-01**	**6.00E-01**	**1.49E-01**

No significant differences are shown in bold.

5. Applicability of DTSMA for solving engineering problems

To test the generalization ability of DTSMA, the DTSMA was tested in eight well-known constrained engineering design problems, i.e., three-bar truss, cantilever beam, pressure vessel, tension compression spring, welded beam, speed reducer, multiple-disc clutch brake and car side impact design problem. The optimization results of SMA and DTSMA given in tables are the optimal results obtained from 30 independent runs of the algorithm with 1000 iterations with 30 individuals. These engineering design problems have various constraints and need to be optimized using constraint handling methods.

5.1. Constraint processing method

In constraint processing techniques, penalty functions are simple and easy to implement. There are different types of penalty functions, such as static, dynamic, annealing, and adaptive penalties, and these methods transform the constrained problem into an unconstrained one by adding a certain penalty value [73]. In this paper, a static penalty function was used to deal with the constraints of the engineering problem. The mathematical model of the penalty function is expressed as Eq. (15).

$$O(\vec{x}) = f(\vec{x}) + w \cdot \left(\sum_{i=1}^{m} \max\left(0, g_i(\vec{x})\right) + \sum_{i=1}^{n} \max\left(0, \left|h_i(\vec{x})\right| - \varepsilon\right) \right) \tag{15}$$

where $O(\vec{x})$ denotes the objective function, $f(\vec{x})$ denotes the objective function without considering the constraints, m and n denote the number of equation constraints and inequality constraints, respectively, $g_i(\vec{x})$ and $h_i(\vec{x})$ denote the inequality constraints and equation constraints, respectively, w denotes the penalty factor.

In this study, the penalty factor was set to 10^{15}. The array-indexed mapping approach was used to solve for discrete and integer variables.

5.2. Three-bar truss design problem

Three-bar truss design optimization is a non-linear fraction optimization [74]. This problem has only two decision parameters A_1 and A_2. The structure of the three-bar truss is presented in Figure 8. The mathematical formulation is defined as Eq. (16).

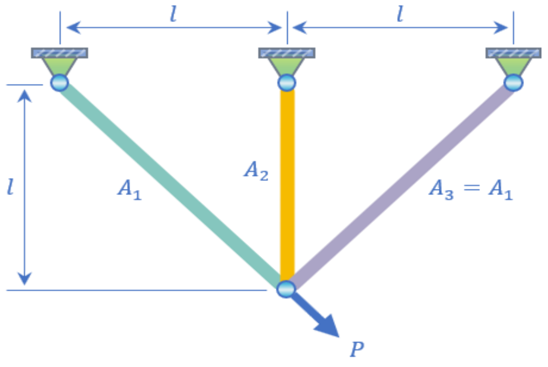

Figure 8. Three-bar truss structure and design variables.

$$\begin{aligned}
&\text{Consider } \vec{x} = [x_1, x_2] = [A_1, A_2] \\
&\text{Minimize } f(\vec{x}) = \left(2\sqrt{2}x_1 + x_2\right)\cdot l \\
&\text{subject to: } g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0 \\
&\qquad g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0 \\
&\qquad g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \le 0 \\
&\text{where } l = 100\text{cm};\ P = 2\text{KN/cm}^2;\ \ \sigma = 2\text{KN/cm}^2. \\
&\text{with } 0 \le x_1, x_2 \le 1.
\end{aligned} \tag{16}$$

Table 7 shows the optimal results obtained by DTSMA and other algorithms in the literature. It can be observed that DTSMA outperforms the original SMA and other comparative algorithms.

Table 7. Optimal results and comparison for the three-bar truss design problem.

Algorithms	A_1	A_2	Optimal weight
CS [75]	0.78867	0.40902	263.9716
AOA [76]	0.79369	0.39426	263.9154
MG-SCA [77]	1.00000	0.42715	263.8986
MGWO [78]	0.7885845	0.4085071	263.8961
IGWO [72]	0.78846	0.40884	263.8959
GOA [79]	0.788897555578973	0.407619570115153	263.895881496069
HHOSCA [80]	0.788498	0.40875	263.8958665
MBA [81]	0.7885650	0.4085597	263.8958522
SMA	0.794012414405404	0.408964522611830	265.477077290129
DTSMA	0.788669196092446	0.408265091531002	**263.895843821065**

5.3. Cantilever beam design problem

The second engineering optimization problem is the cantilever beam design problem, where the main goal of this type of optimization is to reduce the weight of the beam. The structure of the cantilever beam is presented in Figure 9. The mathematical model is defined as Eq. (17) [60].

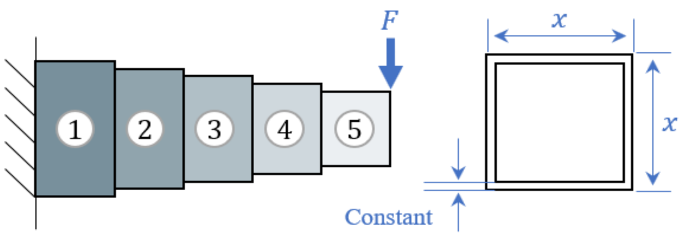

Figure 9. Cantilever beam structure and design variables.

$$
\begin{aligned}
&\text{Consider } \vec{x} = [x_1, x_2, x_3, x_4, x_5] \\
&\text{Minimize } f(\vec{x}) = 0.06224\left(x_1 + x_2 + x_3 + x_4 + x_5\right) \\
&\text{subject to: } g(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \le 1 \\
&\text{with } 0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100.
\end{aligned}
\tag{17}
$$

The comparative results of the different algorithms solved in the literature are shown in Table 8. It can be concluded that DTSMA and SMA are superior to the well-known comparison algorithms, and DTSMA is superior to SMA.

Table 8. Optimal results and comparison for the cantilever beam design problem.

Algorithms	x_1	x_2	x_3	x_4	x_5	Optimum
IMPFA [73]	6.0162	5.3077	4.5034	3.5013	2.1451	1.34
GOA [79]	6.011674	5.31297	4.48307	3.50279	2.16333	1.33996
GCHHO [60]	6.01666788	5.31103898	4.49365848	3.50048281	2.15181437	1.339956541
SSA [23]	6.01513453	5.30930468	4.49500672	3.50142629	2.15278791	1.3399563910
PFA [66]	6.0154633	5.30902227	4.49463146	3.5017851	2.15275783	1.33995638
ALO [63]	6.01812	5.31142	4.48836	3.49751	2.158329	1.33995
WLSSA [57]	6.134865	5.360400	4.439038	3.510499	2.010312	1.338799
SMA	6.01766887	5.29094735	4.50052997	3.51084173	2.15406757	1.3365452138
DTSMA	6.01568509	5.31010010	4.49565207	3.50247159	2.14976144	**1.3365212385**

5.4. Pressure vessel design problem

The third engineering design problem used is the pressure vessel design problem [56]. The objective is to minimize the cost of cylindrical pressure vessels, including the material cost, welding and forming cost of cylindrical vessels. The problem has four decision variables: shell thickness (T_s), head thickness (T_h), radius (R), and cylindrical length (L). This problem is presented in Figure 10. The mathematical model is described as Eq. (18).

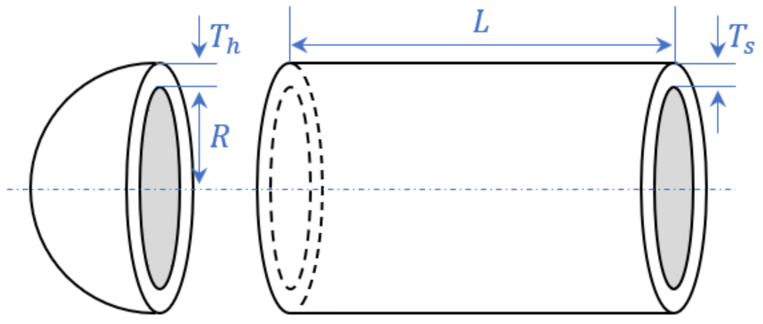

Figure 10. Design variables of pressure vessel problem.

$$
\begin{aligned}
&\text{Consider } \vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \\
&\text{Minimize } f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&\text{subject to: } g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0 \\
&\qquad g_2(\vec{x}) = -x_2 + 0.00954x_3 \le 0 \\
&\qquad g_3(\vec{x}) = -\pi x_3^2 x_4 - 4/3\pi x_3^3 + 1296000 \le 0 \\
&\qquad g_4(\vec{x}) = x_4 - 240 \le 0 \\
&\text{with } 0 \le x_1, x_2 \le 99, 10 \le x_3, x_4 \le 200.
\end{aligned}
\tag{18}
$$

The optimization results are shown in Table 9, from which it can be concluded that the DTSMA algorithm has better performance than SMA and other comparative algorithms in solving the pressure vessel problem.

Table 9. Optimal results and comparison for the pressure vessel design problem.

Algorithms	T_s	T_h	R	L	Optimal cost
HHOSCA [80]	0.945909	0.447138	48.8513	125.4684	6393.092794
AOA [76]	0.8303737	0.4162057	42.75127	169.3454	6048.7844
HHO [27]	0.81758383	0.4072927	42.09174576	176.7196352	6000.46259
POA [82]	0.8291528106	0.4098782427	42.960558426	167.09725713	5999.4001110
GCLPSO [83]	0.784508	0.387656	40.6289	195.8892	5989.654
AO [84]	1.0540	0.182806	59.6219	38.8050	5949.2258
MSCA [85]	0.779256	0.399600	40.325450	199.9213	5935.7161
NM-PSO [86]	0.8036	0.3927	41.6392	182.4120	5930.3137
IHHO [74]	0.8002	0.3955	41.4705	184.5767	5923.5
MALO [87]	0.779889	0.385340	40.35586	199.4961	5894.9214
EFOA [88]	0.78095518361	0.38602688210	40.463958486	198.00039607	5890.1193927
MBA [81]	0.7802	0.3856	40.4292	198.4964	5889.3216
IGWO [72]	0.7784458	0.3854034	40.33393	199.8019	5888.6000
TLPFA [6]	0.7785	0.3848	40.3281	199.8996	5885.8372
SMA	0.78163950498	0.38636499004	40.499442917	197.51182965	5891.2957232
DTSMA	0.77816984767	0.38464982998	40.319661250	199.99941966	**5885.3379777**

Continuous variables version.

5.5. Tension/compression spring design problem

The fourth application is the tension/compression spring design problem, which requires minimizing the weight of the spring by considering constraints on minimum deflection, shear stress and surge frequency, as well as limitations on geometry [89]. This problem has three continuous variables: diameter of the wire (*d*), diameter of the mean coil (*D*) and the active coils number (*N*). This problem is presented in Figure 11. The mathematical model is described as Eq. (19).

