Hindawi
Scientific Programming
Volume 2022, Article ID 1502988, 17 pages
https://doi.org/10.1155/2022/1502988

Research Article

Porcellio scaber Algorithm with *t*-Distributed Elite Mutation for Global Optimization

Chao Zhang [1] **and Yi Yang** [2,3]

[1] *Department of Computer Information, Suzhou Vocational and Technical College, Suzhou, Anhui 234000, China*
[2] *College of Computer Science and Technology, Huaibei Normal University, Huaibei, Anhui 235000, China*
[3] *Anhui Engineering Research Center for Intelligent Computing and Application on Cognitive Behavior (ICACB), Huaibei, Anhui 235000, China*

Correspondence should be addressed to Yi Yang; yiyang2009@chnu.edu.cn

Received 7 February 2022; Revised 12 April 2022; Accepted 1 June 2022; Published 28 June 2022

Academic Editor: Qianchuan Zhao

As the convergence accuracy of the *Porcellio scaber* algorithm (PSA) is low, this study proposes an improved algorithm based on the *t*-distribution elite mutation mechanism. First, the improved algorithm applies a *t*-distribution mutation to each dimension of the optimal solution of each generation. Using the dominant information of the optimal solution and *t*-distribution characteristics, the result of the mutation is then employed as the updated location of the selected porcellio; thus, the algorithm enhances the ability to jump out of local extreme values and improves the convergence speed. Second, the updated iterative rule of PSA may lead to the loss of information of the elite porcellio in the last generation. To solve this problem, the judgment mechanisms of the current and previous optimal solutions are included in the algorithm process. Finally, dynamic self-adaptive improvement is applied to the weight allocation parameters of PSA. Simulation results on 24 benchmark functions show that the improved algorithm has significant advantages in convergence accuracy, convergence speed, and stability when compared with basic PSA, PSO, GSA, and FPA. This indicates that the improved algorithm has certain advantages in terms of optimization. An optimal solution with good practicability is obtained by solving three practical engineering problems: three-bar truss, welded beam, and tension/compression-spring design.

1. Introduction

Many real-world applications involve complex optimization problems [1]. Swarm intelligence optimization algorithms can solve problems by simulating the intelligent characteristics and behavior modes of biological populations, which have high self-organization, self-adaptability, generalization, and abstraction capabilities [2, 3]. They can quickly and approximately solve certain NP-hard problems in the real world and have become an important method to effectively solve complex science and engineering optimization problems [4, 5]. Since the inception of swarm intelligence optimization algorithms, several classical algorithms, such as the genetic algorithm (GA) [6], particle swarm optimization (PSO) [7], and ant colony optimization (ACO) [8], have emerged.

The general framework of swarm intelligence optimization algorithms starts with a set of stochastic solutions and utilizes a series of metaheuristics to explore and exploit the search space to find the best approximate solution to the optimization problem. Maintaining population diversity and finding effective exploration and exploitation methods and a balance between them are the main issues in the research on swarm intelligence optimization algorithms [1, 9]. The non-free lunch theorem [10] indicates that no swarm intelligence optimization algorithm can solve all practical optimization problems, which motivates the development of a swarm intelligence optimization algorithm research field. In recent years, the research directions of swarm intelligence algorithms can be divided into three categories [1, 11]: (1) discovering the behavioral characteristics of biological communities in nature to design new algorithms. To this end, certain swarm intelligence optimization algorithms, such as the black widow optimization algorithm (BWO) [12], flower pollination algorithm (FPA) [13], whale optimization algorithm (WOA) [14], sparrow search algorithm (SSA) [15], Coronavirus herd immunity optimizer (CHIO) [16], chameleon swarm algorithm (CSA) [17], and white shark optimizer (WSO) [1] have emerged. (2) Integrating the advantages of different swarm intelligence algorithms to form hybrid algorithms. For example, some studies [4, 18–21] combine two or more algorithms to improve the efficiency of solving optimization problems. (3) Improving the existing algorithms; the commonly used ones include chaotic maps [22, 23], cellular [9, 24], and mutation operators [25, 26].

The objective functions of many real-world science and engineering problems are mostly nonlinear, nondifferentiable, multimodal, and high dimensional. Traditional gradient-based local optimization methods are not very effective [2]. Global optimization methods based on swarm intelligence do not require gradient information, only knowledge of the input and output parameters of the problem, and have shown good performance in solving complex optimization problems [5]. The areas that involve these problems include vehicle routing [20], coordinating the charging schedule of electric vehicles [27], feature selection [28], flow-shop scheduling [29, 30], wireless sensor network deployment [31], and single batch-processing machines [32].

Porcellio scaber algorithm (PSA) [33] is a new swarm intelligence optimization algorithm, biologically designed by the two main survival principles of *Porcellio scaber*. Currently, PSA has been successfully applied in engineering fields, such as microgrid optimization [34, 35] and pressure vessel design [36]; however, as an emerging algorithm, its relevant research and applications are in their infancy.

PSA has two main shortcomings. First, the iterative updating rules of the algorithm should make full use of the information of the previous generation of optimal *Porcellio scaber*; however, the algorithm lacks the judgment operation of the contemporary ($k+1$ generation) and the previous generation (k generation) optimal solutions. The contemporary ($k+1$ generation) optimal solution may not be as good as the previous generation (k generation) optimal solution; if the contemporary ($k+1$ generation) optimal solution is directly brought into the next generation ($k+2$ generation) operation, the information of the previous generation (k generation) optimal solution is lost, which affects the convergence speed of the algorithm. Second, the algorithm easily falls into the local extreme value, especially when dealing with high-dimensional optimization problems, which can easily lead to a premature algorithm and convergence stagnation.

To solve these problems, this study proposes an improved PSA with t-distribution elitist mutation (TPSA). The main contributions of this study are summarized as follows:

(1) The TPSA algorithm mainly implements dimension-by-dimension t-distribution perturbation on the position vector of each generation of elite *Porcellio scaber* and takes the variation result as the updated position of the selected *Porcellio scaber* with a certain probability. It fully uses the advantageous information contained in the elite *Porcellio scaber* and controls and leads the *Porcellio scaber* population to quickly approach the optimal solution with the help of the t-distribution characteristics to improve its convergence accuracy and speed.

(2) The judgment operation of the contemporary optimal solution and the optimal solution of the previous generation in the PSA algorithm process is added and the dynamic adaptive improvement of weight distribution parameters is carried out. This improves the balance between exploration and exploitation and improves performance.

(3) The performance of TPSA is verified in 24 well-known benchmark functions and compared with the comparison algorithms. The practicability of TPSA is verified by three real-world engineering problems, and the best solution obtained is compared with the algorithms reported in recent years.

The rest of the paper is organized as follows: Section 2 introduces the principle of the PSA algorithm, Section 3 describes the TPSA algorithm improvement strategy and algorithm implementation process in detail, Section 4 demonstrates the experimental results and analysis of convergence and stability, Section 5 introduces the application of TPSA in three real-world engineering problems, and finally, Section 6 concludes the paper.

2. *Porcellio scaber* Algorithm

Porcellio scaber is a worldwide animal species, as shown in Figure 1 [33]. *Porcellio scaber* prefers to live in damp, dark, and humus-rich places and in groups. Several studies [37, 38] have shown that *P. scaber* has two behaviors regarded as their survival laws: (1) group behavior—aggregation and (2) individual behavior—propensity to explore novel environments. When the living environment is unfavorable, they explore a new environment separately; conversely when environmental conditions are favorable, they stay together.

PSA uses the fitness function as a scale to evaluate the advantages and disadvantages of the porcellio scabers' living environment, and the mathematical modeling of the two survival rules of the *Porcellio scaber* is used as the rules of the algorithm update iteration. Its mathematical description is as follows.

In the d-dimensional search space, the population consisting of N porcellio scabers is $X = (X_1,X_2, . . ., X_n)$, and the vector $X_i^k = (x_{i1}{}^k, x_{i2}{}^k, . . ., x_{ij}{}^k . . ., x_{id}{}^k)^T$ represents the position of the ith *Porcellio scaber* of the kth generation. The movement of the aggregation behavior of the *Porcellio scaber* is modeled according to

Figure 1: *Porcellio Scaber* [33].

$$\mathrm{x}_i^{k+1} = \mathrm{x}_i^k - \left(\mathrm{x}_i^k - \underset{\mathrm{x}_j^k}{\operatorname{argmin}}\left\{ f\left(x_j^k\right) \right\} \right). \tag{1}$$

Equation (1) can also be expressed as

$$\mathrm{x}_i^{k+1} = \underset{\mathrm{x}_j^k}{\operatorname{argmin}}\left\{ f\left(x_j^k\right) \right\}. \tag{2}$$

From equation (1), all porcellio scabers eventually stay in the place with the best environmental conditions in the initial position; however, if the initial conditions are the worst environment, only the porcellio scabers with the aggregation behavior given in equation (1) cannot survive. The actual movement of the *Porcellio scaber* should be the weighted result of the two behaviors of gathering and exploring the new environment. Therefore, the final updating rule of PSA is given in the literature [33], which is calculated according to

$$\mathrm{x}_i^{k+1} = \mathrm{x}_i^k - (1-\lambda)\left(\mathrm{x}_i^k - \underset{\mathrm{x}_j^k}{\operatorname{argmin}}\left\{ f\left(x_j^k\right) \right\} \right) + \lambda p \tau, \tag{3}$$

where $\lambda \in (0, 1)$ is the weight allocation parameter of the two behaviors of gathering and exploring new environment tendencies, and the value of λ can be different for different porcellio scabers. $p\tau$ represents the motor behavior of the *Porcellio scaber* exploring a new environment, where τ is a d-dimensional random vector and p is a function of the intensity of the exploratory action, $p = f(\mathrm{x}_i{}^k + \tau)$. Each *Porcellio scaber* randomly selects any direction around its center of mass to explore the new environment. p is the simplest way to select a function that represents the fitness of the *Porcellio scaber* as the intensity of the behavior of exploring the new environment. The literature [33] provides the following definition of p with good performance:

$$p = \frac{f\left(x_i^k + \tau\right) - \min\left\{ f\left(x_i^k + \tau\right) \right\}}{\max\left\{ f\left(x_i^k + \tau\right) \right\} - \min\left\{ f\left(x_i^k + \tau\right) \right\}}. \tag{4}$$

3. *Porcellio scaber* Algorithm with *t*-Distributed Elite Mutation Mechanism

3.1. t-Distribution Elite Dimensional Variation Strategy. Each generation of the optimal *Porcellio scaber* is the elite of the *Porcellio scaber* population. The advantage information contained in it plays a key role in the updating and iteration of the algorithm, which enables its rapid convergence to the global optimal solution and improves its accuracy. To this end, the TPSA algorithm defines an elite mutation probability parameter P_t around an elite *Porcellio scaber* and performs *t*-distribution mutation on the position vector of the optimal *Porcellio scaber* in each generation dimension-by-dimension. The mutation result is directly given to the selected *Porcellio scaber* with P_t probability as the new location of position update, making full use of the advantage information of the optimal *Porcellio scaber*. The search space of the *Porcellio scaber* is controlled by the characteristics of the *t*-distribution, and the algorithm is guided to converge to the optimal solution quickly. This is expressed as follows:

$$\mathrm{x}_i^{k+1} = \gamma \times t(iter) \otimes \underset{\mathrm{x}_j^k}{\operatorname{argmin}}\left\{ f\left(x_j^k\right) \right\}, \tag{5}$$

where γ is the scaling coefficient of the step size and $t(iter)$ is the *t*-distribution of freedom with the number of iterations of the algorithm *iter*.

