Advances in Computer Science Research, volume 87
3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)

# An Advanced Particle Swarm Optimization Method based on T-Distribution Random Process

Tianjia Zhang [a], Yongsheng Yang [b, *]
Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai 200240, China
[a] datazhg@163.com, [b, *] ysyang@sjtu.edu.cn

**Abstract.** There are many real-life engineering problems that entail appropriate optimization methods. Although almost all the problems can be modeled into simple forms described by mathematical formula, it is hard to solve all the decisive problems by a single optimization method. Researchers have developed many effective optimization techniques to solve assorted problems. Among these particle swarm optimization (PSO) has played an important role in optimization of complex and high-dimensional problems. However, PSO suffers from premature convergence and low precision. For this purpose, the paper proposed a TPSO which adapts a stochastic process based on t-distribution and a mechanism of reference set. Subsequently simulations tested on 13 classical benchmark functions demonstrated that the TPSO can achieve faster convergence speed and higher accuracy. Finally, the application on the path planning problem of UAV evaluated the efficiency of the proposed algorithm.

**Keywords:** optimization; PSO; t-distribution; reference set; benchmark functions.

## 1. Introduction

Nowadays many engineering problems can be concluded as an effort for gaining maximum benefit with limited resources. Moreover, it is desired to make the optimal decision for many real-life affairs. In fact, all these decisive problems can be modeled into mathematical problems and consequently be solved by classical optimization method [1]. In a formal way, an optimization problem is defined as maximizing or minimizing a real function by systematically choosing input values from within an allowed set. All the optimization problems can be formulated as following forms:

$$\begin{aligned} \min \quad & f(x) \\ & c(x) = 0 \\ & h(x) \le 0 \end{aligned} \tag{1}$$

Where $f(x)$ is the objective function to be optimized and $c(x)$ and $h(x)$ are equality constraints and inequality constraints respectively Although the mathematical model of an optimization problem is simple, it is sometimes too difficult to solve due to complexity and indeterminacy. The mathematical expressions of the objective functions which usually contain non-linear terms, non-convex terms and multiple constraints can be extremely complicated. Another challenge is that not all design variables are continuous, and some variables can only take certain discrete values[2]. Moreover, there are many problems that do not have analytic expressions and thus can only be regarded as black-box problems.

Researchers have developed many effective optimization techniques to solve assorted problems in spite of so many challenges. The responsibility of choosing the algorithm that is appropriate for a specific application often falls on the user. This choice is an important one, as it may determine whether the problem is solved rapidly or slowly and, indeed, whether the solution is found at all[3]. In general, optimization algorithms can be classified into two main groups according to their principles: classical methods and stochastic algorithms.

Classical methods are almost deterministic methods and can be repeatable that have some fixed steps to follow[1]. They are quite efficient in solving problems which have analytical solutions and finding local optima. According to the convex optimization, classical methods are divided into

derivative-based methods and derivative-free methods. Derivative-based methods make use of gradients and usually have fast convergence speed. Frequently-used derivative-based methods include gradient descent method, trust-region method, conjugate gradient method, Newton method, et al. Since derivatives of many problems are not available, derivative-free optimization (DFO) algorithms differ in the way they use the sampled function values to determine the new iterate.

With the increased complexity and uncertainty, the classical methods sometimes fail to solve large-scale and computational problems[4]because there is a risk to be trapped into local optima. A common practice is to introduce some stochastic components to an algorithm so that it becomes possible to jump out of such locality[5]. Most stochastic algorithms can be considered as metaheuristic.

Biology is a principal source of inspiration to propose new metaheuristic optimization methods. Two famous bio-inspired algorithms are swarm intelligence (SI)-based, and evolutionary algorithms. SI-based algorithms simulate the collective behaviors of biotic population. In social colonies, the individuals perform simple functions while the whole swarm exhibits intelligent behaviors through members' interactions with themselves and with the environment[6]. Evolutionary algorithms are inspired from natural evolution and emulate the biological operators in the genetic field named crossover, mutation, and natural selection. The examples are genetic algorithm (GA) [7,8] and differential evolution algorithm (DE) [9].

Among these numerous types of meta-heuristic algorithms, PSO has shown great advantage of solving complex problems due to its simple structure and less parameters. Up to now, PSO has been successfully applied in several areas, such as pattern clustering[10], crew scheduling problems[11], multi-robot path planning[12], quality control[13], network reliability[14], inventory routing problems[15], time series forecasting[16], constrained shortest path problems[17], layer-packing problems[18], cost-sensitive attribute reduction[19].

However, PSO has some drawbacks resulting in poor performance in some cases. Firstly, PSO is sensitive to the variance of parameters. The parameters adapted to one problem may not be suitable for another problem. Thus, appropriate parameter selection is essential for the PSO. Secondly, PSO is likely to obtain a local optimal solution so that the algorithm is stagnated. The particles tend to move to the position of the best optimal particle and hence will gather at a small range of its neighborhood, which is harmful to search for global best solution. The whole particle swarm will not change anymore when the best particle is trapped in to a local optimum. Thirdly, the result solution of the PSO is not stable because of the introduction of stochastic process. Though stochastic process helps increase the diversity of the swarm and decrease the probability to be trapped into local optima, the randomization of initial position, initial velocity and updating mechanism leads to different solution in each trial. Sometimes it even exists large deviation.

Many efforts have been made to improve the performance of PSO to address the main drawbacks of the particle swarm optimization. The essence of these advances is to achieve a trade-off between exploration and exploitation, which are the two key issues of metaheuristic algorithms[20]. Exploration is the ability to diversely search the space that supports the algorithm to scan the various parts of search space while avoid trapping into local optima[6]. In contrast, exploitation is the local search ability that supports a precise search and convergence.

This paper proposes a modified PSO called TPSO by introducing a stochastic process on T-distribution. A mechanism applying reference set is used to avoid premature. Section 2 illustrates the fundamental principle of PSO and t-distribution. Then the details of TPSO are discussed. Section 3 tests the performance of TPSO on 13 benchmark functions in comparison with seven classical metaheuristic algorithms and discusses the superiority of TPSO. Section 4 shows the application of TPSO on engineering problem. Section 5 concludes the main idea and result of TPSO and outlines directions for further research.

## 2. Algorithm

Here, the details of the original PSO algorithm, the proposed TPSO algorithm are provided. Also, the explanation of the t-distribution and the reference set are discussed.

### 2.1 Particle Swarm Optimization

The PSO algorithm consists of a collection of individuals called particles. Each particle represents a candidate solution to the program problem. Each particle includes three attributes: current position, function value, velocity, personal best position. The current position is a vector which represents the independent variable of the objective function to be optimized. For each particle a function value is calculated to evaluate the quality of the particle. The particles move in the high-dimensional space governed by an iterative calculation and addition of a velocity vector to the position vector. The calculation of each particle's velocity is based on its attraction towards two promising locations in the search space, namely the best position found by the particle and the best position found by any particle within the particle's neighborhood.