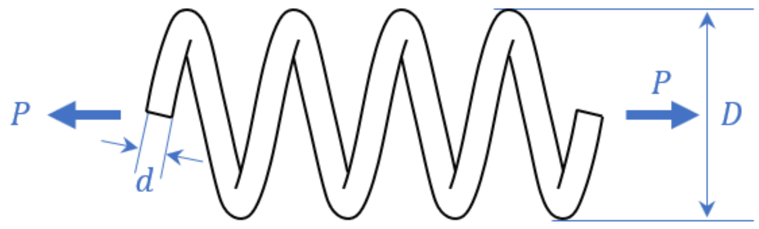

Figure 11. Design variables of tension/compression spring problem.

$$
\begin{aligned}
&\text{Consider } \vec{x}=[x_1,x_2,x_3]=[d,D,N] \\
&\text{Minimize } f(\vec{x})=(x_3+2)x_2x_1^2 \\
&\text{subject to: } g_1(\vec{x})=1-\frac{x_2^3x_3}{71785x_1^4}\le 0 \\
&\qquad g_2(\vec{x})=\frac{4x_2^2-x_1x_2}{12566(x_2x_1^3-x_1^4)}+\frac{1}{5108x_1^2}-1\le 0 \\
&\qquad g_3(\vec{x})=1-\frac{140.45x_1}{x_2^2x_3}\le 0 \\
&\qquad g_4(\vec{x})=\frac{x_1+x_2}{1.5}-1\le 0 \\
&\text{with } 0.05\le x_1\le 2, 0.25\le x_2\le 1.3, 2\le x_3\le 15.
\end{aligned}
\tag{19}
$$

The optimization results are shown in Table 10, the results show that DTSMA obtains better optimal weights than SMA.

Table 10. Optimal results and comparison for the tension/compression spring design problem.

Algorithms	*d*	*D*	*N*	Optimal weight
HHOSCA [80]	0.054693	0.433378	7.891402	0.012822904
IGWO [72]	0.05159	0.354337	11.4301	0.012700
SSA [23]	0.051207	0.345215	12.004032	0.0126763
RW-GWO [90]	0.05167	0.35613	11.33056	0.012674
PVS [91]	0.05169	0.35680	11.28442	0.01267
MGWO [78]	0.051640	0.355530	11.36064	0.012668
MSCA [85]	0.051668	0.356199	11.3207	0.0126670
QISCA [92]	0.051425	0.350404	11.669237	0.012667
SGLSCA [93]	0.05179	0.3591	11.1490	0.0126669
MALO [87]	0.051759	0.358411	11.191500	0.0126660

Continued on next page

Algorithms	*d*	*D*	*N*	Optimal weight
GWO [21]	0.05169	0.356737	11.28885	0.012666
EO [17]	0.0516199100	0.355054381	11.38796759	0.012666
HHO [27]	0.051796393	0.359305355	11.138859	0.012665443
CSA [30]	0.051178	0.358851	11.164981	0.012665370
PFA [66]	0.05172695	0.33576296	11.235724	0.01266528
SMA	0.051042782273	0.341367881985	12.24907263821	0.012672956271
DTSMA	0.051682558573	0.356560684570	11.29820387501	**0.012665270005**

5.6. Welded beam design problem

Another engineering design problem is the welded beam problem, which is often used as a benchmark case for testing different optimization algorithms [62]. The problem contains nearly 3.5% of the feasible region in the search space. The structure and design variables are illustrated in Figure 12. The objective of this problem is to minimize fabricating cost subjected to shear stress (τ), bending stress (σ), buckling load (P_c), deflection (δ), and other constraints [17]. This problem has four parameters: thickness of the weld (h), length of welded part of the beam (l), height of the beam (t), and width of the beam (b). The mathematical model is as Eq. (20).

$$
\begin{aligned}
&\text{Consider } \vec{x}=[x_1,x_2,x_3,x_4]=[h,l,t,b]\\
&\text{Minimize } f(\vec{x})=1.10471x_1^2x_2+0.04811x_3x_4\left(14.0+x_2\right)\\
&\text{subject to: } g_1(\vec{x})=\tau(\vec{x})-\tau_{\max}\le 0;\ \ g_2(\vec{x})=\sigma(\vec{x})-\sigma_{\max}\le 0\\
&\qquad g_3(\vec{x})=\delta(\vec{x})-\delta_{\max}\le 0;\ \ g_4(\vec{x})=x_1-x_4\le 0\\
&\qquad g_5(\vec{x})=P-P_c(\vec{x})\le 0;\ \ g_6(\vec{x})=0.125-x_1\le 0\\
&\qquad g_7(\vec{x})=1.10471x_1^2+0.04811x_3x_4\left(14.0+x_2\right)-5.0\le 0\\
&\text{where } \tau(\vec{x})=\sqrt{(\tau')^2+2\tau'\tau''\frac{x_2}{2R}+(\tau'')^2};\ \ \tau'=\frac{P}{\sqrt{2}x_1x_2};\ \ \tau''=\frac{MR}{J};\\
&\qquad M=P\left(L+\frac{x_2}{2}\right);\ \ R=\sqrt{\frac{x_2^2}{4}+\left(\frac{x_1+x_3}{2}\right)^2};\\
&\qquad J=2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4}+\left(\frac{x_1+x_3}{2}\right)^2\right]\right\};\ \ \sigma(\vec{x})=\frac{6PL}{x_4x_3^2};\\
&\qquad \delta(\vec{x})=\frac{6PL^3}{Ex_3^2x_4};\ \ P_c(\vec{x})=\frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}\left(1-\frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right);\\
&\qquad P=6000\text{ lb};\ \ L=14\text{ inch};\ \ \delta_{\max}=0.25\text{ inch};\ \ E=30\times10^6\text{ psi};\\
&\qquad G=12\times10^6\text{ psi};\ \ \tau_{\max}=13600\text{ psi};\ \ \sigma_{\max}=30000\text{ psi.}\\
&\text{with } 0.1\le x_1,x_4\le 2, 0.1\le x_2,x_3\le 10.
\end{aligned}
\tag{20}
$$

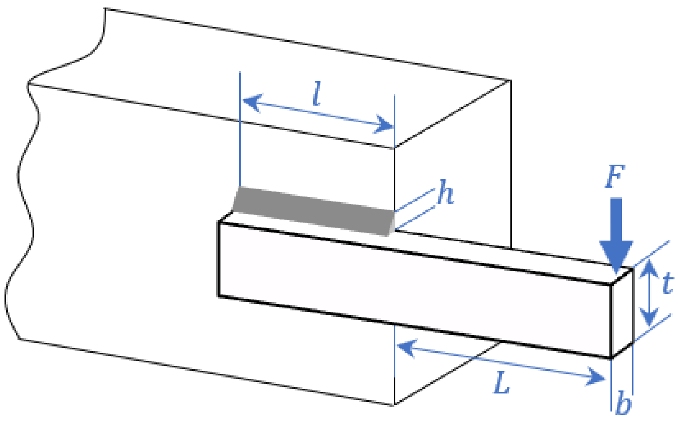

Figure 12. Welded beam structure and design variables.

Table 11 presents the optimization results of DTSMA and other well-known algorithms in the literature. The results demonstrated that DTSMA excelled in solving the welded beam problem, outperforming many improved algorithms and recently proposed metaheuristic algorithms.

Table 11. Optimal results and comparison for the welded beam design problem.

Algorithms	*h*	*l*	*t*	*b*	Optimal cost
HHOSCA [80]	0.190086	3.696496	9.386343	0.204157	1.779032249
POA [82]	0.202511	3.542971	9.033488	0.206170	1.732394281
HHO [27]	0.204039	3.531061	9.027463	0.206147	1.73199057
GWO [21]	0.205676	3.478377	9.03681	0.205778	1.72624
IGWO [72]	0.20496	3.4872	9.0366	0.20573	1.7254
SSA [23]	0.2057	3.4714	9.0366	0.2057	1.72491
EO [17]	0.2057	3.4705	9.03664	0.2057	1.7249
PFA [66]	0.2057295	3.470495	9.036624	0.2057297	1.7248530
MBA [81]	0.205729	3.470493	9.036626	0.205729	1.724853
CSA [30]	0.205730	3.470489	9.036624	0.205730	1.724852
CLSOBBOA [94]	0.205729	3.470488	9.036622	0.205729	1.724852
TLMPA [95]	0.20572964	3.470488666	9.03662391	0.20572964	1.724852
MTDE [71]	0.205730	3.470489	9.036624	0.205730	1.724852
MBFPA [96]	0.205730	3.470473	9.036623	0.205729	1.72485185
NM-PSO [86]	0.205830	3.468338	9.036624	0.205730	1.724717
BBSCA [97]	0.2057	3.4705	9.0373	0.2057	1.7247
IHHO [74]	0.20533	3.47226	9.0364	0.2010	1.7238
SOA [25]	0.205408	3.472316	9.035208	0.201141	1.723485
WQSMA [55]	0.18850	3.56850	9.10685	0.20542	1.72129
AOA [76]	0.194475	2.57092	10.000	0.201827	1.7164
GCLPSO [83]	0.20799	3.25802	9.02820	0.208064	1.715355
WDDA [98]	0.1803	3.5925	9.6537	0.2028	1.6997

Continued on next page

Algorithms	h	l	t	b	Optimal cost
MALO [87]	0.205670	3.247600	9.060900	0.20567	1.698100
MSCA [85]	0.20545	3.252400	9.057600	0.20568	1.697900
TLPFA [6]	0.2051	3.2679	9.0366	0.2059	1.6961
WLSSA [57]	0.205387	3.25923	9.036633	0.205730	1.695573
IMPFA [73]	0.2057	3.2539	9.0375	0.2057	1.6953
GCHHO [60]	0.20572287	3.25324354	9.03661412	0.20573009	1.695255645
SMA	0.205730609	3.253178392	9.036345969	0.205742741	1.695307346
DTSMA	0.205728772	3.253133196	9.036632844	0.205729603	**1.695248922**

5.7. Speed reducer design problem

Another important engineering design problem is the speed reducer design problem, which is a challenging design problem since it is correlated to seven variables [30]. A graphical illustration of this problem is shown in Figure 13. The objective of this problem is to minimize the weight subjected to different constraints of bending stress, surface stress, lateral deflection of the shaft and stress in the shaft. The seven design variables considered are: face width (B), number of tooth modules (M), number of teeth in the pinion (N), length of the first shaft between bearings (L_1), length of the second shaft between bearings (L_2), and diameters of the first and second shafts (D_1, D_2). The third variable is an integer, while the other variables are continuous, and the problem comprises nearly 0.4% of the feasible region. The mathematical formulation of this problem is as Eq. (21).

