Physica A 537 (2020) 122802

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

STSA: A sine Tree-Seed Algorithm for complex continuous optimization problems☆

Jianhua Jiang [a,b,*], Meirong Xu [a], Xianqiu Meng [a], Keqin Li [c,*]

[a] Department of Data Science, Jilin University of Finance and Economics, Changchun 130117, PR China
[b] Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Jilin University, Changchun, 130012, PR China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 28 June 2019
Received in revised form 16 September 2019
Available online 19 September 2019

MSC:
68W20
68T20

Keywords:
Tree-Seed Algorithm
Sine Cosine Algorithm
Continuous optimization
Optimization algorithm

ABSTRACT

Tree-Seed Algorithm (TSA) has good performance in solving various optimization problems. However, it is inevitable to suffer from slow exploitation when solving complex problems. This paper makes an intensive analysis of TSA. In order to keep the balance between exploration and exploitation, we propose an adaptive automatic adjustment mechanism. The number of seeds can be defined in the initialization process of the optimization algorithm. In order to further improve the convergence rate of TSA, we also modify the change model of seed numbers in the initialization process with randomly changing from more to less. With the improvement of two mechanisms, the main weakness of TSA has been overcome effectively. Based on the above two improvements, we propose a new algorithm-Sine Tree-Seed Algorithm (STSA). STSA achieves good results in solving high-dimensional complex optimization problems. The results obtained from 24 benchmark functions confirm the excellent performance of the proposed method.

1. Introduction

Optimization problems refer to the determination of the value of some optional variables under certain conditions so as to optimize the selected objective functions [1]. It is the inherent characteristic of achieving the best (minimum or maximum) in a given situation [2]. Optimization has been widely applied to engineering [3–5], industrial design [6,7], design analysis [8], etc.

Meanwhile, the heuristic algorithms have also been used extensively [9]. Heuristic algorithm is a kind of optimization algorithm. Most of heuristic algorithms are inspired by evolutionary phenomena, collective behavior of creatures, physical rules and human-related concepts [10,11]. Some of the recent and popular algorithms in each of these subclasses are as follows:

- Evolutionary techniques: Genetic Algorithms (GA) [12–15], Differential Evolution (DE) [16–18], Biogeography-Based Optimization algorithm (BBO) [19], Evolution Strategy (ES) [20], etc.

☆ The authors are grateful to the financial support by the National Natural Science Foundation of China (no. 61572225), Natural Science Foundation of the Science and Technology Department of Jilin Province, China (no. 20180101044JC), the Social Science Foundation of Jilin Province, China (nos. 2019B68, 2017BS28), the Foundation of the Education Department of Jilin Province, China (nos. JJKH20180465 kJ) and the Foundation of Jilin University of Finance and Economics (no. 2018Z05).

* Corresponding authors.
E-mail addresses: jianhuajiang@yahoo.com (J. Jiang), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.physa.2019.122802
0378-4371/

- Swarm intelligence techniques: Ant Colony Optimization (ACO) [21], Particle Swarm Optimization (PSO) [22–24], Artificial Bee Colony algorithm (ABC) [25,26], Firefly Algorithm (FA) [27], etc.
- Physics-based techniques: Gravitational Search Algorithm (GSA) [28], Colliding Bodies Optimization (CBO) [29], Black Hole (BH) [30], etc.
- Human-related techniques: League Championship Algorithm (LCA) [31], Mine Blast Algorithm (MBA) [32], Teaching–Learning-Based Optimization (TLBO) [33], etc.

Tree-Seed Algorithm (TSA) was proposed by Kiran in 2015 [34] which was inspired by the evolutionary method of the relationship between trees and seeds [34]. It performs quite well in solving continuous optimization problems [20,34–36], and it is widely studied and applied by scholars.

Although studies show that TSA has achieved better performance in optimization problems [20,35], it still has its inherent shortcomings:

1. The selection of random numbers is simple but unreasonable. The change of random values will have a great impact on the optimization results.
2. The global search ability of TSA is deficient. In the process of running the program, it is uncontrollable for the selection of the next location, and the degree of randomness is strong, which is not conducive to the accuracy of the experimental results.