The neighborhood of a particle refers to the other particles within the swarm from which it may take influence. We consider the star neighborhood strategy where the neighbor-hood is the entire swarm in the paper.

For a particle in the particle swarm with position vector $x_i(t)$ and velocity $v_i(t)$ at the iteration $t$, and its velocity at the next iteration is updated

$$v_i(t+1) = \omega v_i(t) + c_1 \cdot rand() \cdot (p_i - x_i(t)) + c_2 \cdot rand() \cdot (pg - x_i(t)) \tag{2}$$

Where $rand()$ is a random number generator in uniform distribution. $p_g$ represents the global best position of the entire swarm and $p_i$ represents the historic best position of the particle $i$. $c_1$ and $c_2$ are positive acceleration coefficients that weight the importance of their cognitive and social components. Once given the new velocity, the position is updated by

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3}$$

Fig. 1 illustrates the mechanism of the movement of the particles.

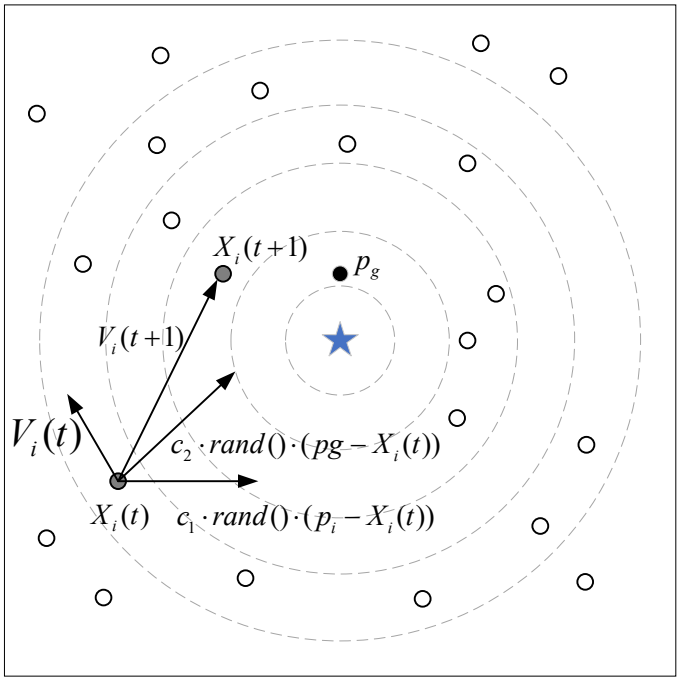



Figure 1. The movement mechanism of the particles in PSO tested by the Sphere benchmark function.

### 2.2 The Student's t-distribution

In probability and statistics, Student's t-distribution is a member of a family of continuous probability distributions that arises when estimating the mean of a normally distributed population in situations where the sample size is small and population standard deviation is unknown. It was developed by William Sealy Gosset under the pseudonym Student.

The t-distribution has been widely used in statistical analyses. Student's t-test can be applied for assessing the statistical significance of the difference between two sample means, the construction of confidence intervals for the difference between two population means, and in linear regression analysis. The Student's t-distribution also plays an important role in the Bayesian analysis.

Suppose $X$ is a normally distributed stochastic variable, whose expected value is $\mu$ and square deviation $\sigma^2$ is unknown. Define the sample mean as:

$$\overline{X}_n = \frac{X_1 + X_2 + \ldots + X_n}{n} \tag{4}$$

And define the variance as

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}_n\right)^2 \tag{5}$$

Then the statistic

$$T = \frac{\overline{X}_n - \mu}{\frac{S_n}{\sqrt{n}}} \tag{6}$$

is in accordance with t-distribution and its probability density function(pdf) is

$$f(t) = \frac{\Gamma(\frac{\nu+2}{2})}{\sqrt{\nu\pi}\Gamma(\nu/2)}(1+\frac{t^2}{\nu})^{\frac{-(\nu+1)}{2}} \tag{7}$$

Where $\nu = n-1$ and $\nu$ is usually called the degree of freedom of the t-distribution. The curves of the t-distribution's probability density function with different degree of freedom are shown in Fig. 2.

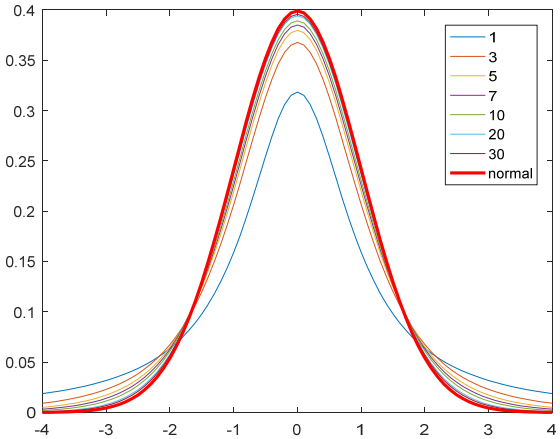



Figure 2. The probability density function of t-distribution and normal distribution.

The pdf of t-distribution is symmetric and bell-shaped, like the normal distribution, but has heavier tails, meaning that it is more prone to producing values that fall far from its mean. It should be noted

that as v → ∞, the pdf of the t-distribution converges to the pdf of the standard normal distribution for every value of x. From the point of view of the random number generator, the t-distribution being introduced into PSO will enhance the ability of exploring the entire search space and hence increases diversity.

### 2.3 Particle Swarm Optimization with t Distributions.

The standard PSO adopts the global best of the particle warm $p_g$ and the individual best $p_i$ to control the movement of the particles. The individual best is introduced to increase the diversity and consequently avoid being trapped into the local optimum to some extent. However, the existence of individual best may slow the convergence speed of PSO and even cause oscillation. Some researchers argue that the diversity can be achieved by using some randomness. Subsequently, there is no compelling reason for using the individual best, unless the optimization problem of interest is highly nonlinear and multimodal. A simplified version that could accelerate the convergence of the algorithm is to use the global best only[21]. However, eliminating the individual best contributes to premature convergence. Therefore, we can substitute the individual term with a stochastic process to improve the convergence speed without damage its diversity.

According to the above theory, we proposed a PSO with t-distribution (TPSO) which modified the standard PSO by replacing the individual best term with a t-distribution to maintain the diversity. The updated equation of a particle's velocity in TPSO is

$$v_i(t+1) = v_i^t + c_1 \cdot trnd() + c_2 \cdot (pg - x_i(t)) \tag{8}$$

where $trnd()$ is an random number can be drawn from t-distribution. The updated equation of a particle's position remains the same as Eq. (3).