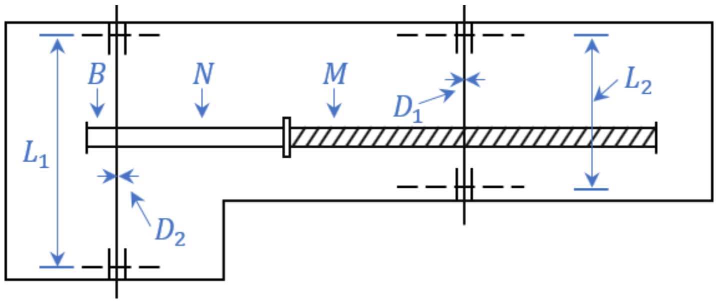

Figure 13. Design variables of speed reducer problem.

$$\begin{aligned}
&\text{Consider } \vec{x}=[x_1,x_2,x_3,x_4,x_5,x_6,x_7]=[B,M,N,L_1,L_2,D_1,D_2]\\
&\text{Minimize } f(\vec{x})=0.7854x_1x_2^2\left(14.9334x_3+3.3333x_3^2-43.0934\right)\\
&\qquad -1.508x_1\left(x_6^2+x_7^2\right)+7.4777\left(x_6^3+x_7^3\right)+0.7854\left(x_4x_6^2+x_5x_7^2\right)\\
&\text{subject to: } g_1(\vec{x})=\frac{27}{x_1x_2^2x_3}\le 1;\quad g_2(\vec{x})=\frac{397.5}{x_1x_2^2x_3^2}\le 1\\
&\qquad g_3(\vec{x})=\frac{1.9x_4^3}{x_2x_3x_6^4}\le 1;\quad g_4(\vec{x})=\frac{1.93x_5^3}{x_2x_3x_7^4}\le 1\\
&\qquad g_5(\vec{x})=\frac{\sqrt{\left(745\left(x_4/(x_2x_3)\right)\right)^2+16.9\times 10^6}}{110x_6^3}\le 1\\
&\qquad g_6(\vec{x})=\frac{\sqrt{\left(745\left(x_5/(x_2x_3)\right)\right)^2+157.5\times 10^6}}{85x_7^3}\le 1\\
&\qquad g_7(\vec{x})=\frac{x_2x_3}{40}\le 1;\quad g_8(\vec{x})=\frac{5x_2}{x_1}\le 1\\
&\qquad g_9(\vec{x})=\frac{x_1}{12x_2}\le 1;\quad g_{10}(\vec{x})=\frac{1.5x_6+1.9}{x_4}\le 1\\
&\qquad g_{11}(\vec{x})=\frac{1.1x_7+1.9}{x_5}\le 1\\
&\text{with } 2.6\le x_1\le 3.6, 0.7\le x_2\le 0.8, x_3\in\{17,18,\cdots,28\},\\
&\qquad 7.3\le x_4,x_5\le 8.3, 2.9\le x_6\le 3.9, 5\le x_7\le 5.5.
\end{aligned} \tag{21}$$

The optimization results for DTSMA, SMA and several other algorithms are given in Table 12. The results show that the performance of DTSMA is better than that of SMA.

Table 12. Optimal results and comparison for the speed reducer design problem.

Variables	HHOSCA [80]	HEAACT [99]	MBA [81]	PVS [91]	SMA	DTSMA
B	3.506119	3.50002290	3.5	3.5	3.500000600	3.500000000
M	0.7	0.70000039	0.7	0.7	0.700000000	0.700000000
N	17	17.0000129	17	17	17	17
L_1	7.3	7.30042774	7.300033	7.3	7.300001858	7.300000000
L_2	7.99141	7.71537745	7.715772	7.71532	7.715354167	7.715319916
D_1	3.452569	3.35023097	3.350218	3.35021	3.350214698	3.350214666
D_2	5.286749	5.28666370	5.286654	5.28665	5.286655037	5.286654465
Optimum	3029.873076	2994.49911	2994.482453	2994.47107	2994.472442	**2994.471066**

5.8. Multiple-disc clutch brake design problem

The multi-disc clutch brake problem is a well-known problem in engineering constrained optimization, as shown in Figure 14 [89]. Five discrete design variables are considered to minimize the weight of the multi-disc clutch brake: the inner radius (R_i), the outer radius (R_o), the thickness of the disc (T_h), the driving force (F), and the number of friction surfaces (Z). There are eight different constraints based on geometry and operating conditions. The mathematical model of this optimization problem is described as Eq. (22).

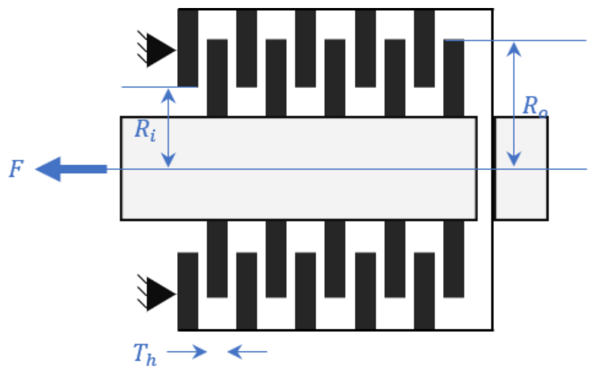

Figure 14. Design variables of multi-disc clutch brake problem.

$$
\begin{aligned}
&\text{Consider } \vec{x}=[x_1,x_2,x_3,x_4,x_5]=[R_i,R_o,T_h,F,Z]\\
&\text{Minimize } f(\vec{x})=\pi\left(R_o^2-R_i^2\right)T_h\left(Z+1\right)\rho\\
&\text{subject to: } g_1(\vec{x})=R_o-R_i-\Delta r\ge 0;\ \ g_2(\vec{x})=l_{\max}-\left(Z+1\right)\left(T_h+\delta\right)\ge 0\\
&\qquad g_3(\vec{x})=p_{\max}-p_{rz}\ge 0;\ \ g_4(\vec{x})=p_{\max}v_{sr\max}-p_{rz}v_{sr}\ge 0\\
&\qquad g_5(\vec{x})=v_{sr\max}-v_{sr}\ge 0;\ \ g_6(\vec{x})=T_{\max}-T\ge 0\\
&\qquad g_7(\vec{x})=M_h-sM_s\ge 0;\ \ g_8(\vec{x})=T\ge 0\\
&\text{where } M_h=\frac{2}{3}\mu FZ\frac{R_o^3-R_i^3}{R_o^2-R_i^2};\ \ p_{rz}=\frac{F}{\pi\left(R_o^2-R_i^2\right)};\\
&\qquad v_{sr}=\frac{2\pi n\left(R_o^3-R_i^3\right)}{90\left(R_o^2-R_i^2\right)};\ \ T=\frac{I_z\pi n}{30\left(M_h+M_f\right)};\\
&\qquad \Delta r=20\text{ mm};\ \ l_{\max}=30\text{ mm};\ \ v_{sr\max}=10\ \text{m/s};\mu=0.5;\\
&\qquad \delta=0.5\text{ mm};\ \ M_s=40\text{ Nm};\ \ M_f=3\text{ Nm};\ \ n=250\text{ rpm};\ \ s=1.5;\\
&\qquad p_{\max}=1\text{ MPa};\ \ I_z=55\text{ kg mm}^2;\ \ T_{\max}=15\text{ s};\ \ \rho=7.8\times10^{-6}\ \text{kg/mm}^3.\\
&\text{with } R_i\in\{60,61,\cdots,80\},\ R_o\in\{90,91,\cdots,110\},\ T_h\in\{1,1.5,2,2.5,3\};\\
&\qquad F\in\{600,610,620,\cdots,1000\},\ Z\in\{2,3,4,5,6,7,8,9\}.
\end{aligned}
\tag{22}
$$

Table 13 shows the results of DTSMA and other algorithms in the literature for optimizing multiple-disc clutch brakes. Both DTSMA and SMA find better results than other algorithms, indicating that DTSMA has good performance in solving discrete constraint problems.

Table 13. Optimal results and comparison for the multiple-disc clutch brake design problem.

Algorithms	R_i	R_o	T_h	F	Z	Optimal weight
GOA [89]	71	92	1	835	3	0.3355146
EOBL-GOA [89]	70	90	1	984	3	0.31365661
PVS [91]	70	90	1	980	3	0.31366
FSO [100]	70	90	1	870	3	0.3136566105
TLBO [20]	70	90	1	810	3	0.31365661
WCA [101]	70	90	1	910	3	0.3136566
SMA	70	90	1	1000	3	0.3136566105
DTSMA	70	90	1	980	3	**0.3136566105**

5.9. Car side crash design problem

The car side crash optimization design problem was originally proposed by Gu et al. [102]. This optimization problem is specified as minimizing an objective function with eleven mixed design variables with ten constraint limits, as shown in Figure 15. The simplified mathematical model of the problem can be written in the following Eq. (23).

The optimization results of different algorithms are given in Table 14. The results show that DTSMA outperforms PSO, GA, GOA, ABC, GWO, CODE and SMA algorithms in optimizing the car side impact design problem and can obtain satisfactory results.

Table 14. Optimal results and comparison for the car side crash design problem.