The curve shape of the *t*-distribution is related to the degrees of freedom. When the degree of freedom of the *t*-distribution is 1, its curve shape is similar to that of the Cauchy distribution; as the degree of freedom increases, the curve shape gradually approaches a Gaussian distribution, as shown in Figure 2. In the early iterations of the TPSA algorithm, the *iter* value is small. The *t*-distribution presents the characteristics of the Cauchy distribution, which can mutate and generate the next generation of porcellio scabers far away from the optimal *Porcellio scaber* position. Thus, the diversity of the *Porcellio scaber* population can be maintained to search in a larger space, improve the exploration ability, and prevent the algorithm from falling into local extremum and convergence stagnation. As the algorithm

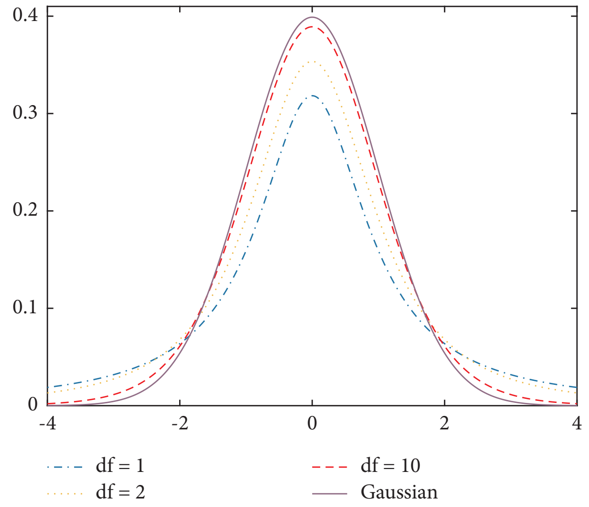

Figure 2: Curve of t-distribution.

runs iteratively to the middle and late stages, the *iter* value increases, and the t-distribution presents the characteristics of a Gaussian distribution, which can mutate around the optimal *Porcellio scaber* to generate the next generation of porcellio scabers with a certain probability; this improves the exploitation ability of PSA and increases the convergence accuracy.

3.2. Optimal Porcellio scaber Retention Strategy. From equation (3), when PSA performs a position iterative update, the k+1-generation *Porcellio scaber* position update requires the k-generation optimal *Porcellio scaber* position information. However, according to the implementation process PSA in the literature [33], the algorithm does not fully consider that, if the optimal solution of $k+1$ generation is inferior to that of k generation, and the information of the optimal solution of $k+1$ generation is included in the calculation of the position update of $k+2$ generation; it easily causes the loss of the information of the optimal solution of k generation. Therefore, after updating the contemporary position, a judgment mechanism between the contemporary and previous generation optimal solutions is added and the execution process is as follows:

```
if MinFitness $(k+1) <$ MinFitness $(k)$
        Proceed with Algorithm 1 in Section 3.4 for the next iteration
else
        $X_i^{k+1} = X_i^k = (x_{i1}{}^k, x_{i2}{}^k, . . ., x_{ij}{}^k . . ., x_{id}{}^k)$
end if
```

Here i is the number of the kth generation optimal *Porcellio scaber*, d is the dimension of the search space, and MinFitness (k) is the optimal fitness of the kth generation *Porcellio scaber* population. If the optimal fitness of the *Porcellio scaber* population of generation k+1 is less than that of the population of generation k, the next iteration continues according to the execution process of Algorithm 1. Otherwise, the optimal *Porcellio scaber* information of generation k population will be reserved for generation $k+1$ and be included in the position update calculation of generation $k+2$.

3.3. Dynamic Adaptive Strategy of Weight Allocation Parameters. The proportion of exploration and exploitation in PSA is controlled by the weight allocation parameter λ. In PSA, λ is a constant, which has a significant influence on the results of different optimization problems and is difficult to determine. To this end, this study introduces the concept of inertia weight inspired by the literature [7] and uses the following equation to calculate λ.

$$\lambda = \min + (\max - \min) \times \frac{(\max iter - iter)}{\max iter}, \tag{6}$$

where [min, max] is the upper and lower boundary of λ, max *iter* is the maximum number of iterations of the algorithm, and *iter* is the current number of iterations. In this manner, λ adaptively adjusts dynamically as the algorithm runs. In the early stage of the algorithm, λ is large and mainly controls the algorithm's exploration, which is conducive to expanding the search space and improving the convergence speed. In the middle and late stages of the algorithm, λ decreases gradually, and the algorithm mainly performs exploitation, which helps improve the convergence accuracy.

3.4. TPSA. After adopting the t-distributed elite mutation, optimal *Porcellio scaber* retention, and dynamic adaptive strategy of weight allocation parameter, the TPSA algorithm is as follows:

3.5. Time Complexity Analysis. Given that the objective function of the optimization problem is $f(x)$, the dimension of the search space is d. According to the description of PSA and the operation rules of time complexity, the time complexity of PSA is $\mathrm{T(PSA)} = \mathrm{O}(d + f(d))$.

TPSA adds the calculation of t-distribution elite variation, dynamic weight parameters, and optimal *Porcellio scaber* retention strategy. According to the execution process of Algorithm 1, the time complexity of each incremental part can be deduced. T(t-distribution elite variation) = $\mathrm{O}(d + f(d))$, T(dynamic weight parameters) = O(1), T(retention strategy of optimal *Porcellio scaber*) = O(1), and T(TPSA) = T(PSA) + T(t-distribution elite variation) + T(dynamic weight parameters) + T(retention strategy of optimal *Porcellio scaber*). After simplification, $\mathrm{T(TPSA)} = \mathrm{O}(d + f(d))$. The time complexity of TPSA and PSA is of the same order of magnitude, with an insignificant increase and no negative impact on the algorithm.

4. Experimental Results and Analysis

4.1. Experiment Design and Parameter Settings. TPSA was compared with classical and emerging intelligent algorithms such as PSA [33], PSO [7], GSA [39], and FPA [13] on 24 benchmark functions to verify the performance of the TPSA.

```
Set the objective function $f(x)$, $x = [x_1,x_2, \ldots, x_d]^T$
Set the algorithm parameters: population size $N$, elite mutation probability parameter $P_t$, the upper and lower boundaries of the weight allocation parameter $\lambda$, and step scale coefficient $\gamma$.
Initialize the Porcellio scaber population $x_i^0 (i = 1,2, \ldots, N)$
$f(x)$ determines the environmental condition $E_x$ at $x$
for $k = 1$ to maxiter
Obtain the current optimal position: $x\_\text{best} = \underset{x_j^k}{\operatorname{argmin}}\left\{f\left(x_j^k\right)\right\}$
Randomly generate an exploration environment vector $\tau = [\tau_1, \tau_2, \ldots, \tau_d]^T$
Calculate the optimal environmental conditions $\min\{E_x\}$ and the worst environmental conditions $\max\{E_x\}$ at position $\mathrm{x}_i^k+\tau$
  for $i = 1$ to $n$
Calculate the action intensity $p$ according to Equation (4)
Update individual position $X_i^{k+1}$ according to Equation (3)
  if $p_t >$ rand
Update the mutant Porcellio scaber position $X_i^{k+1}$ according to Equation (5)
  end if
The individual fitness after position update was calculated
  end for
  if MinFitness$(k + 1) >$ MinFitness$(k)$
Reserved the optimal information of the previous generation to the next generation
  end if
end for
```

Algorithm 1: TPSA algorithm.

The 24 benchmark functions are shown in Table 1, including nine high-dimensional unimodal functions, nine high-dimensional multimodal functions, and six low-dimensional functions. The following hardware and software configurations were used: CPU—2.70 GHZ (COREi5), RAM—16 GB, OS—Win10 (64 bits), DISK—SSD (512 GB), and MATLAB R2020a.

Without loss of generality, the algorithm was independently run 50 times for each function, and the worst, best, mean, standard deviation, and success rate were calculated as evaluation metrics for the algorithm performance.

The optimization precision was set to 10^{-10} and the success rate of optimization was calculated as follows:

$$\text{Success rate (SR)} = \frac{(\text{optimization success number})}{(\text{total number of experiments})}. \tag{7}$$

In the experiment, if |actual solution value − theoretical optimal value |<10^{-10}, the search was considered as a success.

Parameter Settings. The population size of the five algorithms was $N = 30$, and the number of iterations was set to 1,000. Through a large number of numerical experiments, we concluded that TPSA performs better by setting the following parameters: $P_t = 0.3$, a step-size scaling coefficient on f_{13}, f_{19}–f_{22} as 30, and the others as 1; λ linearly reduced from 0.9 to 0.2. The PSA weight allocation parameter $\lambda = 0.8$. The PSO learning factor $c_1 = c_2 = 2$, the inertia weight linearly reduced from 0.9 to 0.4, and the maximum speed was half of the search space. The GSA gravitational constant $G_0 = 100$ and acceleration $a = 20$. The FPA probability conversion parameter $p = 0.2$.

4.2. Results and Discussion.

The experimental results and correlation analysis are presented separately according to the type of each benchmark function.