It is obvious that the TPSO is easier to implement than the standard PSO and its classical varieties. Also, the mechanism is clear to understand. The TPSO retains two parameters to control the balance between exploitation and exploration. A bigger value of $c_1$ provides the ability to escape local optima while a bigger value of $c_2$ accelerates the convergence speed. In practice, is desired to be set as a monotonically decreasing function with respect to the iteration index $t$ while $c_2 = 0.2$ to 0.7 is proper for most cases.

### 2.4 A Modification with a Reference Set

The updating equation is sufficient for most optimization. However, when faced with a complex objective function with extremely high dimension or numerous local optimums, the other particles are prone to be attracted by the initialized one best particle. Although tuning parameters can improve the efficiency, it makes no sense to avoid premature.

To overcome the dilemma, we introduced a mechanism with a reference set to avoid rapid convergence in the early period of TPSO. The reference set consists of a certain number of position vectors which have the top optimal values. Compared with the original TPSO, particles are attracted to the weighted average of the position vectors in the reference set. So, the word 'reference' denotes that the position vectors in the set are role models for ordinary particles to follow. Consequently, the modified updating equation of a particle can be rewritten as

$$x_i(t+1) = x_i(t) + c_1 \bullet trnd() + c_2(p_t - x_i(t)) \tag{9}$$

where $p_t = \omega_1 p_g + (1-\omega_1) \cdot avg(S_t), 0 \le \omega \le 1$ . $p_g$ is the global best position and $avg(S_t)$ is the average position of position vectors in the reference set. $\omega_1$ is applied for adjusting the convergence speed and controlling the diversity of the swarms. The modification would degrade into the original

TPSO when $\omega_1$ decreases to 0. The reference set will be updated as the algorithm iterates and it will finally be replaced by those consists of global best particles only because the whole swarm will converge to a particle. Hence, the balance between exploitation and exploration can be self-adaptive.

The reference set can help increase diversity and reduce the possibility of premature in that the particles can learn from a swarm of particles rather than only one particle. The proposed mechanism will broaden ordinary particles' sight.

The following steps explain the mechanism of integrated TPSO.

Assume a swarm population with $N$ particles searching for an optimal solution in a *d*-dimensional search space. Each particle $i(i = 1, 2, ..., N)$ one *d*-dimensional position vector $x_i$. The objective of the TPSO algorithm is to minimize the given objective function $f(x)$ .Initialization: Iteration, $t = 0$. The size of the reference set is $S$.

**Step 1:** The positions of the particles are randomly initialized by

$$x_i(0) = (upper - lower) \cdot rand() + lower \tag{10}$$

where upper and lower are the bound of maximum and minimum respectively.

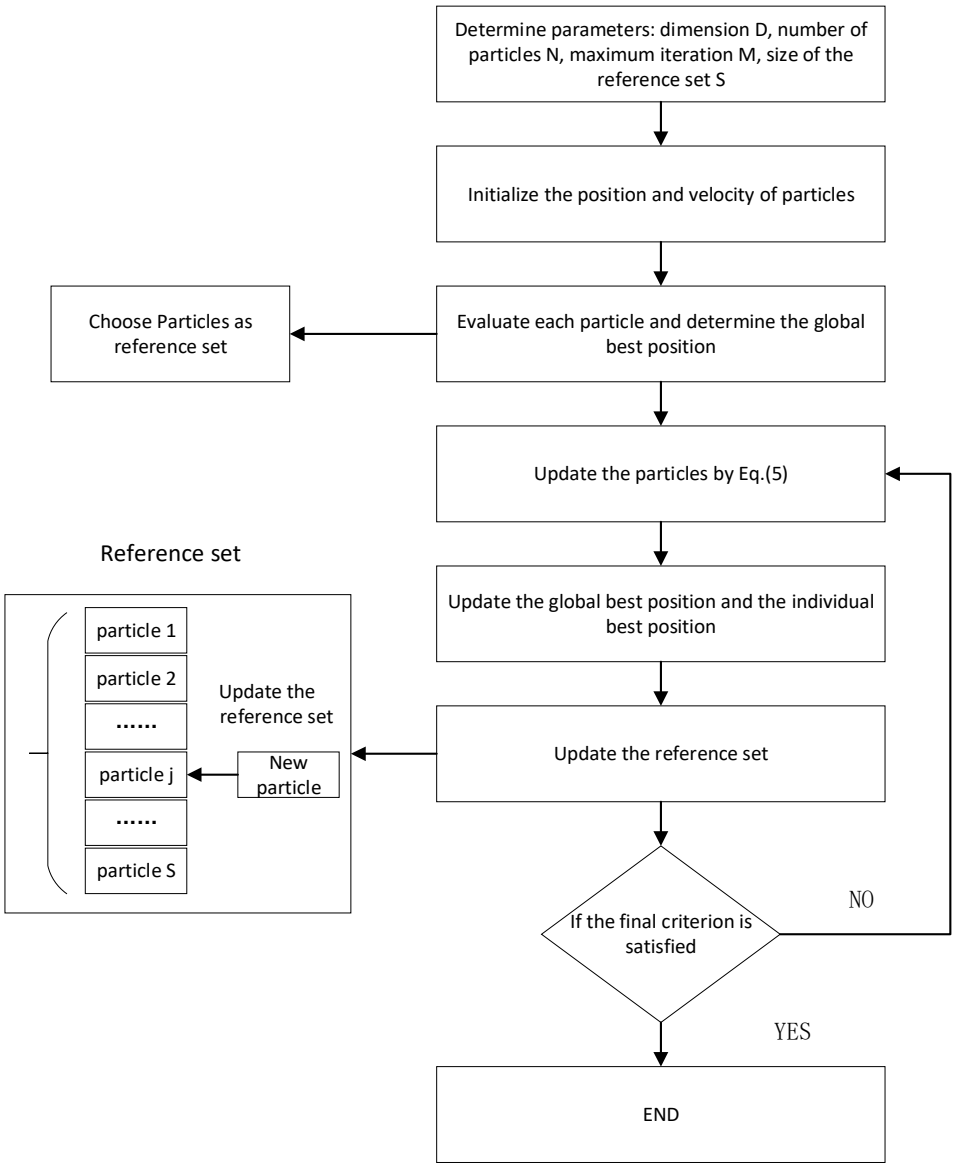



Figure 3. The flow chart of TPSO

**Step 2:** For each particle $i$, evaluate the objective function $f(x)$ using the position vector $x_i(0)$. Select the position of the particle with the minimal function value as the global best position vector $p_g$. Select the positions with the S minimal function value into the reference set.