Variables	PSO [104]	GA [104]	GOA [89]	ABC [105]	GWO	CODE	SMA	DTSMA
x_1	0.50000	0.50005	0.50000	0.50000	0.50043	0.50001	0.50000	0.50000
x_2	1.11670	1.28017	1.11670	1.06240	1.11516	1.11678	1.11975	1.11610
x_3	0.50000	0.50001	0.50000	0.51480	0.50000	0.50000	0.50000	0.50000
x_4	1.30208	1.03302	1.30208	1.44910	1.30518	1.30160	1.29678	1.30264
x_5	0.50000	0.50001	0.50000	0.50000	0.50109	0.50000	0.50000	0.50000
x_6	1.50000	0.50000	1.50000	1.50000	1.50000	1.49996	1.50000	1.50000
x_7	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000
x_8	0.34500	0.34994	0.34500	0.34500	0.34500	0.34500	0.34500	0.34500
x_9	0.19200	0.19200	0.19200	0.19200	0.34500	0.34500	0.19200	0.34500
x_{10}	-19.54935	10.3119	-19.54935	-29.34000	-19.78034	-19.49082	-18.95652	-19.60863
x_{11}	-0.00431	0.00167	-0.00431	0.74109	0.65129	-0.14707	0.25926	0.06832
Optimum	22.84474	22.85653	22.84474	23.17500	22.85094	22.84339	22.84380	**22.84301**

$$
\begin{aligned}
&\text{Consider } \vec{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}] \\
&\text{Minimize } f(\vec{x}) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \\
&\text{subject to: } g_1(\vec{x}) = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \le 1 \\
&\qquad g_2(\vec{x}) = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 \\
&\qquad\qquad + 0.0008757x_5x_{10} + 0.080405x_6x_9 + 0.00139x_8x_{11} \\
&\qquad\qquad + 0.00001575x_{10}x_{11} \le 0.32 \\
&\qquad g_3(\vec{x}) = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 \\
&\qquad\qquad - 0.018x_2x_7 + 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 \\
&\qquad\qquad + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + 0.00121x_8x_{11} \le 0.32 \\
&\qquad g_4(\vec{x}) = 0.074 - 0.061x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} \\
&\qquad\qquad - 0.166x_7x_9 + 0.227x_2^2 \le 0.32 \\
&\qquad g_5(\vec{x}) = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} \\
&\qquad\qquad + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \le 32 \\
&\qquad g_6(\vec{x}) = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11x_2x_8 \\
&\qquad\qquad - 0.0215x_5x_{10} - 9.98x_7x_8 + 22x_8x_9 \le 32 \\
&\qquad g_7(\vec{x}) = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} \le 32 \\
&\qquad g_8(\vec{x}) = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} \\
&\qquad\qquad + 0.009325x_6x_{10} + 0.000191x_{11}^2 \le 4 \\
&\qquad g_9(\vec{x}) = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} \\
&\qquad\qquad - 0.0198x_4x_{10} + 0.028x_6x_{10} \le 9.9 \\
&\qquad g_{10}(\vec{x}) = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} \\
&\qquad\qquad - 0.0556x_9x_{11} - 0.000786x_{11}^2 \le 15.7 \\
&\text{with } 0.5 \le x_1, x_2, x_3, x_4, x_5, x_6, x_7 \le 1.5, x_8, x_9 \in \{0.192, 0.345\}, -30 \le x_{10}, x_{11} \le 30.
\end{aligned}
\tag{23}
$$

5.10. Statistics analysis of engineering problems

In order to observe the distribution of the best fitness when solving engineering problems by DTSMA and SMA, the results of 30 runs are presented in the form of box plots, as shown in Figure 16. The corresponding Wilcoxon *p*-value test results are shown in Figure 17, where the number 1 represents three-bar truss problem, the number 2 represents cantilever beam problem, and so on.

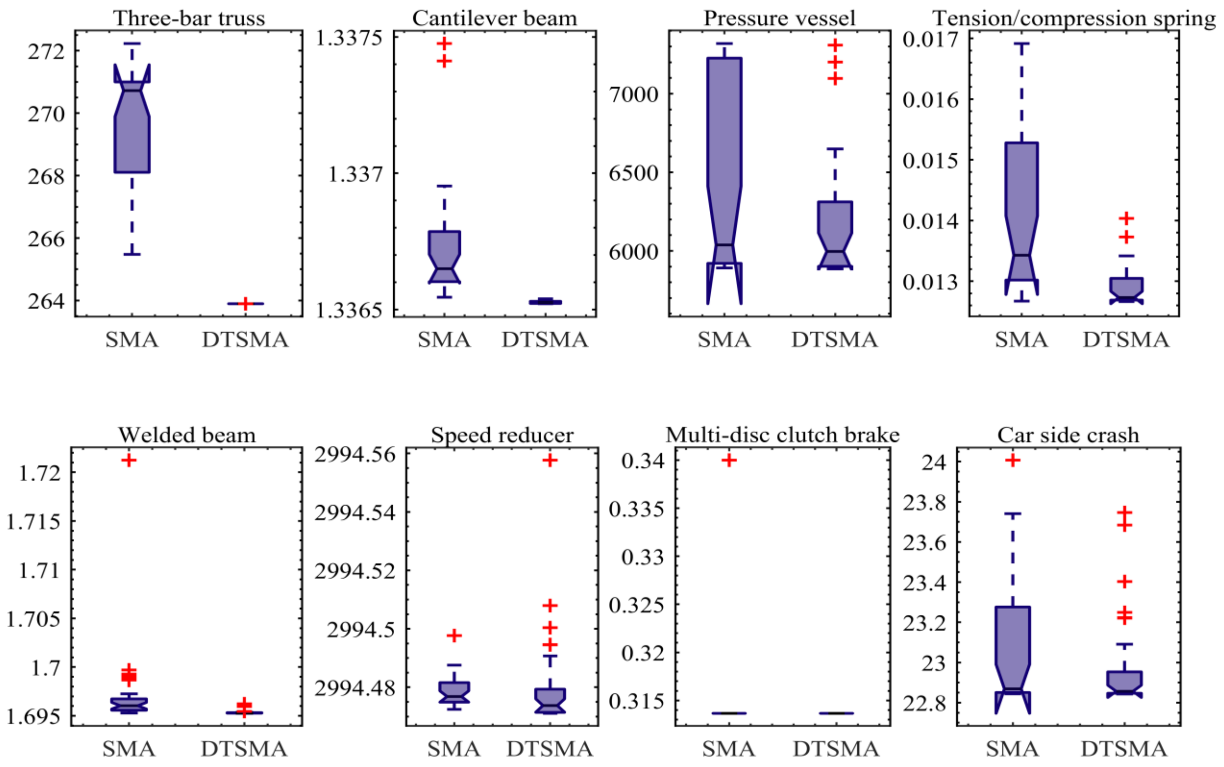

Figure 15. Box plots of the DTSMA and SMA in solving engineering problems.

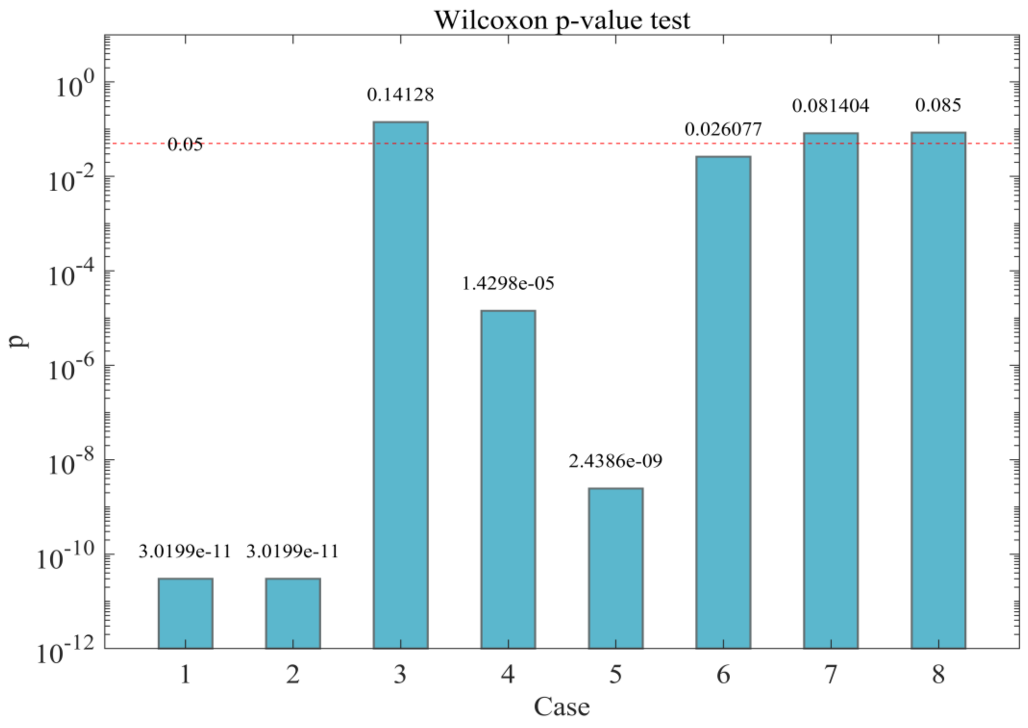

Figure 16. Statistical test of the DTSMA and SMA for eight engineering problems.

It can be summarized that as follows from Figure 16 and Figure 17.

(1) For the three-bar truss, cantilever beam, tension/compression spring, and welded beam engineering design problems, the boxes of DTSMA are significantly lower than those of SMA, and

the Wilcoxon *p*-value test results are much less than 0.05, indicating that the solution accuracy and robustness of DTSMA for these four engineering problems are significantly better than SMA.

(2) DTSMA performs better than SMA but not significantly enough for the pressure vessel, speed reducer, and car side crash engineering design problems. Although DTSMA obtains better results, it also produces more outliers and is less stable. For the multi-disc clutch brake problem, both DTSMA and SMA find the same optimal solution because it is a discrete numerical problem and the difference in feasible solutions is smaller.

(3) On the whole, DTSMA still outperforms SMA, and according to the NFL theorem [31], no single algorithm can be applied to all problems. DTSMA outperforms SMA for most engineering problems, which indicates that the improvement is meaningful.

6. Inverse kinematics solution of 7-DOF robot manipulator

Seven-degree-of-freedom (7-DOF) robot manipulators are widely used in industry for their ability to easily avoid obstacles, move flexibly, and work in larger spaces. The inverse kinematics of a robotic arm is defined as finding the joint angle by using the kinematics equations of the desired end-effector position. Due to its complex nonlinear structure, the inverse kinematics problem can be considered as a challenging optimization problem [106].

The most used method for kinematics modeling of robotic arms is the Denavit-Hartenberg (DH) coordinate parameter method. The robot manipulator model solved in this paper is proposed by Serkan et al. in [107], and its DH parameter table is listed in Table 15, where $a_i, \alpha_i, d_i, \theta_i$ refer to link length, link twist, link offset and joint angle, respectively. The model structure of a robotic arm can be determined according to the DH parameter table, as shown in Figure 18.

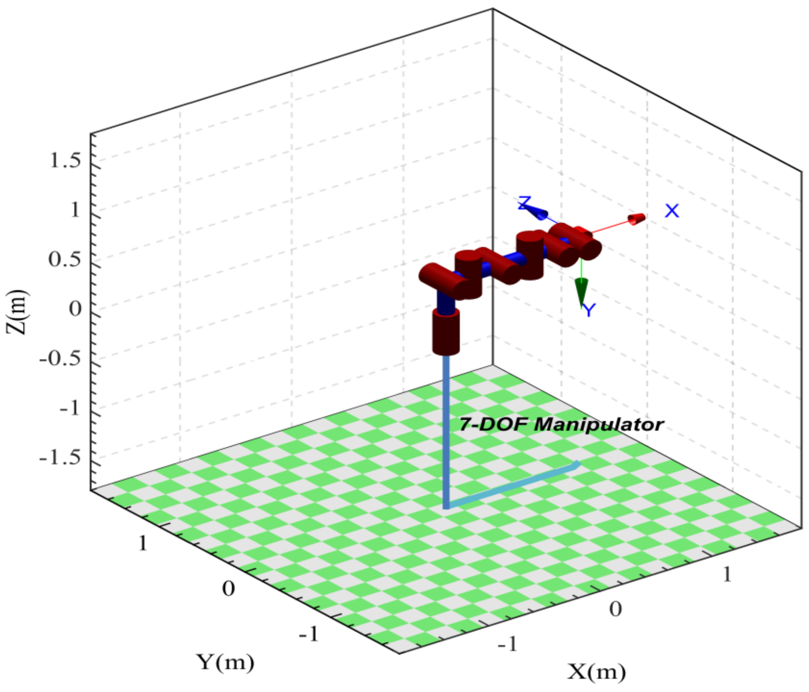

Figure 17. 7-DOF robot manipulator link structure.