4.2.1. Type I: High-Dimensional Unimodal Function.

Table 2 shows the experimental results of the five algorithms on nine high-dimensional unimodal functions. For the success rate of the search, TPSA had a success rate of 100% for all nine functions, and the success rate of the comparison algorithms was mostly zero or lower, except for GSA, which had a success rate of 100% on the f_1 function. For the convergence to the theoretical optimal value of the function, TPSA found the theoretical optimal value 0 of eight functions except f_4, and the robustness of the algorithm was strong from the mean and standard deviation indicators. PSA, GSA, and FPA could not find the theoretical optimal value of all functions. PSO, which sets the maximum speed to half of the search space, observed the theoretical optimal value of the function on f_1, f_4, and f_6 functions; however, its robustness was poor, and judging from the mean and standard deviation indicators, the success rate of the search was low. For the f_4 function, although TPSA could not converge to the theoretical optimal value 0 of the function, the mean value of the algorithm was $1.7908E-290$, whereas the worst value reached $5.2583E-289$, with a standard deviation of approximately zero. This indicates that TPSA is more stable and robust than PSA, PSO, GSA, and FPA on

Table 1: Benchmark functions.

Category	Function	Dim	Range	f_{opt}
I	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-100, 100]$	0
	$f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]$	0
	$f_4(x) = \max_{1 \le i \le n} \lvert x_i \rvert$	30	$[-100, 100]$	0
	$f_5(x) = \sum_{i=1}^{(n/4)} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4.$	30	$[-4, 5]$	0
	$f_6(x) = (\sum_{i=1}^{n} x_i^2)^2$	30	$[-600, 600]$	0
	$f_7(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5x_i)^2 + (\sum_{i=1}^{n} 0.5x_i)^4$	30	$[-5, 10]$	0
	$f_8(x) = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$	30	$[-100, 100]$	0
	$f_9(x) = 10^6 x_1^2 + \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
II	$f_{10}(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)$	30	$[-100, 100]$	0
	$f_{11}(x) = -20e\left(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}\right) - e\left((1/n)\sum_{i=1}^{n} \cos 2\pi x_i\right) + 20 + e$	30	$[-32, 32]$	0
	$f_{12}(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	30	$[-600, 600]$	0
	$f_{13}(x) = (\pi/n)\left\{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]\right.$ $\left. + \left\{(y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\right\}$ where $y_i = 1 + (x_i + 1)/4,\ u(x_i, a, k, m) = \left\{\begin{array}{cc} k(x_i - a)^m, & x_i > a, \\ 0, & -a < x_i < a, \\ k(-x_i - a)^m, & x_i < -a, \end{array}\right\}$	30	$[-50, 50]$	0
	$f_{14}(x) = \sum_{i=1}^{n} x_i^6 (2 + \sin(1/x_i))$	30	$[-600, 600]$	0
	$f_{15}(x) = \sum_{i=1}^{n} \left[\sum_{k=0}^{k_{\max}} a^k \cos(2\pi b^k (x_i + 0.5)) - n\sum_{k=0}^{k_{\max}} a^k \cos(\pi b^k)\right]$	30	$[-0.5, 0.5]$	0
	$f_{16}(x) = 0.1n - \left(0.1\sum_{i=1}^{n} \cos(5\pi x_i) - \sum_{i=1}^{n} x_i^2\right)$	30	$[-1, 1]$	0
	$f_{17}(x) = \left[(1/(n-1))\sum_{i=1}^{n-1} (\sqrt{s_i} \cdot (\sin(50.0 s_i^{0.2}) + 1))\right]^2,\ s_i = \sqrt{x_i^2 + x_{i+1}^2}$	30	$[-100, 100]$	0
	$f_{18}(x) = \left\lvert (\sum_{i=1}^{n} x_i^2)^2 - (\sum_{i=1}^{n} x_i)^{1/2} \right\rvert + \left(05\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i\right)/n + 0.5$	30	$[-100, 100]$	0
III	$f_{19}(x) = -\sum_{i=1}^{5} [(X - \mathrm{a}_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.1532
	$f_{20}(x) = -\sum_{i=1}^{7} [(X - \mathrm{a}_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.428
	$f_{21}(x) = -\sum_{i=1}^{10} [(X - \mathrm{a}_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363
	$f_{22}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$	-3.32
	$f_{23}(x) = \left(\sin^2\sqrt{x_1^2 + x_2^2} - 0.5\right)/[1 + 0.001(x_1^2 + x_2^2)]^2 - 0.5$	2	$[-100, 100]$	-1
	$f_{24}(x) = -(1 + \cos(12\sqrt{x_1^2 + x_2^2}))/(0.5(x_1^2 + x_2^2) + 2)$	2	$[-100, 100]$	-1

this function. In essence, TPSA achieved a better performance in convergence accuracy and robustness than PSA, PSO, GSA, and FPA when searching for a high-dimensional unimodal function.

4.2.2. Type II: High-Dimensional Multimodal Functions. These functions often have numerous local minima distributed in the solution space and the global optimal solution is not easily found; however, according to the experimental results in Table 3, TPSA can converge to the theoretical optimal value 0 on f_{10}, f_{12}, f_{14}, f_{15}, f_{16}, and f_{17} functions. Considering the mean and standard deviation indicators, the robustness of the algorithm was good, and the success rate of finding the optimal value was 100% except f_{18}; PSA, GSA, and FPA could not find the theoretical optimal value, and the success rate was zero. Although PSO found the theoretical optimal values of f_{10}, f_{12}, and f_{16} functions, judging from the mean and standard deviation indicators, its robustness was poor and the success rate of finding the optimal values for f_{10}, f_{12}, and f_{16} was low, at 26%, 38%, and 12%, respectively. Evidently, it could not find the theoretical optimal value of other functions. On the f_{11} function, TPSA outperformed all other comparison algorithms except PSO, which is comparable to TPSA in terms of optimal value. On the f_{13} function, TPSA was inferior to PSO and GSA in terms of optimal value metrics, but better than PSA and FPA; TPSA was better than PSA, PSO, GSA, and FPA in terms of worst value and standard deviation metrics. On the f_{18} function, none of the algorithms could find the theoretical

Table 2: Comparison of experimental results of a high-dimensional unimodal function.

Function	Algorithm	Worst	Best	Mean	SD	SR (%)
f_1	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	5.7995E + 03	1.2498E + 03	2.7206E + 03	1.0170E + 03	0
	PSO	1.0238E + 04	**0**	6.3237E + 02	2.3876E + 03	30
	GSA	6.6294E − 16	4.8668E − 17	1.2535E − 16	8.9575E − 17	**100**
	FPA	3.7164E + 02	4.4779E + 01	1.8784E + 02	7.2688E + 01	0
f_2	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	8.6501E + 18	2.4299E + 02	1.8195E + 17	1.2108E + 18	0
	PSO	5.0133E + 02	1.8937E − 08	1.2650E + 02	1.1368E + 02	0
	GSA	2.0481E + 02	2.7189E + 01	1.2732E + 02	4.4311E + 01	0
	FPA	7.8048E + 24	6.3226E + 14	2.2139E + 23	1.1082E + 24	0
f_3	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	1.8668E + 04	3.1818E + 03	1.0352E + 04	3.1218E + 03	0
	PSO	4.3081E + 04	8.8462E + 02	1.0044E + 04	7.8544E + 03	0
	GSA	7.8187E + 02	1.7901E + 02	4.3920E + 02	1.5134E + 02	0
	FPA	8.1612E + 02	1.5802E + 02	3.3916E + 02	1.3053E + 02	0
f_4	TPSA	**5.2583E − 289**	5.8643E − 303	**1.7908E − 290**	**0**	**100**
	PSA	2.9045E + 01	1.3501E + 01	2.0567E + 01	3.4324E + 00	0
	PSO	1.4003E + 01	**0**	3.8561E + 00	3.4079 E+00	24
	GSA	7.5445E + 00	9.6194E-09	1.2244E + 00	1.4659E + 00	0
	FPA	2.2979E + 01	9.3920 E+00	1.5441E + 01	2.6835E + 00	0
f_5	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	5.8848E + 02	3.2698E + 01	1.7675E + 02	1.2039E + 02	0
	PSO	3.0490E + 03	2.0097E + 00	4.8320E + 02	7.8745E + 02	0
	GSA	2.4491E + 01	7.2369E − 02	3.3335E + 00	4.4974E + 00	0
	FPA	1.0469E + 01	9.5436E − 01	3.5689E + 00	2.3036E + 00	0
f_6	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	3.4823E + 10	2.0340E + 09	1.0593E + 10	6.8734E + 09	0
	PSO	1.3010E + 11	**0**	1.0386E + 10	3.5207E + 10	26
	GSA	5.0482E + 09	4.3220E + 07	9.9831E + 08	9.5587E + 08	0
	FPA	2.4020E + 08	1.0554E + 07	6.1445E + 07	4.2426E + 07	0
f_7	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	1.8910E + 10	3.0492E + 02	2.0857E + 09	3.4464E + 09	0
	PSO	4.2990E + 02	2.3275E + 01	1.2350E + 02	8.4891E + 01	0
	GSA	3.0694E + 02	8.5564E + 01	2.0274E + 02	4.8898E + 01	0
	FPA	5.5954E + 01	1.4387E + 01	3.4120E + 01	9.9402E + 00	0
f_8	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	2.1925E + 08	2.6450E + 07	8.1240E + 07	4.0316E + 07	0
	PSO	6.1954E + 08	2.4364E + 05	7.6950E + 07	1.3218E + 08	0
	GSA	3.2377E + 05	9.3534E + 03	9.9924E + 04	6.7542E + 04	0
	FPA	1.1461 E+06	2.0683 E+05	4.9640 E+05	2.1939E+05	0
f_9	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	1.1378E + 05	3.0631E + 04	5.9152E + 04	1.6057E + 04	0
	PSO	3.5381E + 04	6.9216E + 01	7.9496E + 03	9.0176E + 03	0
	GSA	4.9584E + 03	4.3621E + 01	1.4216E + 03	1.0296E + 03	0
	FPA	9.4585E + 02	1.0100E + 02	3.8616E + 02	1.6617E + 02	0

optimal value of the function, but the optimal accuracies of TPSA and GSA convergence were significantly better than the other algorithms. This indicates that the algorithm is more stable. In conclusion, the performance of TPSA on high-dimensional multimodal functions was significantly better than that of the other algorithms, showing that the improvement strategy of this study is effective.

4.2.3. Type III: Low-Dimensional Function. The global optimum of a low-dimensional function is usually surrounded by many local extrema and the function shows strong oscillation characteristics; the global optimum is not easily found. Therefore, it is often used as a test of the exploration capability of a swarm intelligence algorithm [40]. Table 4 presents the experimental results. With respect to the f_{19} and f_{20} functions, the success rate of TPSA was 100%, and the worst, mean, and standard deviation were better than those of the PSA, PSO, GSA, and FPA algorithms. TPSA was comparable to PSO in the f_{21} function and to FPA in the f_{22} function, outperforming other comparative algorithms. For f_{23} and f_{24} functions, TPSA and PSO achieved the best search performance, and the theoretical optimal value −1 was successfully found in 50 experiments, with a search success

Table 3: Comparison of experimental results of high-dimensional multimodal functions.