**Step 3:** Update the particles' positions by Eq. (9).

**Step 4:** For each particle $i$, the $f(x_i)$ is evaluated using the position vector $x_i(t)$. Suppose that the position $j$ in the reference set and corresponding function value are $y_j$ and $S(y_j)$. And the reference set is sorted in ascending order according its objective function value. If $f(x_i)$ satisfies $S(y_j) \le f(x_i) \le S(y_{j+1})$, substitute $y_{j+1}$ with $x_i$ to update the reference set.

**Step 5:** For each particle, if the function value of a new position is lower than the global best value, then update $p_g$.

**Step 6:** If the iteration index $t$ reaches the maximum iteration $M$, end the loop. Determine as $p_g$ the solution of the problem.

In conclusion, a flowchart of the TPSO is show in Fig. 3.

## 3. Experimental Results and Analysis

We have conducted intensive experiments and statistical tests to evaluate the performance of the proposed TPSO and compared it with several other algorithms. The experimental results disclose several interesting outcomes in addition to establishing the effectiveness of the proposed method. All the tests conducted is performed on a PC with a 2.7 GHz CPU and 8 GB RAM. All programs are coded in MATLAB scripts.

### 3.1 Benchmark Functions and Parameter Selection

We selected 13 benchmark functions to test the performance of TPSO and the expression of the benchmark functions can be referred to Tab. 1. A more detailed description of these functions can be found in the literature [22]. Moreover, we compared TPSO on global optimization problem with 7 classical algorithms including ACO, BBO, DE, ES, GA, PSO and APSO. The former 6 algorithms are implemented by Gai-Ge Wang et al[23]. The APSO is proposed in Ref. [21].

The parameters of the former 6 algorithms are set to be the same as the parameters in Ref. [23]. The population size, problem dimension, maximum number of iterations and size of the reference set are 100, 20, 200, 10 respectively. In order to avoid random interference, all algorithms have been run for 100 times on each benchmark functions independently. It's important to highlight that the initial particles are generated under randomly uniform distribution. $c_1$ and $c_2$ are set as: $c_1 = 0.8, c_2 = 0.8^t$.

### 3.2 Results

The mean optimum value and the minimum optimum value of the 8 algorithms are shown in Tab. 2 and Tab. 3 respectively. In order to clearly distinguish the differences among the algorithms, all the results have been normalized according to the optimal results, and the data of the best performance algorithm is represented in black font.

From Tab. 2 and Tab. 3, it is obvious that TPSO has the strongest search ability of finding the optimal value of the 14 benchmark functions. Tab. 2 shows that TPSO performs the best in mean optimal value on 8 of the best benchmark functions (F1-F4, F7, F9, F11, F13. BBO ranks three and performs the best on F5, F6 and F8. APSO performs the best on F7, F10 and F12. It is worth noting that the optimum value obtained by TPSO in the tests that TPSO did not rank best is not far behind other algorithms. The worst rank that TPSO get is 5(F8). Therefore, it is crystal clear that TPSO has a large advantage in optimization of traditional benchmark functions. Although APSO performs best on more functions than TPSO in Table 3, it achieves extremely poorer performance on particularly functions (F8) while TPSO looks more balanced, which can be proved by rank analysis in the latter

section. Table 3 indicates that APSO has stronger local search ability. However, limited exploration resulted in less stability.

Table 1. Benchmark functions

| No. | Name | Definition |
|---|---|---|
| F1 | Ackely | $f(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e$ |
| F2 | Griewank | $f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$ |
| F3 | Penalty #1 | $f(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4), y_i = 1 + 0.25(x_i + 1)$<br>$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ |
| F4 | Penalty #2 | $f(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ |
| F5 | Quartic noise | $f(x) = \sum_{i=1}^{D}\left(i(x_i)^4 + randn(0,1)\right)$ |
| F6 | Rastigin | $f(x) = \sum_{i=1}^{D} x_i^2 - 10\cos 2\pi x_i + 10$ |
| F7 | Rosenbrock | $f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$ |
| F8 | Schwefel 2.26 | $f(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\lvert x_i \rvert^{1/2})$ |
| F9 | Schwefel 1.2 | $f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$ |
| F10 | Schwefel 2.22 | $f(x) = \sum_{i=1}^{n}\lvert x_i \rvert + \prod_{i=1}^{n}\lvert x_i \rvert$ |
| F11 | Schwefel 2.21 | $f(x) = \max\{\lvert x_i \rvert, 1 \le i \le n\}$ |
| F12 | Sphere | $f(x) = \sum_{i=1}^{n} x_i^2$ |
| F13 | Step | $f(x) = 6n + \sum_{i=1}^{n}\lfloor x_i \rfloor$ |

### 3.3 Statistical Analysis and Convergence Behavior Analysis

Rank analysis give us a normalized criterion to evaluate an algorithm. We ranked the mean optimization results of the eight algorithms on 13 benchmark functions to sort its performance according to its average rank. The individual ranks and average ranks of all the selected algorithms are given in Table 4. We can know that the TPSO achieved the top average rank and APSO achieved

the second top average rank. The results apparently indicate that TPSO has strong robustness and performs well over the diverse group of benchmark functions.

Table 2. The mean optimum value of the algorithms

| FUN | ACO | BBO | DE | ES | GA | PSO | APSO | TPSO |
|---|---|---|---|---|---|---|---|---|
| F1 | 1591.11 | 353.87 | 79.23 | 2741.61 | 1911.08 | 2150.23 | 17.89 | **1.00** |
| F2 | 42.83 | 24.01 | 20.53 | 4236.46 | 140.26 | 1168.20 | 32.94 | **1.00** |
| F3 | 1.15E+10 | 47.36 | 83.04 | 3.07E+10 | 861.26 | 6.34E+08 | 946.97 | **1.00** |
| F4 | 2.80E+12 | 1.00E+05 | 5.27E+04 | 2.17E+13 | 1.68E+06 | 1.04E+12 | 1.26E+05 | **1.00** |
| F5 | 1.92 | **1.00** | 1.16 | 7.33 | 1.95 | 2.24 | 1.84 | 1.81 |
| F6 | 37.58 | **1.00** | 30.64 | 86.04 | 5.84 | 51.01 | 12.17 | 5.70 |
| F7 | 85.56 | 2.33 | 1.18 | 166.09 | 1.64 | 22.30 | **1.00** | **1.00** |
| F8 | 4.23 | **1.00** | 44.49 | 86.00 | 14.48 | 105.76 | 108.66 | 44.78 |
| F9 | 2.88E+04 | 7879.16 | 40984.32 | 8.52E+04 | 2.96E+04 | 4.59E+04 | 3.56 | **1.00** |
| F10 | 4311.01 | 98.96 | 30.59 | 1.84E+06 | 1977.96 | 3435.50 | **1.00** | 361.75 |
| F11 | 6143.32 | 3876.18 | 2485.11 | 1.21E+04 | 7197.81 | 7877.95 | 156.24 | **1.00** |
| F12 | 2.93E+06 | 1.19E+04 | 433.92 | 1.71E+07 | 1.57E+05 | 3.50E+06 | **1.00** | 12.38 |
| F13 | 2.48 | 1.13 | 1.00 | 221.52 | 6.00 | 51.64 | 1.01 | **1.00** |