Table 15. DH parameters for 7-DOF robot manipulator.

Joint	a_i (m)	α_i (°)	d_i (m)	θ_i (°)
1	0	−90	l_1=0.5	$-180<\theta_1<180$
2	l_2=0.2	90	0	$-90<\theta_2<30$
3	l_3=0.25	−90	0	$-90<\theta_3<120$
4	l_4=0.3	90	0	$-90<\theta_4<90$
5	l_5=0.2	−90	0	$-90<\theta_5<90$
6	l_6=0.2	0	0	$-90<\theta_6<90$
7	l_7=0.1	0	d_7=0.05	$-30<\theta_7<90$

The general homogeneous transformation matrix can be expressed as Eq. (24).

$$ {}_{i-1}^{\;\;i}T = \begin{bmatrix} c\theta_i & -c\alpha_i \cdot s\theta_i & s\alpha_i \cdot s\theta_i & a_i \cdot c\theta_i \\ s\theta_i & c\alpha_i \cdot c\theta_i & -c\theta_i \cdot s\alpha_i & a_i \cdot s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

where ${}_{i-1}^{\;\;i}T$ is the transformation matrix relating joint $i-1$ to joint i, s and c denote sine and cosine functions, respectively.

The kinematics equations of the serial robot manipulator can be obtained by substituting the values of the DH parameter in Table 15 into Eq. (24) and then multiplying them successively, as shown in Eq. (25).

$$ \begin{gathered} T_{\text{end-effector}} = {}_0^7T = {}_0^1T \cdot {}_1^2T \cdot {}_2^3T \cdot {}_3^4T \cdot {}_4^5T \cdot {}_5^6T \cdot {}_6^7T = \begin{bmatrix} n_x & s_x & a_x & P_x \\ n_y & s_y & a_y & P_y \\ n_z & s_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}_0^1T = \begin{bmatrix} c\theta_1 & 0 & -s\theta_1 & 0 \\ s\theta_1 & 0 & c\theta_1 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; {}_1^2T = \begin{bmatrix} c\theta_2 & 0 & s\theta_2 & l_2c\theta_2 \\ s\theta_2 & 0 & -c\theta_2 & l_2s\theta_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; {}_2^3T = \begin{bmatrix} c\theta_3 & 0 & -s\theta_3 & l_3c\theta_3 \\ s\theta_3 & 0 & c\theta_3 & l_3s\theta_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \\ {}_3^4T = \begin{bmatrix} c\theta_4 & 0 & s\theta_4 & l_4c\theta_4 \\ s\theta_4 & 0 & -c\theta_4 & l_4s\theta_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; {}_4^5T = \begin{bmatrix} c\theta_5 & 0 & -s\theta_5 & l_5c\theta_5 \\ s\theta_5 & 0 & c\theta_5 & l_5s\theta_5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \\ {}_5^6T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & l_6c\theta_6 \\ s\theta_6 & c\theta_6 & 0 & l_6s\theta_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; {}_6^7T = \begin{bmatrix} c\theta_7 & -s\theta_7 & 0 & l_7c\theta_7 \\ s\theta_7 & c\theta_7 & 0 & l_7s\theta_7 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{gathered} \tag{25}$$

where Tend-effector is the homogeneous transformation matrix of the end-effector with respect to the base frame, $n_x, n_y, n_z, s_x, s_y, s_z, a_x, a_y, a_z$ denote the rotational elements of the transformation matrix, P_x, P_y, P_z denote the elements of the position vector of end-effector.

The main task of the inverse kinematics problem is to determine the corresponding joint angles based on the desired position of the end-effector in the Cartesian coordinate system. Thus, the objective function can be defined as minimizing the Euclidean distance between the desired position and the predicted position, as shown in Eq. (26).

$$\text{Minimize } Error(\vec{\theta}) = \left\| P_{\text{desired}} - P_{\text{predicted}} \right\| \tag{26}$$

where P_{desired} represents the desired position vector of end-effector of robot manipulator, $P_{\text{predicted}}$ represents the predicted position vector.

Table 16. Comparative results for the robot manipulator inverse kinematics problem.

Case		DTSMA	SMA	PSO	DE	GQPSO	CODE	GWO	HHO
	P_x(m)	-0.250000	-0.249993	-0.249914	-0.250243	-0.250220	-0.249936	-0.249992	-0.249963
	P_y(m)	1.000000	0.999992	0.999774	0.999946	0.997496	0.999968	1.000030	0.999873
	P_z(m)	0.500000	0.500006	0.499520	0.499851	0.500095	0.500034	0.500024	0.500016
	θ_1(°)	98.1094	-180.0000	129.2701	179.9137	13.4544	121.8851	180.0000	93.0924
	θ_2(°)	27.5052	29.1515	-47.3514	-33.5445	1.5614	0.7168	-24.5950	-32.6298
	θ_3(°)	10.2280	-89.9894	-55.9539	-85.6810	97.7035	7.8961	-90.0000	50.0495
	θ_4(°)	-20.5711	4.9214	71.7076	11.4524	-0.0844	10.3278	1.8493	63.6740
P_1	θ_5(°)	-13.4537	-4.1895	7.4930	-14.2824	2.4710	-74.2812	-2.8740	-54.3151
	θ_6(°)	-76.2966	-41.7951	-26.4275	-22.2372	0.2848	-39.2841	-1.7651	-8.6285
	θ_7(°)	18.1857	10.2074	7.0484	61.8365	-0.0596	55.8021	61.0002	0.7233
	Best	**3.41E-07**	1.21E-05	5.37E-04	2.90E-04	2.52E-03	7.89E-05	3.93E-05	1.33E-04
	Worst	**3.24E-05**	3.82E-04	4.22E-03	8.56E-03	2.97E-02	8.95E-04	8.31E-03	2.35E-02
	Mean	**8.86E-06**	1.19E-04	2.04E-03	3.60E-03	1.09E-02	3.41E-04	4.77E-04	2.22E-03
	Std.	**9.68E-06**	1.12E-04	7.63E-04	2.52E-03	5.85E-03	2.11E-04	1.49E-03	5.28E-03
	Time(s)	41.19	16.64	16.13	18.73	16.36	22.55	15.97	43.15
	P_x(m)	0.500001	0.500097	0.499959	0.500008	0.490452	0.500182	0.499986	0.500000
	P_y(m)	-0.249998	-0.250043	-0.249860	-0.249966	-0.261733	-0.249910	-0.250022	-0.250000
	P_z(m)	0.750000	0.750026	0.750104	0.750018	0.743542	0.749790	0.750040	0.750000
	θ_1(°)	-180.0000	-118.4679	180.0000	-90.2388	180.0000	-175.9374	-72.7182	-34.4058
P_2	θ_2(°)	-67.6761	0.0001	-90.0000	28.8407	5.9500	-83.5671	-90.0000	27.6332
	θ_3(°)	119.9737	90.4584	1.8322	112.9309	120.0000	27.3226	-0.9113	-79.8749
	θ_4(°)	-54.0772	9.9188	-90.0000	-85.6953	-18.4606	-81.3090	90.0000	-89.8656
	θ_5(°)	-12.2186	86.5447	42.1089	-86.3252	75.2238	45.4308	70.8576	34.9869

Continued on next page

Case	DTSMA	SMA	PSO	DE	GQPSO	CODE	GWO	HHO
θ_6(°)	-85.9133	-87.8100	-85.6967	90.0000	-13.7060	-87.2805	52.1677	-80.1758
θ_7(°)	-29.3020	0.0000	90.0000	75.1437	-2.9405	87.4752	-23.2101	0.3696
Best	2.41E-06	1.09E-04	1.80E-04	3.94E-05	1.64E-02	1.97E-05	4.79E-05	**5.48E-08**
Worst	**9.24E-05**	5.59E-02	3.44E-02	1.10E-02	1.30E-01	5.82E-04	1.57E-02	2.05E-01
Mean	**3.30E-05**	7.14E-03	2.13E-03	3.22E-03	6.22E-02	1.45E-04	9.18E-04	9.55E-03
Std.	**2.42E-05**	1.53E-02	6.11E-03	3.06E-03	2.33E-02	1.08E-04	3.03E-03	3.76E-02
Time(s)	41.49	15.67	16.68	18.49	16.70	22.96	16.46	42.56

The optimal values are shown in bold.

To verify the performance of the proposed DTSMA, two different desired position vectors i.e., $P_1 = [-0.25, 1.00, 0.50]^T$ and $P_2 = [0.50, -0.25, 0.75]^T$ were selected for testing. DTSMA was compared with SMA, PSO, DE, GQPSO [68], CODE [70], GWO, and HHO, and each algorithm was run independently for 30 times with 30 individual populations and 1000 iterations. Table 16 illustrates the numerical optimization results of DTSMA and the other compared algorithms for the inverse kinematics problem with two different desired position coordinates of the end-effector.

From the optimization results in Table 16, it can be seen that the solution accuracy of the DTSMA is better than the comparison algorithm for the inverse kinematics problem, but the computation time is longer. The optimal solution of the HHO algorithm for Case2 optimization is better than DTSMA, but its average solution is poorer and less stable. The convergence history of the algorithm is shown in Figure 19. It can be seen that DTSMA converges faster than the other comparison algorithms. The statistical results of the comparison algorithms are shown in Figure 20 and Figure 21.

It can be seen that the performance of DTSMA is significantly better than SMA and the other six comparison algorithms in optimizing the inverse kinematics problem of the robot manipulator, which reflects the applicability of DTSMA to practical problems.