The global searching ability of TSA is poor. In TSA, exploration requires to search for optimal values extensively in an algorithm, while exploitation is limited to search in local available space. Thus, they have a conflict in that the realization of one means the sacrifice of the other. As a result, it is a key and challenging problem for all optimization algorithms to achieve an appropriate balance between exploration and exploitation. Sine Cosine Algorithm (SCA) has a good ability of global search. Inspired by SCA [37], this paper proposes two improvements for TSA:

1. Adaptive seeds generation mechanism.
2. A regulatory mechanism k linearly varying with the times of iteration.

In this paper, we analyze the effect of seed number generation mechanism on TSA optimization process. In order to improve the searching ability of TSA, we propose an adaptive seed generation mechanism, in which the number of seeds varies from more to less with the number of iterations. The results show that the production mechanism of seeds has a certain positive effect on searching ability. At the same time, inspired by SCA, we also propose a balance parameter k which linearly changing with the number of iterations. k plays an important role in balancing exploration and exploitation. k improves the exploration of TSA in the first part of optimization process, and improves the exploitation in the second part.

The rest of this paper can be divided into the following sections: two basic algorithms: TSA and SCA are introduced in Section 2. The improvement of TSA is detailed in Section 3. The experimental results of Sine Tree-Seed Algorithm(STSA) are presented in Section 4. Applications of STSA in engineering design problem are shown in Section 5. The discussions of the algorithm are reported in Section 6. Finally, the conclusion is given in Section 7.

2. Related work

STSA improves TSA [34] based on the inspiration of SCA [37]. There are details of the related algorithm in the followings. TSA is introduced in the first subsection, and the SCA is depicted in the second subsection.

2.1. TSA: Tree-Seed Algorithm

TSA is proposed by Mustafa Servet Kiran [34] from the University of Selcuk in 2015. It is a population-based heuristic searching algorithm recently proposed to solve continuous optimization problems. In TSA, trees and seeds represent the possible solutions for the optimization problems [34]. The algorithm uses a new intelligent optimizer based on the relationship between trees and seeds for solving continuous optimization problems. TSA is extensively utilized in the field of heuristic and population-based search. This proposed method is considered to have improved the original defects of the optimization problems, that is, the inverse correlation between exploration and exploitation in the searching process. Two position updating equations are used in TSA, and ST controls the selection of the position updating equation to produce seeds for the tree. ns is used to determine the number of seeds produced by the trees. TSA uses some criteria to find the best solution amongst all possible ones for an optimization problem as follows:

$$f(\overrightarrow{x}) \leq f(\overrightarrow{y}), \forall \overrightarrow{y} \in F \tag{1}$$

$$f(\overrightarrow{x}) \geq f(\overrightarrow{y}), \forall \overrightarrow{y} \in F \tag{2}$$

where, provided S is a search space, F is a set of acceptable solutions of S, and f is an objective function to find minimization or maximization through Eqs. (1) and (2).

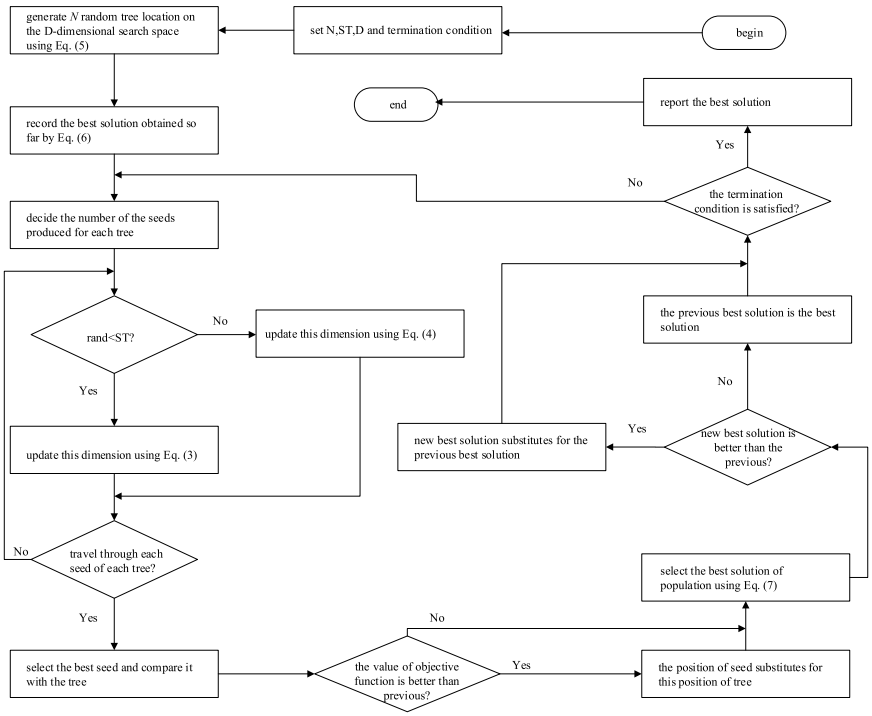


Fig. 1. The flow chart of TSA.