Function	Algorithm	Worst	Best	Mean	SD	SR (%)
f_{10}	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	$5.3243E+03$	$1.5101E+03$	$3.1699E+03$	$8.7013E+02$	0
	PSO	$1.0334E+04$	**0**	$1.1450E+03$	$3.0064E+03$	26
	GSA	$1.0248E+02$	$1.4924E+01$	$3.1083E+01$	$1.2654E+01$	0
	FPA	$1.5850E+04$	$3.0588E+03$	$8.6987E+03$	$3.1586E+03$	0
f_{11}	TPSA	**$8.8818E-16$**	**$8.8818E-16$**	**$8.8818E-16$**	**$9.8608E-32$**	**100**
	PSA	$1.2710E+01$	$7.7566E+00$	$1.0667E+01$	$1.1378E+00$	0
	PSO	$1.9963E+01$	**$8.8818E-16$**	$6.6833E+00$	$6.5146E+00$	14
	GSA	$1.3481E-08$	$5.1989E-09$	$8.1720E-09$	$1.8738E-09$	0
	FPA	$1.0798E+01$	$6.2700E+00$	$8.5933E+00$	$1.0135E+00$	0
f_{12}	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	$5.6870E+01$	$1.4115E+01$	$2.6771E+01$	$8.8474E+00$	0
	PSO	$9.1658E+01$	**0**	$1.0034E+01$	$2.7102E+01$	38
	GSA	$2.0670E+01$	$2.8808E+00$	$7.7031E+00$	$3.4148E+00$	0
	FPA	$4.8585E+00$	$1.8950E+00$	$2.9952E+00$	$6.7713E-01$	0
f_{13}	TPSA	**$6.5082E-01$**	$2.5722E-01$	$4.3957E-01$	**$8.5590E-02$**	0
	PSA	$5.7305E+04$	$7.4754E+00$	$1.6894E+03$	$8.2435E+03$	0
	PSO	$1.6549E+01$	$3.4360E-02$	$1.5472E+00$	$2.6489E+00$	0
	GSA	$1.5056E+00$	**$2.4170E-19$**	**$1.7726E-01$**	$3.1821E-01$	**54**
	FPA	$6.5574E+00$	$2.6707E+00$	$4.4358E+00$	$9.5450E-01$	0
f_{14}	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	$2.3751E+14$	$1.6164E+12$	$3.7241E+13$	$4.2866E+13$	0
	PSO	$5.9552E+11$	$2.2880E+07$	$5.6009E+10$	$9.8986E+10$	0
	GSA	$3.4627E+12$	$5.6796E+09$	$6.7758E+11$	$7.4711E+11$	0
	FPA	$4.3402E+11$	$5.9743E+09$	$9.8737E+10$	$9.6611E+10$	0
f_{15}	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	$2.4480E+01$	$1.5150E+01$	$1.8746E+01$	$2.3177E+00$	0
	PSO	$1.7371E+01$	$1.4570E-01$	$7.6387E+00$	$3.7595E+00$	0
	GSA	$4.7486E-01$	$3.8953E-04$	$2.4014E-02$	$9.1282E-02$	0
	FPA	$2.2947E+01$	$1.5719E+01$	$1.8563E+01$	$1.6641E+00$	0
f_{16}	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	$1.9898E+00$	$1.3864E-01$	$1.2006E+00$	$3.6791E-01$	0
	PSO	$1.8408E+00$	**0**	$4.3056E-01$	$5.6643E-01$	12
	GSA	$1.4778E-01$	$4.4409E-16$	$3.2513E-02$	$6.1219E-02$	78
	FPA	$1.6005E+00$	$7.9387E-01$	$1.1614E+00$	$1.8987E-01$	0
f_{17}	TPSA	**0**	**0**	**0**	**0**	**100**
	PSA	$5.4875E+00$	$3.8224E+00$	$4.5343E+00$	$4.1785E-01$	0
	PSO	$4.5043E+00$	$1.4720E-01$	$2.3700E+00$	$1.1092E+00$	0
	GSA	$1.7893E+00$	$1.1465E-02$	$6.0515E-01$	$3.9595E-01$	0
	FPA	$6.5163E+00$	$4.3662E+00$	$5.5227E+00$	$5.4053E-01$	0
f_{18}	TPSA	**$5.0000E-01$**	$4.6881E-01$	$4.9888E-01$	**$4.9713E-03$**	0
	PSA	$5.5137E+03$	$1.4041E+03$	$3.0520E+03$	$8.8917E+02$	0
	PSO	$2.0339E+04$	$4.8126E-01$	$1.6625E+03$	$4.2390E+03$	0
	GSA	$7.7704E-01$	**$3.3108E-01$**	**$4.8888E-01$**	$6.2067E-02$	0
	FPA	$3.9094E+02$	$7.9807E+01$	$2.0618E+02$	$7.3277E+01$	0

rate of 100%. The optimization performance of PSA, GSA, and FPA was poor as they could not find the theoretical optimal value of the function, and the search success rate was low.

4.2.4. Convergence and Stability Analysis. The fitness convergence curve for some functions is shown in Figure 3, and a boxplot of the experimental results is shown in Figure 4. Figures 3 and 4 showed the advantages of the optimization performance of TPSA over the other algorithms. The theoretical optimal values of the selected functions were zero. For the convenience of display and observation, the fitness of the algorithm for all functions is treated with a logarithm base of 10.

Figure 3 shows that TPSA converged to the theoretical optimal value of the function, whereas PSA, PSO, GSA, and FPA could not. Particularly, for f_6, f_{10}, f_{14}, and f_{15} functions, TPSA converged to the theoretical optimal value with only a few iterations. Therefore, its convergence speed has obvious advantages over the other algorithms.

Figure 4 shows that TPSA converged to the theoretical optimal value of the function each time, and its convergence accuracy and stability were better than those of PSA, PSO,

Table 4: Comparison of experimental results of low-dimensional functions.

Function	Algorithm	Worst	Best	Mean	SD	SR (%)
f_{19}	TPSA	**−10.1532**	**−10.1532**	**−10.1532**	**5.2670E - 09**	**100**
	PSA	−1.5165	**−10.1532**	−5.6914	3.5463E + 00	30
	PSO	−5.0552	**−10.1532**	−9.7454	1.3831E + 00	92
	GSA	−0.8890	−5.0552	−4.1730	1.6788E + 00	0
	FPA	−10.1511	**−10.1532**	−10.1531	3.0174E − 04	18
f_{20}	TPSA	**−10.4029**	**−10.4029**	**−10.4029**	**3.2162E − 08**	**100**
	PSA	−2.0021	**−10.4029**	−6.7117	3.5352E + 00	36
	PSO	−0.9098	**−10.4029**	−9.5752	2.1231E + 00	86
	GSA	−0.9081	**−10.4029**	−6.5803	2.9854E + 00	34
	FPA	−10.4000	**−10.4029**	−10.4027	4.9174E − 04	62
f_{21}	TPSA	−2.8066	**−10.5364**	−10.2272	1.5147E + 00	96
	PSA	−1.6610	**−10.5364**	−6.8643	3.5147E + 00	38
	PSO	−5.1285	**−10.5364**	−10.4283	7.5711E − 01	**98**
	GSA	−5.1285	**−10.5364**	−9.9956	1.6224E + 00	90
	FPA	**−10.4971**	**−10.5364**	**−10.5351**	**5.6111E − 03**	32
f_{22}	TPSA	−3.1628	**−3.3220**	−3.3017	5.0245E − 02	86
	PSA	−3.0580	**−3.3220**	−3.2443	8.1362E − 02	40
	PSO	−2.2671	**−3.3220**	−3.1373	2.3962E − 01	40
	GSA	−0.7011	−2.7094	−1.4795	4.6529E − 01	0
	FPA	**−3.2935**	**−3.3220**	**−3.3196**	**4.4393E − 03**	**90**
f_{23}	TPSA	**−1**	**−1**	**−1**	**0**	**100**
	PSA	−0.92181082	−0.99028409	−0.98891462	9.5863E − 03	0
	PSO	**−1**	**−1**	**−1**	**0**	**100**
	GSA	−0.87252750	−0.99865097	−0.97599226	2.6079E − 02	0
	FPA	−0.99028409	−0.99996853	−0.99407718	3.5339E − 03	0
f_{24}	TPSA	**−1**	**−1**	**−1**	**0**	**100**
	PSA	−0.93624533	−0.99999999	−0.96939776	3.1852E − 02	20
	PSO	**−1**	**−1**	**−1**	**0**	**100**
	GSA	−0.95502962	−0.99995018	−0.98787620	1.1911E − 02	0
	FPA	−0.93624533	−0.99999430	−0.99574642	1.0612E − 02	0

GSA, and FPA. The convergence results of PSA showed a low precision and high fluctuation; although PSO converged to the theoretical optimal value on f_1, f_6, and f_{10} functions, the fitness fluctuation was strong, and the algorithm performance was unstable. The fitness fluctuation of GSA and FPA improved; however, neither of them could converge to the theoretical optimal value of the function.

In conclusion, TPSA has significantly improved convergence accuracy, convergence speed, robustness, and stability, compared with the other algorithms and is competitive among intelligent optimization algorithms.

4.3. Wilcoxon Rank-Sum test. As the performance of the algorithm cannot be fully described using only the mean and standard deviation, a statistical validation is required [41]. Therefore, in this study, the Wilcoxon rank-sum test was chosen to verify whether TPSA is significantly different from PSA, PSO, GSA, and FPA at the $p = 5\%$ significance level to determine the level of superiority and inferiority among them. When $p < 5\%$, it is considered that the H0 hypothesis is rejected and there is a significant difference between the two algorithms. Conversely, there is no obvious difference between the two algorithms, and their performance is equivalent.

Table 5 shows the values of the Wilcoxon rank-sum test p of TPSA and other algorithms for all benchmark functions, where “N/A” indicates that the performances of the two algorithms are quite incomparable. TPSA cannot be compared with itself and is marked as “N/A”, whereas “+“, “–,” and “=” indicate that TPSA is superior, inferior, or equivalent to the other algorithms, respectively. The value of p in Table 5 is greater than 0.05 only when TPSA is compared with PSO on the f_{13} function, and with FPA on the f_{22} function, which is marked in bold in the table; it is less than 0.05 in other cases. Table 5 shows that the optimization performance of TPSA was significantly better than that of PSA, PSO, GSA, and FPA, thus verifying the effectiveness of the improved strategy in this study.