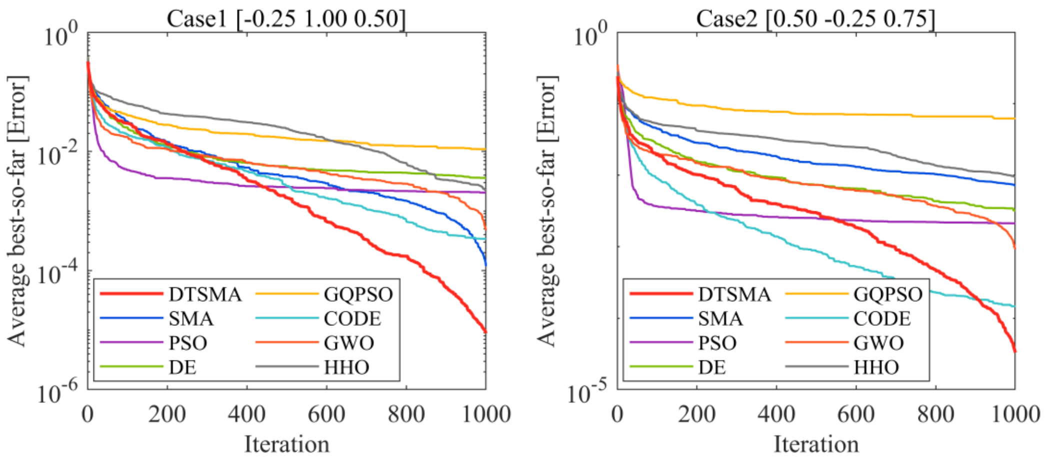

Figure 18. Convergence curves of the algorithms for inverse kinematics problems.

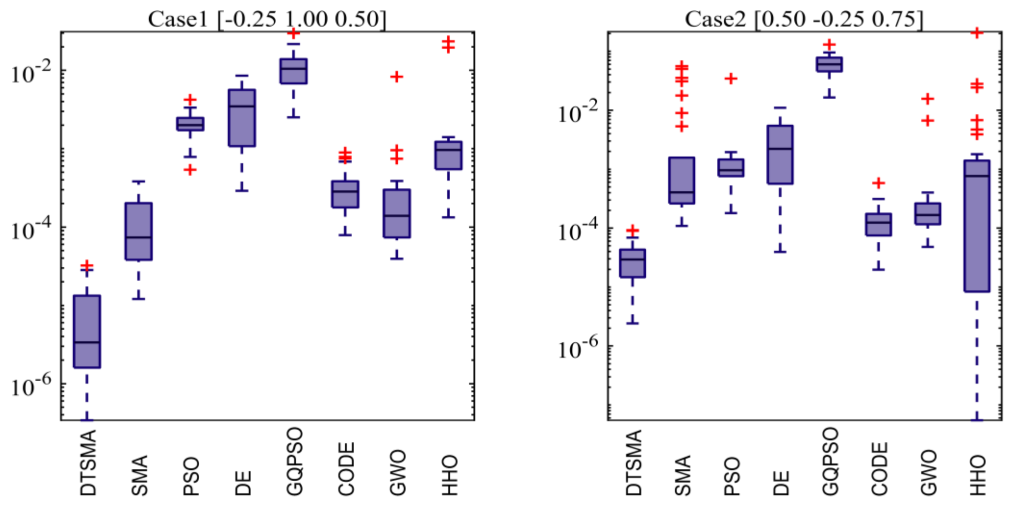

Figure 19. Box plots of the algorithms for inverse kinematics problems.

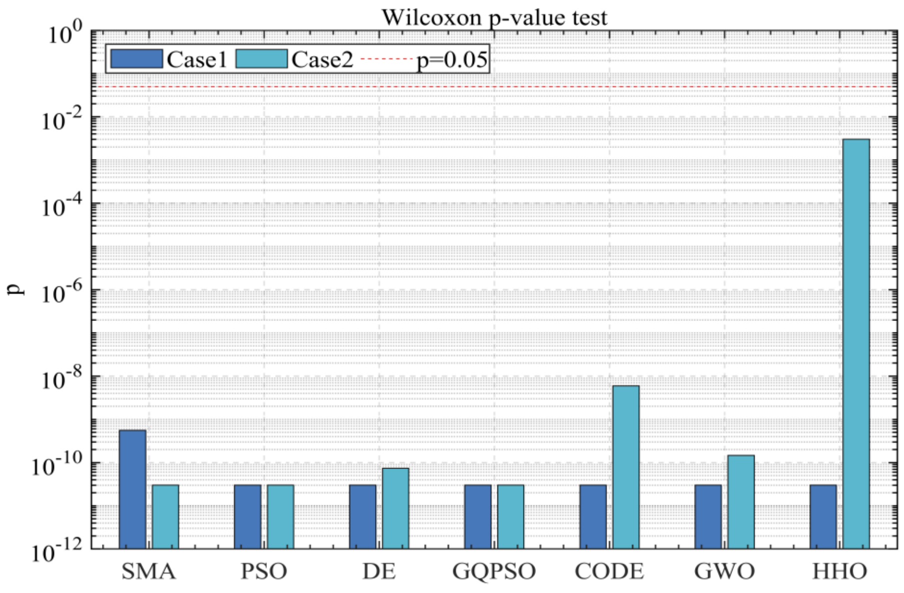

Figure 20. Statistical test of the DTSMA and compare algorithms for inverse kinematics problems.

7. Conclusions and future works

In this paper, an improved slime mould algorithm, DTSMA, is proposed for the shortcomings of slow convergence, weak exploration ability, and easy to fall into local optimal of the SMA. In DTSMA, the dominant swarm strategy is firstly introduced to retain the historical optimal position of each slime mould individual. In the position updating formula of exploration stage, both the historical optimal position of the population and the historical optimal position of the individual are used to make the population look for places with high probability of the optimal solution as much as possible. Secondly, by making full use of SMA's fitness ranking information, the dominant population is further divided into dominant and inferior population, and the two sub-populations cooperate with each other to make the population search more extensive in the exploration stage. Then, a nonlinear adaptive t-distribution mutation strategy is introduced to perturb the dominant swarm to avoid premature convergence. Finally, the exploitation mechanism for convergence to individual historical optimal is added to improve the diversity of the population and the robustness and generalization ability of DTSMA. The effectiveness and efficiency of DTSMA in solving numerical optimization problems were tested on CEC2019 functions. Then, DTSMA was applied to eight classical engineering application problems and the inverse kinematics problem of a 7-DOF robot manipulator. Experimental results show that the solution accuracy of DTSMA on CEC2019 ranks first overall among 23 algorithms, significantly outperforming SMA and numerous comparative algorithms. In eight engineering instances, DTSMA obtains better optimal solutions, far outperforming SMA for the three-bar truss, cantilever beam, tension/compression spring, and welded beam problems, and slightly outperforming SMA for the remaining four problems. In the inverse kinematics of the robot manipulator, DTSMA significantly outperforms SMA, PSO, DE, GQPSO, CODE, GWO and HHO in terms of solution accuracy and stability. The statistical results demonstrate that DTSMA has the following superiority.

(1) The test function results show that DTSMA converges quickly and accurately, has the ability to escape from local optimum, and has a good balance between exploration and exploitation. The convergence curve shows that the proposed method has fast convergence speed and avoids premature convergence and local optimal stagnation.

(2) Friedman and Wilcoxon rank test illustrate that DTSMA has better performance compared to SMA and well-known algorithms and there are significant differences.

(3) Experimental results of DTSMA in engineering problems show that it is an ideal choice for solving continuous and discrete constrained optimization problems as well as inverse kinematics problems of robot manipulator.

Although DTSMA overcomes many drawbacks of the original SMA, its long running time makes it unsuitable for real-time control systems. More in-depth study on how to reduce the time complexity of DTSMA will be conducted in the future. Then, DTSMA will be applied to the inverse kinematics of robot manipulator with comprehensive consideration of position and posture of end-effector. In addition, DTSMA has stronger scalability and can also be applied to solve high or ultra-high dimensional problems, such as the traveling salesman problem, the job shop scheduling problem, and the time series forecasting problem, etc.

Acknowledgments

This work was supported by the National Science Foundation of China under Grant No. 62066005, and Project supported by Hainan Provincial Natural Science Foundation of China, No.620QN237, and Project supported by the Education Department of Hainan Province, No. Hnky2020-5.

Conflict of interests

The authors declare no conflict of interest.

References

1. J. Fliege, L. M. G. Drummond, B. F. Svaiter, Newton's method for multiobjective optimization, *SIAM J. Optim.*, **20** (2009), 602–626. doi: 10.1137/08071692X.

2. Ž. Povalej, Quasi-Newton’s method for multiobjective optimization, *J. Comput. Appl. Math.*, **255** (2013), 765–777. doi: 10.1016/j.cam.2013.06.045.
3. J. Zhang, Y. Xiao, Z. Wei, Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained optimization, *Math. Probl. Eng.*, **2009** (2009), 1–16. doi: 10.1155/2009/243290.
4. M.-W. Li, Y.-T. Wang, J. Geng, W.-C. Hong, Chaos cloud quantum bat hybrid optimization algorithm, *Nonlinear Dyn.*, **103** (2021), 1167–1193. doi: 10.1007/s11071-020-06111-6.
5. D. Izci, S. Ekinci, Comparative performance analysis of slime mould algorithm for efficient design of proportional–integral–derivative controller, *Electrica*, **21** (2021), 151–159. doi: 10.5152/electrica.2021.20077.
6. C. Tang, Y. Zhou, Z. Tang, Q. Luo, Teaching-learning-based pathfinder algorithm for function and engineering optimization problems, *Appl. Intell.*, (2020). doi: 10.1007/s10489-020-02071-x.
7. J. J. Grefenstette, Genetic algorithms and machine learning, *Mach. Learn.*, **3** (1988), 95–99. doi: 10.1023/A:1022602019183.
8. R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.*, **11** (1997), 341–359. doi: 10.1023/A:1008202821328.
9. S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, *J. Stat. Phys.*, **34** (1984), 975–986. doi: 10.1007/BF01009452.
10. L. K. Grover, A fast quantum mechanical algorithm for database search, *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC ’96*, (1996), 212–219. doi: 10.1145/237814.237866.
11. O. K. Erol, I. Eksin, A new optimization method: Big Bang–Big Crunch, *Adv. Eng. Softw.*, **37** (2005), 106–111. doi: 10.1016/j.advengsoft.2005.04.005.
12. B. Alatas, ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization, *Expert Syst. Appl.*, **38** (2011), 13170–13180. doi: 10.1016/j.eswa.2011.04.126.
13. H. Shareef, A. A. Ibrahim, A. H. Mutlag, Lightning search algorithm, *Appl. Soft Comput.*, **36** (2015), 315–333. doi: 10.1016/j.asoc.2015.07.028.
14. S. Mirjalili, S. M. Mirjalili, A. Hatamlou, Multi-Verse Optimizer: A nature-inspired algorithm for global optimization, *Neural Comput. Appl.*, **27** (2015), 495–513. doi: 10.1007/s00521-015-1870-7.
15. V. K. Patel, V. J. Savsani, Heat transfer search (HTS): A novel optimization algorithm, *Inf. Sci.*, **324** (2015), 217–246. doi: 10.1016/j.ins.2015.06.044.
16. W. Zhao, L. Wang, Z. Zhang, A novel atom search optimization for dispersion coefficient estimation in groundwater, *Future Gener. Comput. Syst.*, **91** (2018), 601–610. doi: 10.1016/j.future.2018.05.037.
17. A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, *Knowl.-Based Syst.*, **191** (2020), 105190. doi: 10.1016/j.knosys.2019.105190.
18. R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, (1995), 39–43. doi: 10.1109/MHS.1995.494215.
19. 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. doi:

10.1007/s10898-007-9149-x.