Table 3. The minimum optimum value of the algorithms

| FUN | ACO | BBO | DE | ES | GA | PSO | APSO | TPSO |
|---|---|---|---|---|---|---|---|---|
| F1 | 6.95E+00 | 1.57E+00 | 4.10E-01 | 1.81E+01 | 1.02E+01 | 1.32E+01 | **1.57E-03** | 5.05E-03 |
| F2 | 1.58E+00 | 1.10E+00 | 1.00E+00 | 1.56E+02 | 2.52E+00 | 3.94E+01 | 5.44E-01 | **4.00E-06** |
| F3 | 7.83E-03 | 4.40E-02 | 1.64E-01 | 4.30E+07 | 1.10E+00 | 3.72E+05 | 4.73E-01 | **7.39E-07** |
| F4 | 7.45E+00 | 4.03E-01 | 1.92E-01 | 1.23E+08 | 4.33E+00 | 3.52E+06 | **5.30E-07** | 4.53E-06 |
| F5 | 7.69E+00 | **4.67E+00** | 5.55E+00 | 2.17E+01 | 8.73E+00 | 9.57E+00 | 7.98E+00 | 7.07E+00 |
| F6 | 9.37E+01 | **0.00E+00** | 7.52E+01 | 2.17E+02 | 2.83E+00 | 1.26E+02 | 1.29E+01 | 7.97E+00 |
| F7 | 6.23E+02 | 1.19E+01 | 1.82E+01 | 1.74E+03 | 1.87E+01 | 2.11E+02 | 1.46E+01 | **1.14E+01** |
| F8 | 8.19E+01 | **1.54E+01** | 1.47E+03 | 2.87E+03 | 1.53E+02 | 3.12E+03 | 3.24E+03 | 1.27E+03 |
| F9 | 2.77E+03 | 3.90E+02 | 4.16E+03 | 8.80E+03 | 2.58E+03 | 4.54E+03 | **1.23E-05** | 3.31E-03 |
| F10 | 2.94E+01 | 4.00E-01 | 1.73E-01 | 8.61E+01 | 8.31E+00 | 1.98E+01 | **6.46E-03** | 1.96E-02 |
| F11 | 1.93E+01 | 6.20E+00 | 9.56E+00 | 5.35E+01 | 1.60E+01 | 3.25E+01 | **7.18E-04** | 3.29E-03 |
| F12 | 6.36E+00 | 2.02E-02 | 1.19E-03 | 5.78E+01 | **0.00E+00** | 1.22E+01 | 2.89E-06 | 3.26E-05 |
| F13 | 1.90E+02 | 1.24E+02 | **1.20E+02** | 2.28E+04 | 3.68E+02 | 4.79E+03 | **1.20E+02** | **1.20E+02** |

Table 4. The ranks for the mean optimum value of the 8 algorithms

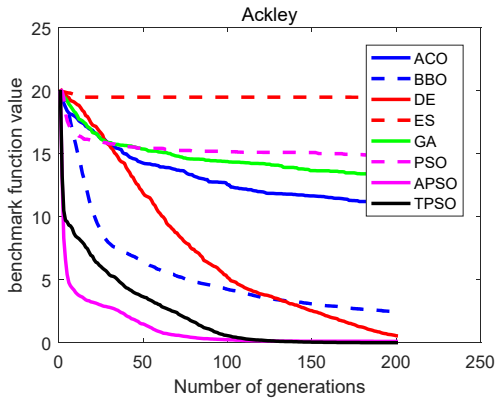
| FUN | ACO | BBO | DE | ES | GA | PSO | APSO | TPSO |
|---|---|---|---|---|---|---|---|---|
| F1 | 5 | 4 | 3 | 8 | 6 | 7 | 2 | **1** |
| F2 | 5 | 3 | 2 | 8 | 6 | 7 | 4 | **1** |
| F3 | 7 | 2 | 3 | 8 | 4 | 6 | 5 | **1** |
| F4 | 7 | 3 | 2 | 8 | 5 | 6 | 4 | **1** |
| F5 | 5 | **1** | 2 | 8 | 6 | 7 | 4 | 3 |
| F6 | 6 | **1** | 5 | 8 | 3 | 7 | 4 | 2 |
| F7 | 7 | 5 | 3 | 8 | 4 | 6 | 2 | **1** |
| F8 | 2 | **1** | 4 | 6 | 3 | 7 | 8 | 5 |
| F9 | 4 | 3 | 6 | 8 | 5 | 7 | 2 | **1** |
| F10 | 7 | 3 | 2 | 8 | 5 | 6 | **1** | 4 |
| F11 | 5 | 4 | 3 | 8 | 6 | 7 | 2 | **1** |
| F12 | 6 | 4 | 3 | 8 | 5 | 7 | **1** | 2 |
| F13 | 5 | 4 | 2 | 8 | 6 | 7 | 3 | **1** |
| AVE | 5.46 | 2.92 | 3.08 | 7.85 | 4.92 | 6.69 | 3.23 | 1.85 |

Table 5. The time cost of the algorithms (s)