4.4. TPSA in Higher-Dimension Performance Analysis. To further verify the stability and convergence accuracy of TPSA in higher dimensions, three functions, f_5, f_{12}, and f_{15}, were selected from two types of high-dimensional unimodal and multimodal test functions for simulation experiments in 500, 1,000, and 2,000 dimensions. The program independently ran 50 times, and the mean and standard deviation were used as evaluation metrics. Table 6 presents the experimental results. It is observed that, with a significant increase in the function dimension, the mean value and standard deviation of fitness of TPSA were 0 and remained unchanged, while the optimized mean value of PSA

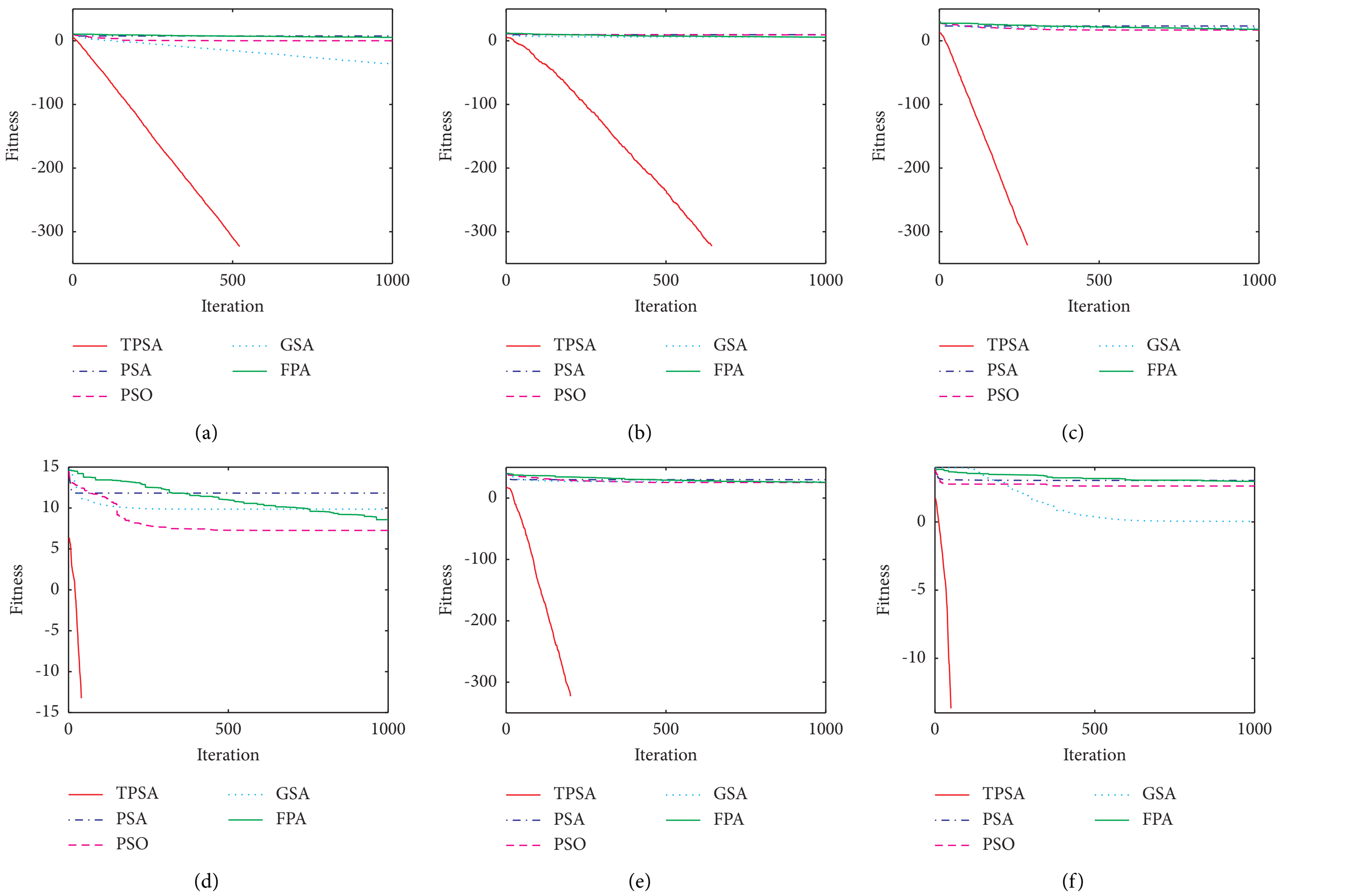

Figure 3: Comparison of fitness convergence curves of five algorithms. (a) f_1. (b) f_3. (c) f_6. (d) f_{10}. (e) f_{14}. (f) f_{15}.

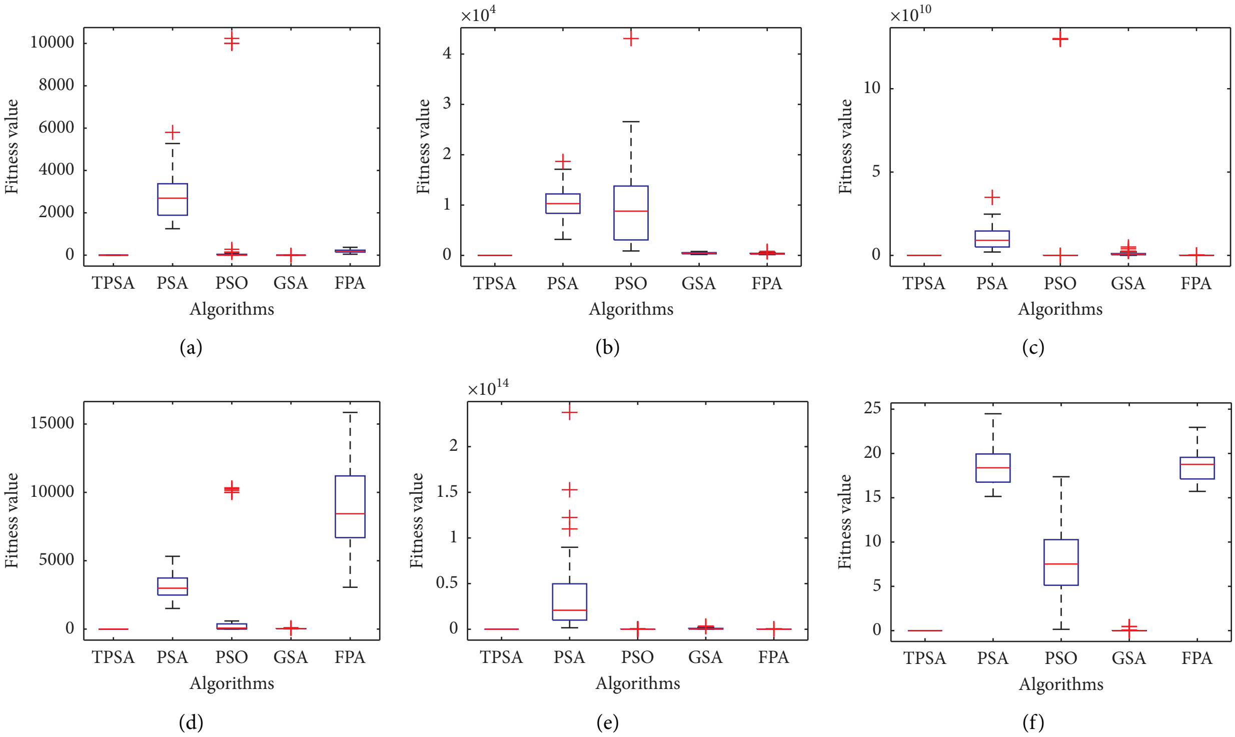

Figure 4: Boxplot of optimal fitness. (a) f_1. (b) f_3. (c) f_6. (d) f_{10}. (e) f_{14}. (f) f_{15}.

Table 5: p values of Wilcoxon rank-sum test on benchmark function.

Function	PSA	PSO	GSA	FPA	TPSA
f_1	3.31E − 20 +	7.24E − 11 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_2	4.03E − 19 +	4.01E − 19 +	4.03E − 19 +	4.03E − 19 +	N/A=
f_3	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_4	7.07E − 18 +	5.62E − 04 +	7.07E − 18 +	7.07E − 18 +	N/A=
f_5	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_6	3.31E − 20 +	7.24E − 11 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_7	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_8	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_9	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_{10}	3.31E − 20 +	2.81E − 11 +	3.30E − 20 +	3.31E − 20 +	N/A=
f_{11}	3.31E − 20 +	2.13E − 14 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_{12}	3.31E − 20 +	5.25E − 13 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_{13}	7.07E − 18 +	**2.13E − 01**=	8.46E − 14 −	7.07E − 18 +	N/A=
f_{14}	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_{15}	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_{16}	3.31E − 20 +	6.95E − 17 +	3.09E − 20 +	3.31E − 20 +	N/A=
f_{17}	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	3.31E − 20 +	N/A=
f_{18}	7.07E − 18 +	7.77E − 04 +	5.60E − 04 −	7.07E − 18 +	N/A=
f_{19}	7.07E − 18 +	2.63E − 14 +	6.54E − 19 +	7.07E − 18 +	N/A=
f_{20}	2.62E − 17 +	1.02E − 10 +	5.29E − 03 +	7.07E − 18 +	N/A=
f_{21}	1.31E-15 +	1.17E − 18 −	2.87E − 13 +	2.29E − 15 +	N/A=
f_{22}	8.08E − 06 +	3.29E-07 +	2.75E-19 +	**4.13E-01**=	N/A=
f_{23}	1.85E − 21 +	N/A=	3.31E − 20 +	3.31E − 20 +	N/A=
f_{24}	3.25E − 20 +	N/A=	3.31E − 20 +	3.31E − 20 +	N/A=
+/=/−	24/0/0	20/3/1	22/0/2	23/1/0	0/24/0

Table 6: Comparison of experimental results for higher dimensions.