In TSA, the update rules are Eqs. (3) and (4). Eq. (3) takes into consideration the tree location where the seeds will be produced, as well as the best location for the tree population. This equation also improves the local searching capability. Eq. (4) uses two different tree locations for producing a new seed for the tree.

$$S_{i,j} = T_{i,j} + \alpha_{i,j} \times (B_j - T_{r,j}) \tag{3}$$

$$S_{i,j} = T_{i,j} + \alpha_{i,j} \times (T_{i,j} - T_{r,j}) \tag{4}$$

where, $S_{i,j}$ is j_{th} dimension of i_{th} seed that will be produced in i_{th} tree, $T_{i,j}$ is the j_{th} dimension of i_{th} tree, B_j is the j_{th} dimension of best tree location obtained so far, $T_{r,j}$ is the j_{th} dimension of the r_{th} tree randomly selected from the population, α is the scaling factor randomly produced in range of $[-1, 1]$, i and r are different indices.

In the beginning of the search progress with TSA, the initial tree locations that are possible solutions of the optimization problem are produced by Eq. (5):

$$T_{i,j} = L_{j,min} + r_{i,j} \times (H_{j,max} - L_{j,min}) \tag{5}$$

where, $L_{j,min}$ is the lower bound of the search space, $H_{j,max}$ is the higher bound of the search space, and $r_{i,j}$ is a random number produced for each dimension and location, in range of [0, 1].

In order to realize minimization, the best solution is selected from the population via Eq. (6).

$$B = min(f(\overrightarrow{T_i})) \tag{6}$$

where, i is the serial number of trees in the population. It takes integer values from 1.

For all experiments, the maximum number of function evaluations *MaxFEs* is selected as the termination condition and it is set by using the dimensionality of the function given in Eq. (7) as follows.

$$MaxFEs = D \times 10000 \tag{7}$$

The flow chart of TSA is shown in Fig. 1.

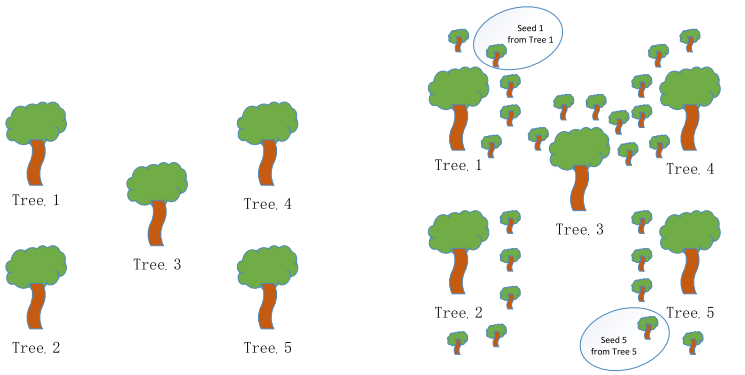
The working diagram of TSA can be summarized as Fig. 2:

In Fig. 2(a), trees are scattered to the search space and the fitness of the trees are calculated by using objective function specific for the optimization problem [34].

In Fig. 2(b), the number of seeds for each tree is changeable. In the diagram, five seeds are produced for each tree and the best seeds are compared with the parent tree [34].

In Fig. 2(c), if fitness of the best seed is better than the fitness of its parent tree, the parent tree is removed, and its best seed replaces to stand [34].

The maximum number of evaluations of a function is the termination condition of TSA.



(a) The initialization of TSA (b) The seed production mechanism

(c) The replacement procedure

Fig. 2. Working diagram of TSA.

2.2. SCA: Sine Cosine algorithm

SCA is proposed by Seyedali Mirjalili (2015) [37], which is a novel population-based optimization algorithm for solving optimization problems. The optimization process of SCA starts with a set of random solutions repeatedly evaluated by an objective function and improved by a set of rules that make up for the kernel of an optimization technique.