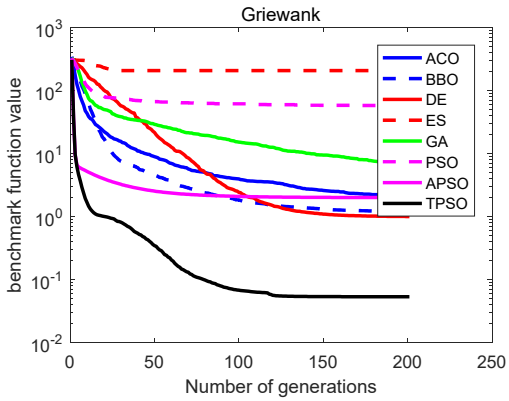
| FUN | ACO | BBO | DE | ES | GA | PSO | APSO | TPSO |
|---|---|---|---|---|---|---|---|---|
| F1 | 5.686 | 3.194 | 4.162 | 2.988 | 3.68 | 5.954 | 0.183 | 0.404 |
| F2 | 21.644 | 4.566 | 4.395 | 3.113 | 3.795 | 6.081 | 0.193 | 0.404 |
| F3 | 6.912 | 3.776 | 5.54 | 3.67 | 4.263 | 6.54 | 0.206 | 0.403 |
| F4 | 6.827 | 3.714 | 5.392 | 3.549 | 4.171 | 6.445 | 0.169 | 0.387 |
| F5 | 4.706 | 3.53 | 4.765 | 3.309 | 3.924 | 6.27 | 0.247 | 0.454 |
| F6 | 4.202 | 3.359 | 4.882 | 3.35 | 4.016 | 6.28 | 0.152 | 0.399 |
| F7 | 4.326 | 3.613 | 5.043 | 3.445 | 4.073 | 6.416 | 0.104 | 0.312 |
| F8 | 19.872 | 4.7 | 4.764 | 3.348 | 4.031 | 6.317 | 0.216 | 0.403 |
| F9 | 9.776 | 5.875 | 8.313 | 5.296 | 5.802 | 7.979 | 0.142 | 0.355 |
| F10 | 4.224 | 3.108 | 4.112 | 2.923 | 3.611 | 5.849 | 0.166 | 0.39 |
| F11 | 8.498 | 3.651 | 4.71 | 3.315 | 3.949 | 6.231 | 0.137 | 0.35 |
| F12 | 4.132 | 3.297 | 4.559 | 3.179 | 3.85 | 6.083 | 0.136 | 0.346 |
| F13 | 8.244 | 3.468 | 4.418 | 3.122 | 3.719 | 5.957 | 0.099 | 0.308 |
| **SUM** | **109.049** | **49.851** | **65.055** | **44.607** | **52.884** | **82.402** | **2.15** | **4.915** |

Note that TPSO outperforms a few algorithms, TPSO costs less time than other algorithms except APSO. The time cost of each algorithm on 13 benchmark functions is recorded in Table 5. It is apparent that APSO and TPSO are at the same level compared with other algorithms.

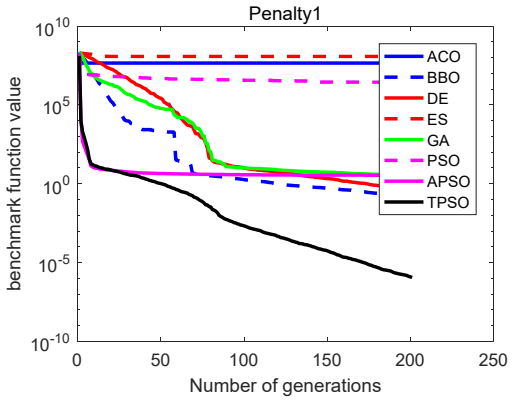
Convergence characteristic is the representative criterion of evaluating an algorithm and the convergence characteristic of the 8 algorithms is shown on Fig. 4(a)-(l). The convergence curve of F5 is not present due to its noise interference and hence resulting oscillations. Each figure expresses a convergence curve of a benchmark function and different algorithms are represented by variable line-style. The horizontal coordinate represents the number of iterations, and the ordinate represents the optimal function value on the current iteration number. Each point in the convergence curve is averaged by times of independent tests. From the figures, we can guarantee the local convergence of the TPSO as it always converges to a local extremum point. While the global convergence of TPSO can't be proved strictly, it has more accuracy than existing algorithms. Moreover, TPSO has the fast convergence speed on three benchmark functions (F2-F4). And TPSP has convergence rate of the same magnitude as APSO. Due to its wide search space, TPSO converges more slowly than APSO in the beginning until APSO is trapped into local optimum.