Function	Algorithm	$d = 500$		$d = 1,000$		$d = 2,000$	
		Mean	SD	Mean	SD	Mean	SD
f_5	TPSA	0	0	0	0	0	0
	PSA	1.773E + 04	1.731E + 03	4.025E + 04	2.399E + 03	8.885E + 04	4.129E + 03
f_{12}	TPSA	0	0	0	0	0	0
	PSA	1.285E + 03	6.565E + 01	2.796E + 03	7.661E + 01	5.898E + 03	1.481E + 02
f_{15}	TPSA	0	0	0	0	0	0
	PSA	4.625E + 02	1.081E + 01	9.719E + 02	1.567E + 01	1.993E + 03	1.908E + 01

increased significantly. This shows that TPSA did not fall into a "dimension disaster" with an increase in dimensions, which reduces the optimization performance. Compared with PSA, TPSA had a stronger ability to jump out of the local extreme value to maintain a good stability and could converge to the theoretical extreme value in higher dimensions. This further demonstrates the effectiveness of the t-distributed elite mutation strategy proposed in this study.

4.5. Strategy Effectiveness Analysis. PSA using only the t-distribution mutation strategy, optimal *Porcellio scaber* retention strategy, and dynamic adaptive weight allocation strategy were denoted as TMPSA, ORSPSA, and DAPSA, respectively. PSA was compared experimentally with the three single-strategy improved algorithms described above, and the results are listed in Table 7. As seen in Table 7, for unimodal and multimodal functions of $f_1, f_3, f_5, f_7, f_9, f_{11}, f_{13}$, f_{15}, and f_{17}, the three strategies improved PSA performance overall, in which the t-distribution mutation strategy played a key role. For the functions f_3, f_9, and f_{13}, TPSA obtained the optimal solution with the integrated action of the three strategies. For the fixed-dimensional multimodal functions f_{19}–f_{22}, the optimization results of ORSPSA and DAPSA were comparable to PSA, but TMPSA performed worse. However, as shown in Table 4, with the integrated action of the three strategies, TPSA had 100, 100, 96, and 86% of the search success of them, respectively. The optimization performance was substantially improved compared with same of PSA; therefore, the strategy proposed in this paper is effective.

5. Engineering Applications of TPSA

To further study the performance of TPSA, it was applied to solve three practical engineering problems—three-bar truss, welded-beam, and tension/compression-spring designs—and the results were compared with those reported in the literature. These engineering application problems are multiconstraint optimization problems; however, PSA

Table 7: Comparison of experimental results of single-strategy improvement algorithms.

Function	Index	Algorithm			
		TMPSA	ORSPSA	DAPSA	PSA
f_1	Mean	0	$2.7617E+03$	$1.9585E+03$	$3.1699E+03$
	SD	0	$7.9656E+02$	$6.5230E+02$	$8.7013E+02$
f_3	Mean	$3.2077E-46$	$1.0321E+04$	$1.0779E+04$	$1.0352E+04$
	SD	$1.7274E-45$	$2.9644E+03$	$4.2197E+03$	$3.1218E+03$
f_5	Mean	0	$1.6753E+02$	$1.2979E+02$	$1.7675E+02$
	SD	0	$1.2045E+02$	$8.1983E+01$	$1.2039E+02$
f_7	Mean	0	$1.6270E+09$	$1.9854E+09$	$2.0857E+09$
	SD	0	$2.0596E+09$	$3.5314E+09$	$3.4464E+09$
f_9	Mean	$2.8676E-157$	$6.8316E+04$	$5.8668E+04$	$5.9152E+04$
	SD	0	$1.8130E+04$	$1.5567E+04$	$1.6057E+04$
f_{11}	Mean	$8.8818E-16$	$1.0599E+01$	$8.9165E+00$	$1.0667E+01$
	SD	$9.8608E-32$	$9.6940E-01$	$1.1473E+00$	$1.1378E+00$
f_{13}	Mean	$1.2186E+00$	$9.1168E+02$	$1.8062E+01$	$1.6894E+03$
	SD	$1.8625E-01$	$3.1424E+03$	$9.9452E+00$	$8.2435E+03$
f_{15}	Mean	0	$1.9208E+01$	$1.7209E+01$	$1.8746E+01$
	SD	0	$1.6213E+00$	$2.3737E+00$	$2.3177E+00$
f_{17}	Mean	0	$4.6834\,E$+00	$4.1558E+00$	$4.5343E+00$
	SD	0	5.1606E-01	$5.8120E-01$	$4.1785E-01$
f_{19}	Mean	−1.9358	−6.7180	−4.8770	−5.6914
	SD	$1.9013E+00$	$3.5153E+00$	$2.8937E+00$	$3.5463E+00$
f_{20}	Mean	−2.3798	−5.8143	−7.1833	−6.7117
	SD	$2.1818E+00$	$3.2631E+00$	$3.5881E+00$	$3.5352E+00$
f_{21}	Mean	−2.3632	−5.8501	−5.5687	−6.8643
	SD	$1.8139E+00$	$3.4774E+00$	$3.5058E+00$	$3.5147E+00$
f_{22}	Mean	−2.2669	−3.2684	−3.2610	−3.2443
	SD	$5.5717E-01$	$6.7457E-02$	$7.9217E-02$	$8.1362E-02$

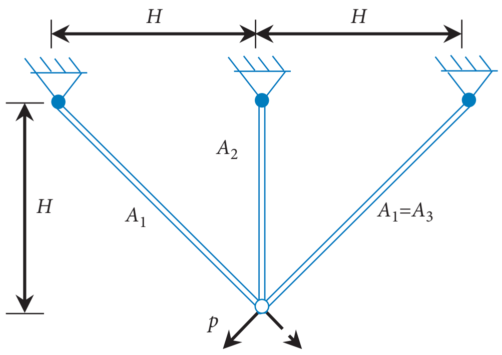

Figure 5: Three-bar truss design.

cannot solve constrained optimization problems. This study uses the penalty function in the literature [15] to transform constrained optimization into unconstrained optimization for the solution.

5.1. Three-Bar Truss Design Problem. Figure 5 illustrates the three-bar truss structure. The height of the truss is H; the cross-sectional areas of each bar are $A_{1,}$ A_2, and A_3; and the concentrated load is p. The problem is solved to minimize the volume of a three-bar truss under stress constraints, deflection, and buckling. This problem requires the optimization of two variables (A_1 and A_2) to adjust the cross-sectional area of each bar. The problem is mathematically described as follows:

$$\begin{aligned}
&\text{Minimize}\, f(x) = \left(2\sqrt{2}\,x_1 + x_2\right) \times l,\\
&\text{subject to}\, g_1(x) = \left(\left(\sqrt{2}\,x_1 + x_2\right)/\left(\sqrt{2}\,x_1^2 + 2x_1x_2\right)\right)p - \sigma \le 0,\\
&\qquad g_2(x) = \left(x_2/\left(\sqrt{2}\,x_1^2 + 2x_1x_2\right)\right)p - \sigma \le 0,\ \text{and}\ g_3(x)\\
&\qquad\quad = \left(1/\left(\sqrt{2}\,x_1 + x_2\right)\right)p - \sigma \le 0,
\end{aligned} \tag{8}$$

where $0 \le x_1, x_2 \le 1$, $l = 100\text{cm}$, $p = 2\text{kN/cm}^2$, and $\sigma = 2\text{kN/cm}^2$.

The population size was set to 50 and the maximum iteration to 1,000. The optimal solutions of TPSA and PSA were compared with the optimal solution of the algorithm reported in the relevant literature. The results are listed in Table 8, which shows that the optimal solution solved by TPSA is superior to PSA, CS [42], HHO [43], SC-GWO [44], and GLF-GWO [45] and equivalent to the results of AEO [46]. This shows that TPSA can optimize the design of the three-bar truss with a good effect.

5.2. Welded-Beam Design Problem. The welded-beam structure is shown in Figure 6 with four design parameters: weld thickness (h), length of the clamped bar (l), height of

Table 8: Comparison of optimal solutions for three-bar truss design problem.

Algorithms	$x_1(A_1)$	$x_2(A_2)$	$f(x)$
TPSA	0.7886705	0.4082615	263.895843
PSA	0.7912275	0.4011807	263.9110069
CS	0.78867	0.40902	263.9716
HHO	0.7886628	0.4082831	263.8958434
SC-GWO	0.78941	0.40617	263.8963
GLF-GWO	0.788174	0.4096753	263.8969
AEO	0.788707	0.408159	263.895843

the bar (t), and thickness of the bar (b). The optimal design should minimize the construction cost under the constraints of shear stress τ, bending stress σ, buckling load P_c, and deflection δ. The mathematical description of the problem is as follows:

$$\begin{aligned}
\text{Minimize}\, f(x) &= 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),\\
\text{Subject to}\, g_1(x) &= \tau(x) - 13600 \le 0,\\
g_2(x) &= \sigma(x) - 30000 \le 0,\\
g_3(x) &= x_1 - x_4 \le 0,\\
g_4(x) &= 0.10471x_1^2\\
&\quad + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,\\
g_5(x) &= 0.125 - x_1 \le 0,\\
g_6(x) &= \delta(x) - 0.25 \le 0, \text{ and}\\
g_7(x) &= p - p_c(x) \le 0,
\end{aligned} \tag{9}$$

where $\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''(x_2/(2R)) + (\tau'')^2}$, $\tau' = (p/(\sqrt{2}\ x_1x_2)$, $\tau'' = (QR/J)$, $Q = p(L + (x_2/2))$,

$$\begin{aligned}
R &= \sqrt{x_2^2/4 + ((x_1 + x_3)/2)^2},\\
\sigma(x) &= 6pL/\left(x_4x_3^2\right),\\
J &= 2\left\{\sqrt{2}\,x_1x_2\left[\left(x_2^2/12\right) + ((x_1 + x_3)/2)^2\right]\right\},\\
\delta(x) &= 4pL^3/\left(Ex_3^3x_4\right), \text{ and}\\
p_c(x) &= \left(4.013E\sqrt{\left(x_3^2x_4^6/36\right)}/L^2\right)\left(1 - (x_3/2L)\sqrt{E/(4G)}\right),
\end{aligned} \tag{10}$$

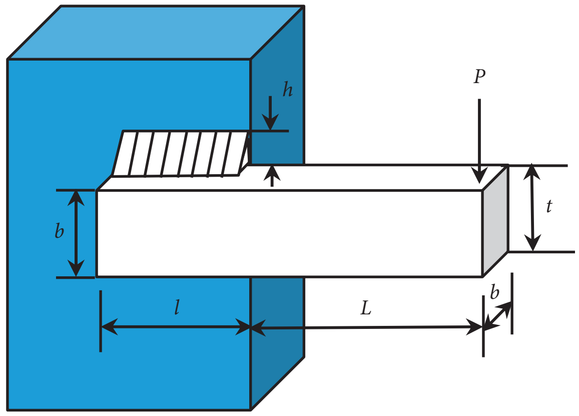

Figure 6: Welded-beam design.

where $p = 6000$, $L = 14$, $E = 30 \times 10^6$, $L = 14$, and $G = 12 \times 10^6$.