In SCA [37], the position updating equations are as follows:

$$X_i^{t+1} = X_i^t + r_1 \times sin(r_2) \times \mid r_3 \times P_i^t + X_i^t \mid, r_4 < 0.5 \tag{8}$$

$$X_i^{t+1} = X_i^t + r_1 \times cos(r_2) \times \mid r_3 \times P_i^t + X_i^t \mid, r_4 \geq 0.5 \tag{9}$$

where, X_i^t is the position of the current solution in i_{th} dimension at t_{th} iteration, r_1 is a random number, r_2/r_3 are random numbers, r_4 is a random number in [0, 1], P_i^t is the position of the destination point in i_{th} dimension.

In order to balance exploration and exploitation, the range of sine and cosine in Eqs. (8) and (9) is changed adaptively using Eq. (10):

$$r_i = a - t \times \frac{a}{T} \tag{10}$$

where, t is the current iteration, T is the maximum number of iterations, and a is a constant. In SCA, the value of a is generally 2 [37].

The flow chart of SCA is shown in Fig. 3.

3. Methods

The optimization of algorithms has always been a hot topic in academic research. Optimization refers to the process of finding out the optimal solution in all solutions of an optimization problem, which usually involves maximum and minimum optimization. In the process of optimization, the existence of random factors has a certain impact on the results of optimization. There are many methods to determine random factors, but TSA only uses a relatively simple method to

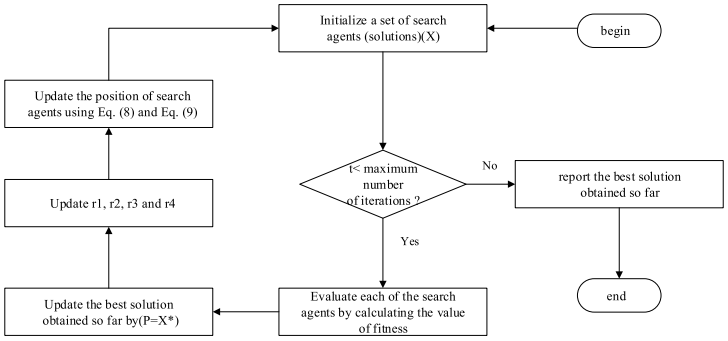


Fig. 3. The flow chart of SCA.

obtain random numbers. This needs to be improved, so we improve it to form a new algorithm, which is a more accurate optimization algorithm.

The first improvement: adaptive seeds generation mechanism is introduced in Section 3.1. Section 3.2 reports the second improvement: a regulate mechanism k linearly varying with the times of iteration. This paper combines two improvements and obtains a new algorithm. The new algorithm based on the above two improvements is detailed in Section 3.3.

3.1. Adaptive seeds generation mechanism

In TSA [34], seeds play an important role in spreading and searching for optimal values, but it is unreasonable since the way of producing seeds is extremely simple and random. The production of seeds will has a certain impact on the optimization results, which will deviate from the optimal solution. Therefore, we have made the following improvements: modify the *ns* value so that it can be processed according to the change of *FEs* value. It mainly updates the number of seeds randomly from more to less, so it can find the best solution gradually. We set a ratio of *ratioFEs* by Eq. (11) and let *ns* vary with the change of *ratioFEs* by Eqs. (12) and (13).

$$ratioFEs = \frac{FEs}{MaxFEs} \tag{11}$$

$$xTheat = 0.5 \times ratioFEs \times \pi \tag{12}$$

$$ns = L + |(H - L) \times cos(xTheat)| + 1 \tag{13}$$

where, *ns* is the number of seeds, *FEs* is the number of function evaluations, *MaxFEs* is $D\times$ 10000, which is the termination condition of the all experiments, *ratioFEs* is the radio of *FEs* and *MaxFEs*, and the range of *ratioFEs* is [0, 1]. The range of *xTheat* is $[0, 0.5\pi]$, L is the low bound of the number of seeds generated by a tree, H is the upper bound of the number of seeds generated by a tree. By changing the random number selection of *ns*, the global search ability is greatly improved.

3.2. A regulate mechanism k linearly varying with the time of iterations

k is a new parameter based on the original TSA, which combines the inspiration of SCA. It linearly changes with the number of iterations. In the first part of the optimization process, exploration can be improved. In the second part of the optimization process, exploitation can be improved. This mechanism has a certain effect on the current solution to jump out of the local environment and convert it into another. The calculation method of k is as Eq. (14).

$$k = 2 \times (1 - ratioFEs) \tag{14}$$

where, *ratioFEs* is the radio of *FEs* and *MaxFEs*, the range of *ratioFEs* is [0, 1], the range of k is [0, 2].