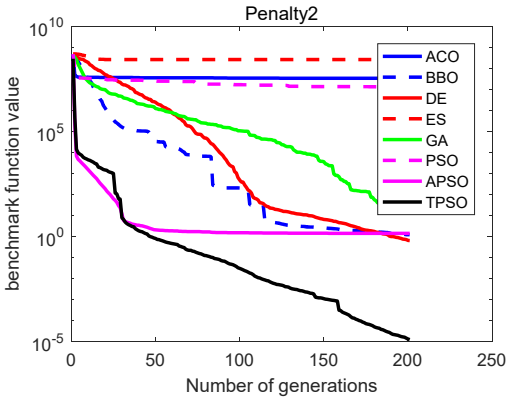

(a) F1



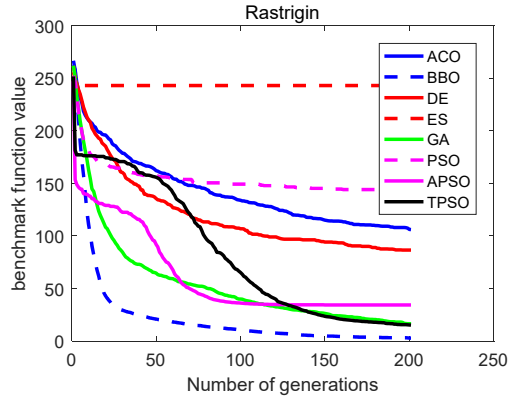

(b) F2



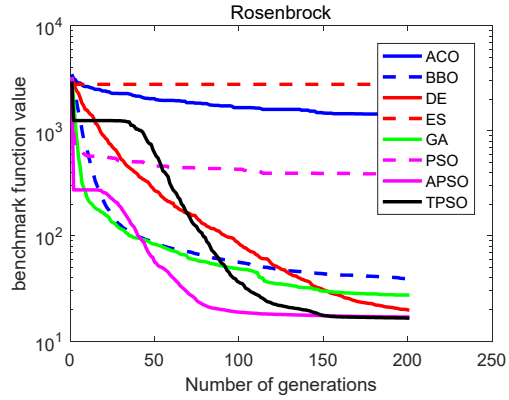

(c) F3



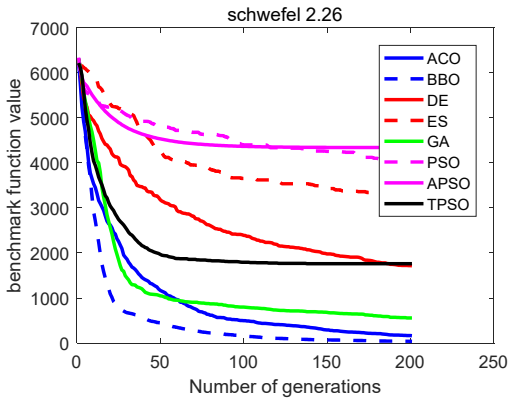

(d) F4



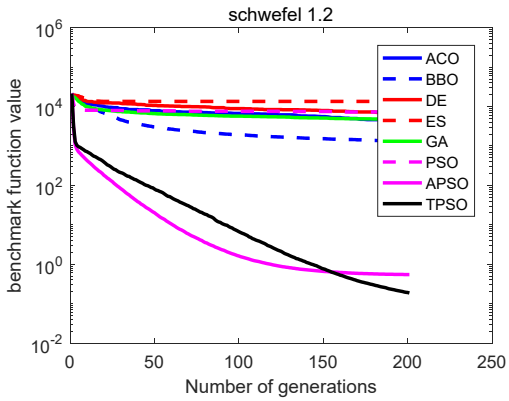

(e) F5



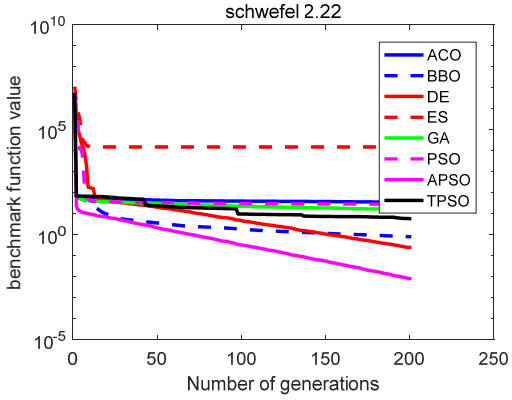

(f) F6



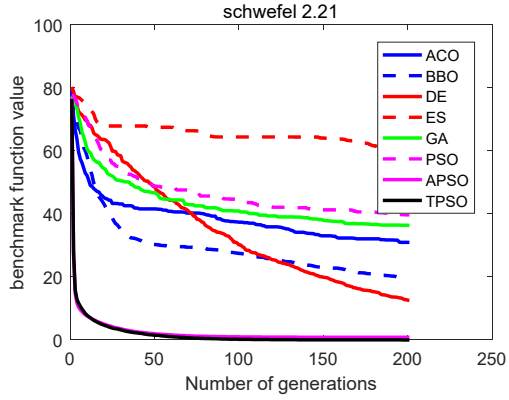

(g) F7



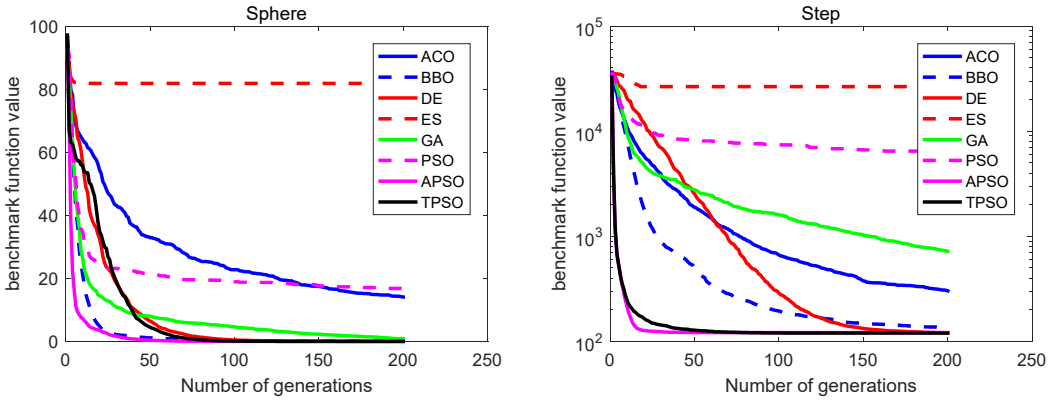

(h) F8




(i) F9




(j) F10




(k) F11 (l) F12

Figure 4. The convergence line graphs of 12 benchmark functions (the Quartic noise function (F5) is ignored due to instability brought by noise)

## 4. Engineering Benchmark

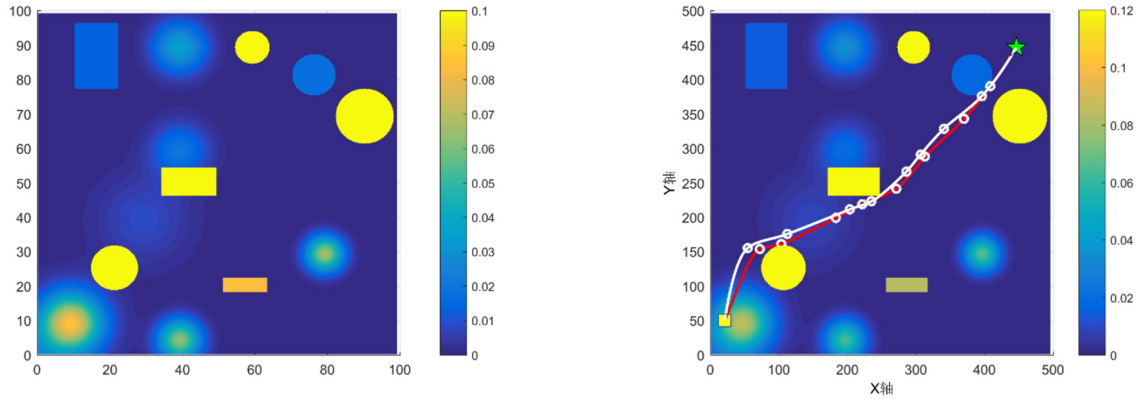
In this section, the path planning problem of UAV is solved as a benchmark to demonstrate the capability of the TPSO. The path planning problem is a complex Np-hard engineering problem.

Assume that UAV flies from start point S to target point T. The feasible path consists of N waypoints and two endpoints. It can be expressed as $Path=\{S,P_1,P_2,...P_N,T\}$, where $x=P_1,P_2,...,P_N$ is the coordinate sequence of the waypoints. Find the optimal sequence is the objective of the path planning problem.

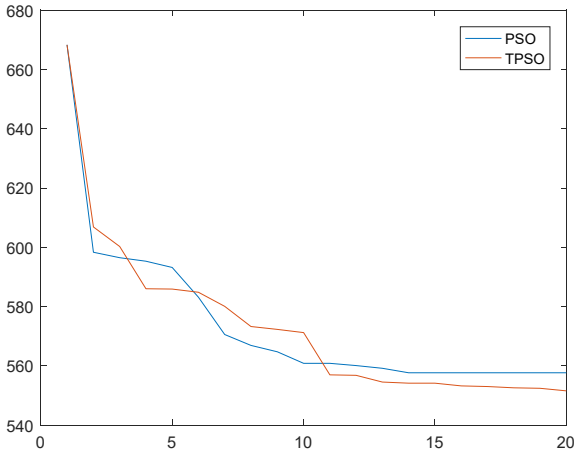
The cost function is applied to evaluate the feasible paths' quality. It is determined by two indexes: overall length $L$ and threat degree $P$ .Overall length represents fuel consumption and threat degree represents probability of failure in flight process. The cost function is calculated by:

$$J=\omega_1 L+\omega_2 P \tag{11}$$

$\omega_1$ and $\omega_2$ are weight parameters determined by the user. In this paper, they are set as $\omega_1=1$, $\omega_2=100$.