Variable range is $01 \le x_1, x_2 \le 2$, $0.1 \le x_3, x_4 \le 10$.

The population size was set to 50; the maximum number of iterations was 10,000, the step-size scaling factor γ was 2; and the program independently ran 30 times. TPSA and PSA were used to solve the welded-beam design problem and the results were compared with those of the algorithms reported in the relevant literature. A comparison of the optimal solution and statistical results are presented in Tables 9 and 10, respectively. It is observed that the optimal solution by TPSA is superior to PSA, CPSO [47], IGMM [48], TEO [49], IGWO [50], SFOA [51], CS-BSA [52], and WAROA [53] and is equivalent to the results of AEO [46]. However, the mean and standard deviation of TPSA are better than those of AEO. This indicates that TPSA is better than the others in solving welded-beam design problems.

5.3. *Tension/Compression-Spring Design Problem.*

Figure 7 shows the structure of the tension/compression spring with three design parameters: average coil diameter (D), number of active coils (p), and wire diameter (d). The design goal was to minimize the weight under certain constraints. The mathematical description of the problem is as follows:

$$\begin{aligned}
\text{Minimize}\, f(x) &= (x_3 + 2)x_2x_1^2,\\
\text{Subject to}\, g_1(x) &= 1 - \left(x_2^3x_3/\left(71785x_1^4\right)\right) \le 0,\\
g_2(x) &= \left(4x_2^2 - x_1x_2\right)/\left(12566\left(x_2x_1^3 - x_1^4\right)\right) + 1/\left(5108x_1^2\right) \le 0\\
g_3(x) &= 1 - \left(140.45x_1/\left(x_2^2x_3\right)\right) \le 0, \text{ and}\\
g_4(x) &= (x_1 + x_2)/1.5 - 1 \le 0,
\end{aligned} \tag{11}$$

where $.05 \le x_1 \le 2.00$, $0.25 \le x_2 \le 1.30$, and $2.00 \le x_3 \le 15.00$.

The population size was set to 50; the maximum number of iterations was 1,000; the step-size scaling

Table 9: Comparison of optimal solutions for welded-beam design problem.

Algorithm	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	$f(x)$
TPSA	0.2057292	3.4704977	9.0366239	0.2057296	1.7248529
PSA	0.240753	3.068655	8.694284	0.240753	1.915351
CPSO	0.202369	3.544214	9.048210	0.205723	1.728024
IGMM	0.205729	3.470496	9.036625	0.205730	1.724855
TEO	0.205681	3.472305	9.035133	0.205796	1.725284
IGWO	0.20571	3.4714	9.0369	0.20573	1.7250
SFOA	0.205728	3.47040	9.03694	0.205728	1.72488
CS-BSA	0.20573	3.4705	9.0366	0.20573	1.7249
WAROA	0.20573	3.4705	9.0366	0.20573	1.7249
AEO	0.2057296	3.4704886	9.0366239	0.2057296	1.724852

Table 10: Comparison of statistical results for welded-beam design problem.

Algorithm	Best	Worst	Mean	SD
TPSA	1.7248529	1.7257362	1.7249917	$1.9533E{-}04$
PSA	1.915351	4.743892	3.183695	0.700671
CPSO	1.728024	1.782143	1.748831	0.012926
IGMM	1.7248552	1.74769	1.732152	$7.14E{-}03$
TEO	1.725284	1.931161	1.768040	0.0581661
IGWO	1.7250	1.7263	1.7255	$4.72E{-}04$
SFOA	1.72488	1.73930	1.72874	$3.71610E{-}03$
CS-BSA	1.7249	1.7249	1.7249	$4.44089E{-}16$
WAROA	1.7249	1.7252	1.7249	—
AEO	1.724852	1.7255664	1.7250057	$2.4763E{-}04$

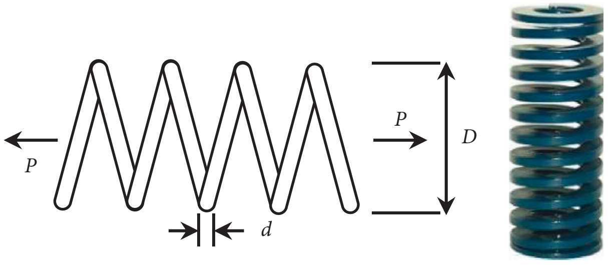

Figure 7: Tension/compression-spring design.

Table 11: Comparison of optimal solutions for tension/compression-spring design problem.

Algorithm	$x_1(d)$	$x_2(D)$	$x_3(p)$	$f(x)$
TPSA	0.0516851	0.3566230	11.2945227	0.0126652
PSA	0.0556512	0.459735	7.086870	0.0129381
HMPA	0.051608	0.354788	11.40295	0.01266535
AEO	0.051897	0.361751	10.879842	0.0126662
CPSO	0.051728	0.357644	11.244543	0.0126747
IGWO	0.051701	0.356983	11.2756	0.012667
SFOA	0.051794	0.359257	11.1417	0.0126654
PO	0.05248	0.37594	10.24509	0.01267
TEO	0.051775	0.3587919	11.168390	0.012665
WAROA	0.051676	0.35640	11.308	0.012665

Table 12: Comparison of statistical results for tension/compression-spring design problem.

Algorithm	Best	Worst	Mean	SD
TPSA	0.0126652	0.0127196	0.0126856	1.9589 *E*-05
PSA	0.0129381	8.7917*E* + 14	9.2191*E* + 12	8.7488*E* + 13
HMPA	0.01266535	0.01268822	0.01267299	6.6537*E* − 06
AEO	0.0126662	0.0127271	0.01272712	2.5401*E* − 05
CPSO	0.0126747	0.012924	0.012730	5.1985*E* − 05
IGWO	0.012667	0.012718	0.012691	1.97*E* − 05
SFOA	0.0126654	0.0130107	0.0127312	6.76788*E* − 05
PO	0.0127	0.0128	0.0127	—
TEO	0.012665	0.012715	0.012685	4.4079*E* − 06
WAROA	0.012665	0.012821	0.012693	—

factor γ was 5; and the program ran independently for 30 times. TPSA and PSA were used to solve the tension/compression-spring design problem and the results were compared with those of the algorithms reported in the relevant literature. A comparison of the optimal solution and statistical results are shown in Tables 11 and 12, respectively. Tables 11 and 12 show that the optimal solution of TPSA is superior to PSA, HMPA [4], AEO [46], CPSO [47], IGWO [50], SFOA [51], and PO [54] and is equivalent to the results of the TEO [49] and WAROA [53]. However, the worst and average values of TPSA are better than those of WAROA. This indicates that TPSA can effectively solve a tension/compression-spring design problem.

Table 12 shows that the optimal solution of PSA is inferior to all other algorithms and its average value and standard deviation are relatively large. This indicates that PSA is unstable in solving this problem. TPSA results are superior to those of the algorithms reported in the literature, which further shows that the improved strategy in this study effectively improves the performance of PSA.

TPSA mutates the elite *Porcellio scaber* using the t-distribution operation. The degrees of freedom of the t-distribution are the number of iterations of the algorithm, and their values dynamically increase as the algorithm runs. Thus, its morphology changes dynamically from a Cauchy to Gaussian distribution. In the early stage of the algorithm, the dynamically changing characteristics maintain the diversity of the population to a large extent so that the population can sufficiently explore the search space. In the late stage of algorithm operation, the t-distribution mainly presents Gaussian distribution characteristics, and the mutated population implements fine exploitation in a smaller space around the optimal *Porcellio scaber*, which improves the convergence accuracy while avoiding the algorithm from sinking into a local optimum. The optimal *Porcellio scaber* retention strategy in TPSA partially overcomes PSA's disadvantage of losing the optimal solution of the previous generation. TPSA's dynamic adaptive improvement of weight distribution parameters improves the balance of exploration and exploitation and enhances performance. In summary, the TPSA assembled with these strategies has good performance; therefore, good optimal solutions are obtained when solving the three engineering problems.

6. Conclusions

PSA is a new swarm intelligence optimization algorithm, which has the disadvantages of low convergence accuracy and easy premature convergence. To address these issues, an improved PSA (TPSA), based on the t-distribution elite mutation mechanism, is proposed. First, using t-distribution for each generation of elite *Porcellio scaber* for a dimension-by-dimension mutation, the characteristics of t-distribution are used to maintain population diversity, thus enhancing the algorithm's ability to explore and exploit the global and local space. Secondly, the judgment mechanism between the contemporary optimal solution and the previous generation optimal solution is added to the algorithm process to address the shortcoming that the basic PSA may lead to the loss of information of the previous generation of elite *Porcellio scaber*. Finally, dynamic adaptive improvements are made to the weight assignment parameters of PSA to achieve a balance between exploitation and exploration and improve its performance in finding the best solution.

The performance and practicality of TPSA were evaluated on 24 benchmark functions and applied to three real-world engineering problems. First, the convergence accuracy, convergence speed, and stability of TPSA were evaluated on 24 benchmark functions, including high-dimensional unimodal, high-dimensional multimodal, and low-dimensional functions. The results show that TPSA has significant advantages over basic PSA, PSO, GSA, and FPA. This was also proved by the Wilcoxon rank-sum test on the experimental results. To further validate the performance of TPSA, experiments were conducted in 500, 1,000, and 2,000 dimensions on some functions, and the results show that TPSA converged to the theoretical optimum of the function without falling into "dimension disaster," which further demonstrated its good convergence and stability. In the three practical engineering problems of three-bar truss, welded-beam, and tension/compression-spring designs, the optimal solution of TPSA was better than those of the algorithms reported in related literature, which verified its practicality. However, it should also be noted that TPSA is inferior to PSO and GSA in optimizing some functions. Therefore, in the future, we will continue to improve the optimization performance of TPSA and apply it to more complex engineering optimization problems.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Key Project of the Excellent Youth Talent Support Program in Higher Education Institutions of Anhui Province (Grant no. gxyqZD2019125), the Key Project of the Natural Science Foundation in Higher Education Institutions of Anhui Province (Grants nos. KJ2020A0035 and KJ2017A843), and the Project of Provincial Quality Engineering in Higher Education Institutions of Anhui Province (Grants nos. 2020kfkc577 and 2020jyxm2226).

References

[1] M. Braik, A. Hammouri, J. Atwan, M. A. Al-Betar, and M. A. Awadallah, “White Shark Optimizer: a novel bio-inspired meta-heuristic algorithm for global optimization problems,” *Knowledge-Based Systems*, vol. 243, Article ID 108457, 2022.