20. R. V. Rao, V. J. Savsani, D. P. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *Comput.-Aided Des.*, 43 (2011), 303–315. doi: 10.1016/j.cad.2010.12.015.
21. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey Wolf Optimizer, *Adv. Eng. Softw.*, **69** (2014), 46–61. doi: 10.1016/j.advengsoft.2013.12.007.
22. S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, *Adv. Eng. Softw.*, **95** (2016), 51–67. doi: 10.1016/j.advengsoft.2016.01.008.
23. 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. Softw.*, **114** (2017), 163–191. doi: 10.1016/j.advengsoft.2017.07.002.
24. E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-Cisneros, A swarm optimization algorithm inspired in the behavior of the social-spider, *Expert Syst. Appl.*, **40** (2013), 6374–6384. doi: 10.1016/j.eswa.2013.05.041.
25. G. Dhiman, V. Kumar, Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems, *Knowl.-Based Syst.*, **165** (2018), 169–196. doi: 10.1016/j.knosys.2018.11.024.
26. A. Faramarzi, M. Heidarinejad, S. Mirjalili, A. H. Gandomi, Marine Predators Algorithm: A nature-inspired metaheuristic, *Expert Syst. Appl.*, **152** (2020), 113377. doi: 10.1016/j.eswa.2020.113377.
27. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. doi: 10.1016/j.future.2019.02.028.
28. H. A. Alsattar, A. A. Zaidan, B. B. Zaidan, Novel meta-heuristic bald eagle search optimisation algorithm, *Artif. Intell. Rev.*, **53** (2019), 2237–2264. doi: 10.1007/s10462-019-09732-5.
29. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, *Future Gener. Comput. Syst.*, **111** (2020), 300–323. doi: 10.1016/j.future.2020.03.055.
30. M. S. Braik, Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems, *Expert Syst. Appl.*, **174** (2021), 114685. doi: 10.1016/j.eswa.2021.114685.
31. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, **1** (1996), 67–82. doi: 10.1109/4235.585893.
32. Y. Zhang, X. Han, Y. Dong, J. Xie, G. Xie, X. Xu, A novel state transition simulated annealing algorithm for the multiple traveling salesmen problem, *J. Supercomput.*, (2021). doi: 10.1007/s11227-021-03744-1.
33. K. Yu, B. Qu, C. Yue, S. Ge, X. Chen, J. Liang, A performance-guided JAYA algorithm for parameters identification of photovoltaic cell and module, *Appl. Energy*, **237** (2019), 241–257. doi: 10.1016/j.apenergy.2019.01.008.
34. C. Fan, Y. Zhou, Z. Tang, Neighborhood centroid opposite-based learning Harris Hawks optimization for training neural networks, *Evol. Intell.*, (2020). doi: 10.1007/s12065-020-00465-x.
35. A. A. Ewees, L. Abualigah, D. Yousri, Z. Y. Algamal, M. A. A. AI-qaness, R. A. Ibrahim, et al., Improved Slime Mould Algorithm based on Firefly Algorithm for feature selection: A case study on QSAR model, *Eng. Comput.*, (2021). doi: 10.1007/s00366-021-01342-6.

36. M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, M. J. Ryan, S. Mirjalili, An efficient binary slime mould algorithm integrated with a novel attacking-feeding strategy for feature selection, *Comput. Ind. Eng.*, **153** (2021), 107078. doi: 10.1016/j.cie.2020.107078.
37. M. Abdel-Basset, V. Chang, R. Mohamed, HSMA_WOA: A hybrid novel Slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images, *Appl. Soft Comput.*, **95** (2020), 106642. doi: 10.1016/j.asoc.2020.106642.
38. S. Zhao, P. Wang, A. A. Heidari, H. Chen, H. Turabieh, M. Mafarja, et al., Multilevel threshold image segmentation with diffusion association slime mould algorithm and Renyi's entropy for chronic obstructive pulmonary disease, *Comput. Biol. Med.*, **134** (2021), 104427. doi: 10.1016/j.compbiomed.2021.104427.
39. M. K. Naik, R. Panda, A. Abraham, Normalized square difference based multilevel thresholding technique for multispectral images using leader slime mould algorithm, *J. King Saud Univ. - Comput. Inf. Sci.*, (2020). doi: 10.1016/j.jksuci.2020.10.030.
40. D. Yousri, A. Fathy, H. Rezk, T. S. Babu, M. R. Berber, A reliable approach for modeling the photovoltaic system under partial shading conditions using three diode model and hybrid marine predators-slime mould algorithm, *Energy Convers. Manag.*, **243** (2021), 114269. doi: 10.1016/j.enconman.2021.114269.
41. M. Mostafa, H. Rezk, M. Aly, E. M. Ahmed, A new strategy based on slime mould algorithm to extract the optimal model parameters of solar PV panel, *Sustain. Energy Technol. Assess.*, **42** (2020), 100849. doi: 10.1016/j.seta.2020.100849.
42. A. A. El-Fergany, Parameters identification of PV model using improved slime mould optimizer and Lambert W-function, *Energy Rep.*, **7** (2021), 875–887. doi: 10.1016/j.egyr.2021.01.093.
43. Y. Liu, A. A. Heidari, X. Ye, G. Liang, H. Chen, C. He, Boosting slime mould algorithm for parameter identification of photovoltaic models, *Energy*, **234** (2021), 121164. doi: 10.1016/j.energy.2021.121164.
44. C. Kumar, T. D. Raj, M. Premkumar, T. D. Raj, A new stochastic slime mould optimization algorithm for the estimation of solar photovoltaic cell parameters, *Optik*, **223** (2020), 165277. doi: 10.1016/j.ijleo.2020.165277.
45. D. Agarwal, P. S. Bharti, Implementing modified swarm intelligence algorithm based on Slime moulds for path planning and obstacle avoidance problem in mobile robots, *Appl. Soft Comput.*, **107** (2021), 107372. doi: 10.1016/j.asoc.2021.107372.
46. R. M. Rizk-Allah, A. E. Hassanien, D. Song, Chaos-opposition-enhanced slime mould algorithm for minimizing the cost of energy for the wind turbines on high-altitude sites, *ISA Trans.*, (2020). doi: 10.1016/j.isatra.2021.04.011.
47. M. H. Hassan, S. Kamel, L. Abualigah, A. Eid, Development and application of slime mould algorithm for optimal economic emission dispatch, *Expert Syst. Appl.*, **182** (2021), 115205. doi: 10.1016/j.eswa.2021.115205.
48. Y. Wei, Y. Zhou, Q. Luo, W. Deng, Optimal reactive power dispatch using an improved slime mould algorithm, *Energy Reports*, **7** (2021), 8742–8759. doi: 10.1016/j.egyr.2021.11.138.
49. B. Abdollahzadeh, S. Barshandeh, H. Javadi, N. Epicoco, An enhanced binary slime mould algorithm for solving the 0–1 knapsack problem, *Eng. Comput.*, (2021). doi: 10.1007/s00366-021-01470-z.
50. S. L. Zubaidi, I. H. Abdulkareem, K. S. Hashim, H. Al-Bugharbee, H. M. Ridha, S. K. Gharghan, et al., Hybridised Artificial Neural Network Model with Slime Mould Algorithm: A Novel

Methodology for Prediction of Urban Stochastic Water Demand, *Water*, **12** (2020), 2692. doi: 10.3390/w12102692.

51. Z. Chen, W. Liu, An Efficient Parameter Adaptive Support Vector Regression Using K-Means Clustering and Chaotic Slime Mould Algorithm, *IEEE Access*, **8** (2020), 156851–156862. doi: 10.1109/ACCESS.2020.3018866.
52. S. Ekinci, D. Izci, H. L. Zeynelgil, S. Orenc, An Application of Slime Mould Algorithm for Optimizing Parameters of Power System Stabilizer, *in 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey*, (2020), 1–5. doi: 10.1109/ISMSIT50672.2020.9254597.
53. Y. M. Wazery, E. Saber, E. H. Houssein, A. A. Ali, E. Amer, An Efficient Slime Mould Algorithm Combined With K-Nearest Neighbor for Medical Classification Tasks, *IEEE Access*, **9** (2021), 113666–113682. doi: 10.1109/ACCESS.2021.3105485.
54. M. Premkumar, P. Jangir, R. Sowmya, H. H. Alhelou, A. A. Heidari, H. Chen, MOSMA: Multi-Objective Slime Mould Algorithm Based on Elitist Non-Dominated Sorting, *IEEE Access*, **9** (2021), 3229–3248. doi: 10.1109/ACCESS.2020.3047936.
55. C. Yu, A. Asghar Heidari, X. Xue, L. Zhang, H. Chen, W. Chen, Boosting Quantum Rotation Gate Embedded Slime Mould Algorithm, *Expert Syst. Appl.*, (2021), 115082. doi: 10.1016/j.eswa.2021.115082.
56. E. H. Houssein, M. A. Mahdy, M. J. Blondin, D. Shebl, W. M. Mohamed, Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems, *Expert Syst. Appl.*, **174** (2021), 114689. doi: 10.1016/j.eswa.2021.114689.
57. H. Ren, J. Li, H. Chen, C. Li, Adaptive levy-assisted salp swarm algorithm: Analysis and optimization case studies, *Math. Comput. Simul.*, **181** (2020), 380–409. doi: 10.1016/j.matcom.2020.09.027.
58. J. Zhao, Z.-M. Gao, W. Sun, The improved slime mould algorithm with Levy flight, *J. Phys. Conf. Ser.*, **1617** (2020), 012033. doi: 10.1088/1742-6596/1617/1/012033.
59. X. Zhang, Y. Xu, C. Yu, A. A. Heidari, S. Li, H. Chen, et al., Gaussian mutational chaotic fruit fly-built optimization and feature selection, *Expert Syst. Appl.*, **141** (2019), 112976. doi: 10.1016/j.eswa.2019.112976.
60. S. Song, P. Wang, A. A. Heidari, M. Wang, X. Zhao, H. Chen, et al., Dimension decided Harris hawks optimization with Gaussian mutation: Balance analysis and diversity patterns, *Knowl.-Based Syst.*, **215** (2020), 106425. doi: 10.1016/j.knosys.2020.106425.
61. N. Kumar, I. Hussain, B. Singh, B. Panigrahi, Single Sensor-Based MPPT of Partially Shaded PV System for Battery Charging by Using Cauchy and Gaussian Sine Cosine Optimization, *IEEE Trans. Energy Convers.*, (2017), 983–992. doi: 10.1109/TEC.2017.2669518.
62. S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowl.-Based Syst.*, **89** (2015), 228–249. doi: 10.1016/j.knosys.2015.07.006.
63. S. Mirjalili, The Ant Lion Optimizer, *Adv. Eng. Softw.*, **83** (2015), 80–98. doi: 10.1016/j.advengsoft.2015.01.010.
64. S. Mirjalili, Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Comput. Appl.*, **27** (2016), 1053–1073. doi: 10.1007/s00521-015-1920-1.
65. S. Mirjalili, SCA: A Sine Cosine Algorithm for solving optimization problems, *Knowl.-Based*

Syst., **96** (2016), 120–133. doi: 10.1016/j.knosys.2015.12.022.