3.3. New algorithm: Sine Tree-Seed Algorithm (STSA)

Obtaining a location of a seed that will be produced from a tree is important for the optimization problem because this process constitutes is the core of search. After the above two improvements, we propose three search equations Eqs. (15)–(17) for this process.

$$S_{i,j} = tmpRand \times T_{komsu,j} + (1 - tmpRand) \times B_j, rand < 0.5ST \tag{15}$$

$$S_{i,j} = T_{i,j} + k(B_j - r_{i,j} \times T_{i,j}) \times (sin(\pi \times acos(r_{i,j}))), 0.5ST \leq rand < ST \tag{16}$$

$$S_{i,j} = r_{i,j} \times T_{i,j} + k(T_{komsu,j} - r_{i,j} \times T_{i,j}) \times (sin(\pi \times acos(r_{i,j}))), rand \geq ST \tag{17}$$

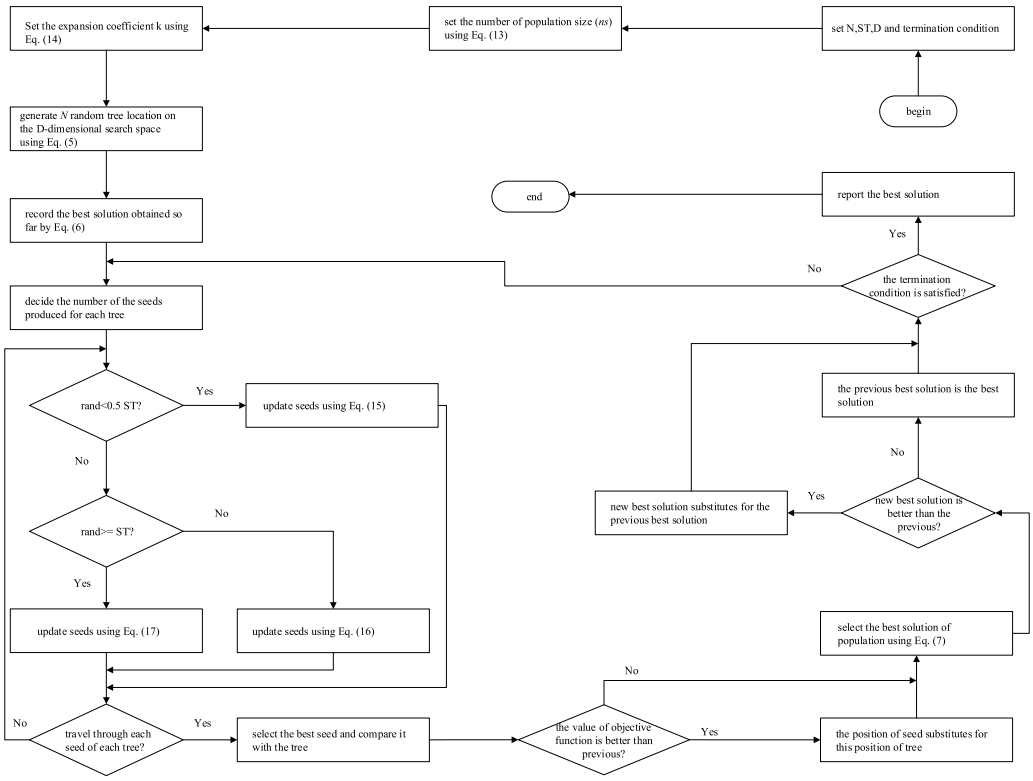


Fig. 4. The flow chart of STSA.

where, $S_{i,j}$ is j_{th} dimension of i_{th} seed that will be produced i_{th} tree, $T_{i,j}$ is the j_{th} dimension of i_{th} tree, B_j is the j_{th} dimension of best tree location obtained so far, $T_{komsu,j}$ is the j_{th} dimension of random tree, $r_{i,j}$ and *tmpRand* is a random number produced for each dimension and location, in range of [0, 1], k is a new parameter based on the original TSA. ST is a control parameter to control search trend. The higher value of ST provides a powerful local search and speed convergence, the lower value of ST causes slow convergence but powerful global search.

After the above three methods, we finally propose STSA. The pseudo code of the STSA is in Algorithm 1. The flow chart of STSA is shown in Fig. 4.

4. Results

The quality of the proposed algorithm's capability is tested by a set of standard test functions. 30 random experiments are performed on benchmark functions.