[2] W. K. Mashwani, R. Haider, and S. Brahim Belhaouari, “A multiswarm intelligence algorithm for expensive bound constrained optimization problems,” *Complexity*, vol. 2021, Article ID 5521951, 18 pages, 2021.

[3] J. R. Rohan, C. N. Prasad, and P. Sadiq, “An Introduction to the Collective Behaviour of Swarm intelligence,” in *Proceedings of the National Conference on Contemporary Research and Innovations in Computer Science*, St. Joseph’s Evening College, Bangalore, India, December 2017.

[4] S. Barshandeh, F. Piri, and S. R. Sangani, “HMPA: An Innovative Hybrid Multi-Population Algorithm Based on Artificial Ecosystem-Based and Harris Hawks Optimization Algorithms for Engineering problems,” *Engineering with Computers*, vol. 38, 2020.

[5] W. K. Mashwani, A. Hamdi, M. Asif Jan, A. Göktaş, and F. Khan, “Large-scale global optimization based on hybrid swarm intelligence algorithm,” *Journal of Intelligent and Fuzzy Systems*, vol. 39, no. 1, pp. 1257–1275, 2020.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: nsga-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] Y. H. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *Proceedings of the IEEE Conference on Evolutionary Computation*, Anchorage, AK, USA, May 1998.

[8] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[9] M. A. Awadallah, M. A. Al-Betar, I. A. Doush et al., “CCSA: cellular Crow Search Algorithm with topological neighbourhood shapes for optimization,” *Expert Systems with Applications*, vol. 194, Article ID 116431, 2022.

[10] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[11] S. Mirjalili, “SCA: a Sine Cosine Algorithm for solving optimization problems,” *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[12] V. Hayyolalam and A. A. Pourhaji Kazem, “Black Widow Optimization Algorithm: a novel meta-heuristic approach for solving engineering optimization problems,” *Engineering Applications of Artificial Intelligence*, vol. 87, Article ID 103249, 2020.

[13] X.-S. Yang, M. Karamanoglu, and X. He, “Flower pollination algorithm: a novel approach for multiobjective optimization,” *Engineering Optimization*, vol. 46, no. 9, pp. 1222–1237, 2014.

[14] S. Mirjalili and A. Lewis, “The whale optimization algorithm,” *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[15] J. Xue and B. Shen, “A novel swarm intelligence optimization approach: sparrow search algorithm,” *Systems Science & Control Engineering*, vol. 8, no. 1, pp. 22–34, 2020.

[16] M. A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah, and I. Abu Doush, “Coronavirus herd immunity optimizer (CHIO),” *Neural Computing & Applications*, vol. 33, no. 10, pp. 5011–5042, 2021.

[17] M. S. Braik, “Chameleon Swarm Algorithm: a bio-inspired optimizer for solving engineering design problems,” *Expert Systems with Applications*, vol. 174, Article ID 114685, 2021.

[18] W. K. Mashwani, S. N. A. Shah, S. B. Belhaouari, and A. Hamdi, “Ameliorated ensemble strategy-based evolutionary algorithm with dynamic resources allocations,” *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 412–437, 2021.

[19] W. K. Mashwani, A. Khan, A. Göktaş, Y. A. Unvan, O. Yaniay, and A. Hamdi, “Hybrid differential evolutionary strawberry algorithm for real-parameter optimization problems,” *Communications in Statistics - Theory and Methods*, vol. 50, no. 7, pp. 1685–1698, 2021.

[20] Y. Shen, M. Liu, J. Yang, Y. Shi, and M. Middendorf, “A hybrid swarm intelligence algorithm for vehicle routing problem with time windows,” *IEEE Access*, vol. 8, Article ID 93882, 2020.

[21] W. K. Mashwani, A. Salhi, O. Yeniay, M. A. Jan, and R. A. Khanum, “Hybrid adaptive evolutionary algorithm based on decomposition,” *Applied Soft Computing*, vol. 57, pp. 363–378, 2017.

[22] C. Ouyang, Y. Qiu, and D. Zhu, “Adaptive Spiral Flying Sparrow Search algorithm,” *Scientific Programming*, vol. 2021, Article ID 6505253, 16 pages, 2021.

[23] H. S. Alhadawi, M. A. Majid, D. Lambić, and M. Ahmad, “A novel method of S-box design based on discrete chaotic maps and cuckoo search algorithm,” *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 7333–7350, 2021.

[24] M. S. Turgut, H. M. Sağban, O. E. Turgut, and Ö. T. Özmen, “Whale optimization and sine-cosine optimization algorithms with cellular topology for parameter identification of chaotic systems and Schottky barrier diode models,” *Soft Computing*, vol. 25, no. 2, pp. 1365–1409, 2021.

[25] C. Qu, Z. Zeng, J. Dai, Z. J. Yi, and W. He, “A modified sine-cosine algorithm based on neighborhood search and greedy levy mutation,” *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 4231647, 19 pages, 2018.

[26] L. Sun, S. Chen, J. Xu, and Y. Tian, “Improved monarch butterfly optimization algorithm based on opposition-based learning and random local perturbation,” *Journal of Complexity*, vol. 2019, Article ID 4182148, 20 pages, 2019.

[27] W. L. Liu, Y. J. Gong, W. N. Chen, Z. Q. Liu, H. Wang, and J. Zhang, “Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2019.

[28] M. Alweshah, S. Alkhalaileh, M. A. Al-Betar, and A. A. Bakar, “Coronavirus herd immunity optimizer with greedy crossover for feature selection in medical diagnosis,” *Knowledge-Based Systems*, vol. 235, Article ID 107629, 2022.

[29] F. Q. Zhao, R. Ma, and L. Wang, “A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system,” *IEEE Transactions on Cybernetics*, Article ID 3086181, 2021.

[30] F. Zhao, X. He, and L. Wang, “A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of No-wait flow-shop problem,” *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5291–5303, 2021.

[31] A. M. Jubair, R. Hassan, A. H. M. Aman, and H. Sallehudin, “Social class particle swarm optimization for variable-length Wireless Sensor Network Deployment,” *Applied Soft Computing*, vol. 113, no. B, Article ID 107926, 2021.

[32] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, “A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times,” *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2021.

[33] Y. Y. Zhang, P. Zhang, and S. Li, “P. S. A.: A novel optimization algorithm based on survival rules of *Porcellio scaber*,” in *Proceedings of the IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, March 2021.

[34] H. E. Keshta, A. A. Ali, O. P. Malik, E. M. Saied, and F. M. Bendary, “Adaptive control strategy for improved dynamic performance of two islanded inter-connected micro-grids,” *International Journal of Electrical Power & Energy Systems*, vol. 126, Article ID 106562, 2021.

[35] H. E. Keshta, O. P. Malik, E. M. Saied, F. M. Bendary, and A. A. Ali, “Energy management system for two islanded interconnected micro-grids using advanced evolutionary algorithms,” *Electric Power Systems Research*, vol. 192, Article ID 106958, 2021.

[36] Y. Y. Zhang, S. Li, and H. Guo, “*Porcellio scaber* Algorithm (PSA) for Solving Constrained Optimization Problems,” 2017, https://arxiv.org/abs/1710.04036.

[37] P. Broly, J.-L. Deneubourg, and C. Devigne, “Benefits of aggregation in woodlice: a factor in the terrestrialization process?” *Insectes Sociaux*, vol. 60, no. 4, pp. 419–435, 2013.

[38] S. Cividini and G. Montesanto, “Aggregative behavior and intraspecific communication mediated by substrate-borne vibrations in terrestrial arthropods: an exploratory study in two species of woodlice,” *Behavioural Processes*, vol. 157, pp. 422–430, 2018.

[39] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: a gravitational search algorithm,” *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[40] H. H. Xiao, C. X. Wan, Y. M. Duan, and Q. L. Tan, “Flower pollination algorithm based on gravity search mechanism,” *Acta Automatica Sinica*, vol. 43, no. 4, pp. 576–594, 2017.

[41] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[42] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, “Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems,” *Engineering Computers*, vol. 29, no. 1, pp. 17–35, 2013.

[43] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, “Harris hawks optimization: algorithm and applications,” *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[44] S. Gupta, K. Deep, H. Moayedi, L. K. Foong, and A. Assad, “Sine cosine grey wolf optimizer to solve engineering design problems,” *Engineering with Computers*, vol. 37, no. 4, pp. 3123–3149, 2021.

[45] S. Gupta and K. Deep, “Enhanced leadership-inspired grey wolf optimizer for global optimization problems,” *Engineering with Computers*, vol. 36, no. 4, pp. 1777–1800, 2020.

[46] W. Zhao, L. Wang, and Z. Zhang, “Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm,” *Neural Computing & Applications*, vol. 32, no. 13, pp. 9383–9425, 2020.

[47] Q. He and L. Wang, “An effective co-evolutionary particle swarm optimization for constrained engineering design problems,” *Engineering Applications of Artificial Intelligence*, vol. 20, no. 1, pp. 89–99, 2007.

[48] H. Varaee and M. R. Ghasemi, “Engineering optimization based on ideal gas molecular movement algorithm,” *Engineering with Computers*, vol. 33, no. 1, pp. 71–93, 2016.

[49] A. Kaveh and A. Dadras, “A novel meta-heuristic optimization algorithm: thermal exchange optimization,” *Advances in Engineering Software*, vol. 110, pp. 69–84, 2017.

[50] W. Long, J. Jiao, X. Liang, and M. Tang, “Inspired grey wolf optimizer for solving large-scale function optimization problems,” *Applied Mathematical Modelling*, vol. 60, pp. 112–126, 2018.

[51] D. Sattar and R. Salim, “A smart metaheuristic algorithm for solving engineering problems,” *Engineering with Computers*, vol. 37, no. 3, pp. 2389–2417, 2021.

[52] A. Tsipianitis and Y. Tsompanakis, “Improved Cuckoo Search algorithmic variants for constrained nonlinear optimization,” *Advances in Engineering Software*, vol. 149, Article ID 102865, 2020.

[53] R. Jiang, M. Yang, S. Wang, and T. Chao, “An improved whale optimization algorithm with armed force program and strategic adjustment,” *Applied Mathematical Modelling*, vol. 81, pp. 603–623, 2020.

[54] Q. Askari, I. Younas, and M. Saeed, “Political Optimizer: a novel socio-inspired meta-heuristic for global optimization,” *Knowledge-Based Systems*, vol. 195, Article ID 105709, 2020.