(a) Simulation Map (b) Comparison of two algorithms




(c)

Figure 5. The visual simulation result and convergence characteristic curve of PSO and TPSO

The map construction method in Ref. [24] is introduced. The simulation map is shown on Fig. 5(a). The map is set as a 100×100 area. The intensity of the color represents the magnitude of threat degree. The more yellow area represents the bigger threat degree and the UAV is less likely to fly through.

The result paths calculated by PSO and TPSO are shown on Fig. 5(b). The white curve represents the path optimized by TPSO while the red curve represents the path optimized by PSO. Fig. 5(c) shows the convergence characteristic during the optimization process. The path length and threat degree of the two algorithms are given in the Tab. 6. It is clear that TPSO can obtain less path length, threat degree and consequently less cost function comparing with PSO. The application of TPSO in path planning problem efficiently illustrates the strong capability of TPSO in solving engineering problems.

Table 6. The optimization criterion comparison between PSO and TPSO

| Algorithm | Path length | Threat degree | Cost function value |
|---|---|---|---|
| PSO | 540.3 | 0.173 | 557.6508 |
| TPSO | **536.7** | **0.147** | **551.5266** |

## 5. Summary

The paper proposed a modified PSO based on t-distribution stochastic process (TPSO) which substitutes normal stochastic process in basic PSO with t-distribution stochastic process. Subsequently the concept of reference set was introduced. There are two main ways to improve the performance of meta-heuristic algorithms: intensification and diversification. TPSO adapts the strategy of improving diversity of Metaheuristics by introducing t-distribution stochastic process and reference set and avoid decreasing intensify by eliminating self- recognition of PSO.

To validate the performance of TPSO, we used TPSO to solve 13 unconstrained benchmark functions and compared it with 7 classical metaheuristic algorithms. The results show that TPSO has more powerful efficiency than other algorithms. The paper also analyzed convergence characteristics of TPSO. Finally, TPSO was applied on the path planning problem of UAV to demonstrate the solving ability of TPSO.

## Acknowledgments

This research was funded by Aeronautical Science Foundation of China, grant number 20165557005.

## References

[1]. Wang G-G, Hossein Gandomi A, Hossein Alavi A, A chaotic particle-swarm krill herd algorithm for global numerical optimization. Kybernetes, Vol. 42(2013), p. 962-978.

[2]. Gandomi A H, Yang X-S, Alavi A H, Mixed variable structural optimization using firefly algorithm. Computers & Structures, Vol. 89(2011), p. 2325-2336.

[3]. Nocedal J, Wright S J, Numerical Optimization Second Edition. Springer, 1999.

[4]. Wang G-G, Gandomi A H, Alavi A H, A comprehensive review of krill herd algorithm: variants, hybrids and applications. Artificial Intelligence Review, Vol. 51 (2017) , p. 1-30.

[5]. Yang X S, Firefly Algorithm, Stochastic Test Functions and Design Optimization. International Journal of Bio-Inspired Computation, Vol. 2(2010), p. 78-84.

[6]. Rashedi E, Rashedi E, Nezamabadi-Pour H, A comprehensive survey on gravitational search algorithm. Swarm and Evolutionary Computation, Vol. 41(2018), p. 141-158.

[7]. Moharam R, Morsy E, Genetic algorithms to balanced tree structures in graphs. Swarm and Evolutionary Computation, Vol. 32(2017), p. 132-139.

[8]. Vanneschi L, Henriques R, Castelli M, Multi-objective genetic algorithm with variable neighbourhood search for the electoral redistricting problem. Swarm and Evolutionary Computation, Vol. 36(2017), p. 37-51.

[9]. Sabar N R, Abawajy J, Yearwood J, Heterogeneous Cooperative Co-Evolution Memetic Differential Evolution Algorithm for Big Data Optimization Problems. IEEE Transactions on Evolutionary Computation, Vol. 21(2017), p. 315-327.

[10]. Raitoharju J, Samiee K, Kiranyaz S, Particle swarm clustering fitness evaluation with computational centroids. Swarm and evolutionary computation, Vol. 34(2017), p. 103-118.

[11]. Azadeh A, Farahani M H, Eivazy H, A hybrid meta-heuristic algorithm for optimization of crew scheduling. Applied Soft Computing, Vol. 13(2013), p. 158-164.

[12]. Das P, Behera H S, Panigrahi B K, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm and Evolutionary Computation, Vol. 28(2016), p. 14-28.

[13]. Chih M, Yeh L-L, Li F-C, Particle swarm optimization for the economic and economic statistical designs of the X control chart. Applied Soft Computing, Vol. 11(2011), p. 5053-5067.

[14]. Yeh W-C, Lin Y-C, Chung Y Y, A particle swarm optimization approach based on Monte Carlo simulation for solving the complex network reliability problem. IEEE Transactions on Reliability, Vol. 59(2010), p. 212-221.

[15]. Dabiri N, Tarokh M J, Alinaghian M, New mathematical model for the bi-objective inventory routing problem with a step cost function: A multi-objective particle swarm optimization solution approach. Applied Mathematical Modelling, Vol. 49(2017), p. 302-318.

[16]. Chouikhi N, Ammar B, Rokbani N, PSO-based analysis of Echo State Network parameters for time series forecasting[J]. Applied Soft Computing, Vol. 55(2017), p. 211-225.

[17]. Marinakis Y, Migdalas A, Sifaleras A, A hybrid particle swarm optimization–variable neighborhood search algorithm for constrained shortest path problems. European Journal of Operational Research, Vol. 261(2017), p. 819-834.

[18]. Chen W, Zhai P, Zhu H, Hybrid algorithm for the two-dimensional rectangular layer-packing problem. Journal of the Operational Research Society, Vol. 65(2014), p. 1068-1077.

[19]. Dai J, Han H, Hu Q, Discrete particle swarm optimization approach for cost sensitive attribute reduction. Knowledge-Based Systems, Vol. 102(2016), p.116-126.

[20]. Lozano M, García-Martínez C, Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. Computers & Operations Research, Vol. 37(2010), p. 481-497.

[21]. Gandomi A H, Yun G J, Yang X S, Chaos-enhanced accelerated particle swarm optimization. Communications in Nonlinear Science and Numerical Simulation, Vol. 18(2), 327-340(2013).

[22]. Simon D, Biogeography-Based Optimization. IEEE Transactions on Evolutionary Computation. Vol. 12 (2008), p. 702-713.

[23]. Wang G G, Gandomi A H, Zhao X J, Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Computing, Vol. 20(2016), p. 273-285.

[24]. Zhang Yu, Chen Jin, Shen Lincheng, Military Aircraft Route Planning Based on Probabilistic Map. Computer Simulation, Vol. 24(2007), p. 62-66.