66. H. Yapici, N. Cetinkaya, A new meta-heuristic optimizer: Pathfinder algorithm, *Appl. Soft Comput.*, **78** (2019), 545–568. doi: 10.1007/s13369-014-1156-x.
67. S. Mirjalili, A. Lewis, A. S. Sadiq, Autonomous Particles Groups for Particle Swarm Optimization, *Arab. J. Sci. Eng.*, **39** (2014), 4683–4697. doi: 10.1007/s13369-014-1156-x.
68. L. dos S. Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, *Expert Syst. Appl.*, **37** (2010), 1676–1683. doi: 10.1016/j.eswa.2009.06.044.
69. S. Mirjalili, S. Z. M. Hashim, A new hybrid PSOGSA algorithm for function optimization, *2010 International Conference on Computer and Information Application*, (2010), 374–377. doi: 10.1109/ICCIA.2010.6141614.
70. S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad, G. F. Naterer, Computing opposition by involving entire population, *2014 IEEE Congress on Evolutionary Computation (CEC)*, (2014), 1800–1807. doi: 10.1109/CEC.2014.6900329.
71. M. H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, H. Faris, MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems, *Appl. Soft Comput.*, **97** (2020), 106761. doi: 10.1016/j.asoc.2020.106761.
72. Y. Li, X. Lin, J. Liu, An Improved Gray Wolf Optimization Algorithm to Solve Engineering Problems, *Sustainability*, **13** (2021), 3208. doi: 10.3390/su13063208.
73. C. Tang, Y. Zhou, Q. Luo, Z. Tang, An enhanced pathfinder algorithm for engineering optimization problems, *Eng. Comput.*, (2021). doi: 10.1007/s00366-021-01286-x.
74. A. G. Hussien, M. Amin, A self-adaptive Harris Hawks optimization algorithm with opposition-based learning and chaotic local search strategy for global optimization and feature selection, *Int. J. Mach. Learn. Cybern.*, (2021). doi: 10.1007/s13042-021-01326-4.
75. A. H. Gandomi, X.-S. Yang, A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, *Eng. Comput.*, **29** (2011), 17–35. doi: 10.1007/s00366-011-0241-y.
76. L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A. H. Gandomi, The Arithmetic Optimization Algorithm, *Comput. Methods Appl. Mech. Eng.*, **376** (2020), 113609. doi: 10.1016/j.cma.2020.113609.
77. S. Gupta, K. Deep, A. P. Engelbrecht, A memory guided sine cosine algorithm for global optimization, *Eng. Appl. Artif. Intell.*, **93** (2020), 103718. doi: 10.1016/j.engappai.2020.103718.
78. S. Gupta, K. Deep, A memory-based Grey Wolf Optimizer for global optimization tasks, *Appl. Soft Comput.*, **93** (2020), 106367. doi: 10.1016/j.asoc.2020.106367.
79. S. Saremi, S. Mirjalili, A. Lewis, Grasshopper Optimisation Algorithm: Theory and application, *Adv. Eng. Softw.*, **105** (2017), 30–47. doi: 10.1016/j.advengsoft.2017.01.004.
80. V. K. Kamboj, A. Nandi, A. Bhadoria, S. Sehgal, An intensify Harris Hawks optimizer for numerical and engineering optimization problems, *Appl. Soft Comput.*, **89** (2019), 106018. doi: 10.1016/j.asoc.2019.106018.
81. A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, *Appl. Soft Comput.*, **13** (2013), 2592–2612. doi: 10.1016/j.asoc.2012.11.026.
82. D. Wei, Z. Wang, L. Si, C. Tan, Preaching-inspired swarm intelligence algorithm and its applications, *Knowl.-Based Syst.*, **211** (2020), 106552. doi: 10.1016/j.knosys.2020.106552.

83. C. Chen, X. Wang, H. Yu, N. Zhao, M. Wang, H. Chen, An Enhanced Comprehensive Learning Particle Swarm Optimizer with the Elite-Based Dominance Scheme, *Complexity*, **2020** (2020), 1–24. doi: 10.1155/2020/4968063.
84. L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-qaness, A. H. Gandomi, Aquila Optimizer: A novel meta-heuristic optimization algorithm, *Comput. Ind. Eng.*, **157** (2021), 107250. doi: 10.1016/j.cie.2021.107250.
85. H. Chen, M. Wang, X. Zhao, A multi-strategy enhanced sine cosine algorithm for global optimization and constrained practical engineering problems, *Appl. Math. Comput.*, **369** (2019), 124872. doi: 10.1016/j.amc.2019.124872.
86. E. Zahara, Y.-T. Kao, Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems, *Expert Syst. Appl.*, **36** (2009), 3880–3886. doi: 10.1016/j.eswa.2008.02.039.
87. M. Wang, A. A. Heidari, M. Chen, H. Chen, X. Zhao, X. Cai, Exploratory differential ant lion-based optimization, *Expert Syst. Appl.*, **159** (2020), 113548. doi: 10.1016/j.eswa.2020.113548.
88. X. Yang, W. Li, L. Su, Y. Wang, A. Yang, An improved evolution fruit fly optimization algorithm and its application, *Neural Comput. Appl.*, **32** (2019), 9897–9914. doi: 10.1007/s00521-019-04512-2.
89. B. S. Yildiz, N. Pholdee, S. Bureerat, A. R. Yildiz, S. M. Sait, Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems, *Eng. Comput.*, (2021). doi: 10.1007/s00366-021-01368-w.
90. S. Gupta, K. Deep, A novel Random Walk Grey Wolf Optimizer, *Swarm Evol. Comput.*, **44** (2018), 101–112. doi: 10.1016/j.swevo.2018.01.001.
91. P. Savsani, V. Savsani, Passing vehicle search (PVS): A novel metaheuristic algorithm, *Appl. Math. Model.*, **40** (2016), 3951–3978. doi: 10.1016/j.apm.2015.10.040.
92. W. Guo, Y. Wang, F. Dai, P. Xu, Improved sine cosine algorithm combined with optimal neighborhood and quadratic interpolation strategy, *Eng. Appl. Artif. Intell.*, **94** (2020), 103779. doi: 10.1016/j.engappai.2020.103779.
93. W. Zhou, P. Wang, A. A. Heidari, M. Wang, X. Zhao, H. Chen, Multi-core sine cosine optimization: Methods and inclusive analysis, *Expert Syst. Appl.*, **164** (2020), 113974. doi: 10.1016/j.eswa.2020.113974.
94. A. S. Assiri, On the performance improvement of Butterfly Optimization approaches for global optimization and Feature Selection, *PLOS ONE*, **16** (2021), e0242612. doi: 10.1371/journal.pone.0242612.
95. K. Zhong, Q. Luo, Y. Zhou, M. Jiang, TLMPA: Teaching-learning-based Marine Predators algorithm, *AIMS Math.*, **6** (2020), 1395–1442. doi: 10.3934/math.2021087.
96. Z. Wang, Q. Luo, Y. Zhou, Hybrid metaheuristic algorithm using butterfly and flower pollination base on mutualism mechanism for global optimization problems, *Eng. Comput.*, (2020). doi: 10.1007/s00366-020-01025-8.
97. N. Li, L. Wang, Bare-Bones Based Sine Cosine Algorithm for global optimization, *J. Comput. Sci.*, **47** (2020), 101219. doi: 10.1016/j.jocs.2020.101219.
98. L. Zhong, Y. Zhou, Q. Luo, K. Zhong, Wind driven dragonfly algorithm for global optimization, *Concurr. Comput. Pract. Exp.*, **33** (2020), 1–31. doi: 10.1002/cpe.6054.
99. Y. Wang, Z. Cai, Y. Zhou, Z. Fan, Constrained optimization based on hybrid evolutionary

algorithm and adaptive constraint-handling technique, *Struct. Multidiscip. Optim.*, **37** (2009), 395–413. doi: 10.1007/s00158-008-0238-3.

100. G. Azizyan, F. Miarnaeimi, M. Rashki, N. Shabakhty, Flying Squirrel Optimizer (FSO): A Novel SI-Based Optimization Algorithm for Engineering Problems, *Iranian Journal of Optimization*, **11** (2019), 177-205.
101. H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems, *Comput. Struct.*, **110–111** (2012), 151–166. doi: 10.1016/j.compstruc.2012.07.010.
102. L. Gu, R.-J. Yang, C. Tho, M. Makowskit, O. Faruquet, Y. Li, Optimisation and robustness for crashworthiness of side impact, *Int. J. Veh. Des. - INT J VEH DES*, **26** (2001), 348–360. doi: 10.1504/IJVD.2001.005210.
103. B. D. Youn, K. K. Choi, R.-J. Yang, L. Gu, Reliability-based design optimization for crashworthiness of vehicle side impact, *Struct. Multidiscip. Optim.*, **26** (2004), 272–283. doi: 10.1007/s00158-003-0345-0.
104. A. H. Gandomi, X.-S. Yang, A. H. Alavi, Mixed variable structural optimization using Firefly Algorithm, *Comput. Struct.*, **89** (2011), 2325–2336. doi: 10.1016/j.compstruc.2011.08.002.
105. S. Sharma, A. K. Saha, G. Lohar, Optimization of weight and cost of cantilever retaining wall by a hybrid metaheuristic algorithm, *Eng. Comput.*, (2021). doi: 10.1007/s00366-021-01294-x.
106. M. Toz, Chaos-based Vortex Search algorithm for solving inverse kinematics problem of serial robot manipulators with offset wrist, *Appl. Soft Comput.*, **89** (2020), 106074. doi: 10.1016/j.asoc.2020.106074.
107. S. Dereli, R. Köker, A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm, *Artif. Intell. Rev.*, **53** (2020), 949–964. doi: 10.1007/s10462-019-09683-x.