4.1. Experimental settings

For these experiments, the variants are coded in Matlab R2016a environment under Chinese version of Windows 8.1 operating system, all simulations are run on computer with Intel(R) Core(TM) i3-5010U CPU @ 2.10 GHz and its memory is 8 G.

4.2. Test the influence from ns and control coefficient k

The two improvements mentioned above have been significantly improved in optimization. 24 benchmark functions (see Tables 10–12) are used to test the effect of STSA and TSA. We set $ST = 0.4$, and test each benchmark function 30 times before taking its average value, with 1000 iterations for each time. Here is a comparison of the results of STSA and TSA on 10 dimensions in Table 1.

From Table 1, we can draw the following conclusion: compared with TSA in lower dimensions such as 10 dimensions, STSA has been greatly improved in some benchmark functions. This shows that our improvement of the algorithm is effective.

Algorithm 1 The pseudo code of the STSA.

Step 1. Initialize of the algorithm
1.1 Set the number of population size (ns) using Eq. (13);
1.2 Set the expansion coefficient k using Eq. (14);
1.3 Set the ST parameter for the method;
1.4 Set the dimensionality of the problem;
1.5 Decide the termination condition;
1.6 Generate N random tree locations on the D-dimensional search space using Eq. (5);
1.7 Evaluate the tree location using objective function specified for the problem;
1.8 Select the best solution using Eq. (6).

Step 2. Search with seeds
FOR all trees
2.1 Decide the number of seeds produced for this tree;
2.2 **FOR** all seeds
2.2.1 **FOR** all dimensions
IF(rand $<$ 0.5$\times$ ST)
Update seeds using Eq. (15)
ELSE IF(0.5$\times$ ST $\leq$ rand $<$ ST)
Update seeds using Eq. (16)
ELSE
Update seeds using Eq. (17)
END IF
END IF
END FOR
END FOR
2.3 Select the best seed and compare it with the tree;
2.4 Seed substitutes for this tree, if the seed objective value is better than tree's.
END FOR

Step 3. Select of Best Solution
3.1 Select the best solution of population using Eq. (7);
3.2 New best solution substitutes for the previous best solution, if new best solution is better than the previous best solution;

Step 4. Test Termination condition
Go to Step 2, if termination condition is not met.

Step 5. Report
Report the best solution.

Table 1
Comparing the performance of TSA and STSA with D = 10.

Function	TSA				STSA			
	Best	Mean	Worst	St.dev	Best	Mean	Worst	St.dev
F_1	1.55744E−59	6.14003E−58	3.55019E−57	9.09095E−58	0	**0**	0	0
F_2	5.00507E−37	2.0848E−36	7.5401E−36	1.99062E−36	6.2571E−196	**3.8669E−193**	2.7369E−192	0
F_3	3.66378E−10	1.38991E−08	1.54248E−07	2.93509E−08	9.0833E−263	**6.4491E−252**	1.8427E−250	0
F_4	1.85202E−12	1.50146E−11	7.53349E−11	1.54457E−11	8.2694E−160	**2.7604E−157**	4.2408E−156	7.9269E−157
F_5	1.64035919	**3.697345435**	5.733900666	0.967684642	5.916132819	6.740343539	7.166741671	0.255655294
F_6	0	**0**	0	0	0.007514272	0.019223256	0.033449979	0.006515981
F_7	0.000500271	0.001514898	0.003396011	0.000721624	1.0065E−05	**7.69259E−05**	0.000169027	4.74073E−05
F_8	−4189.828873	−3983.383222	−3240.381841	260.8319443	−3346.980616	**−2949.583932**	−2652.297991	172.4702049
F_9	0.094655686	5.51063386	12.92082052	2.876688408	0	**0**	0	0
F_{10}	8.88178E−16	4.32247E−15	4.44089E−15	6.48634E−16	8.88178E−16	**8.88178E−16**	8.88178E−16	0
F_{11}	0.002337944	0.116339102	0.291006286	0.074541113	0	**0**	0	0
F_{12}	4.71163E−32	**4.71163E−32**	4.71163E−32	0	0.000868631	0.002158034	0.005568348	0.000992842
F_{13}	1.34978E−32	**1.34978E−32**	1.34978E−32	0	0.005471141	0.017193327	0.036532769	0.008346033
F_{15}	0.000346719	0.000531646	0.000715482	9.1999E−05	0.000307934	**0.000413852**	0.001224059	0.000243004
F_{16}	−1.031628453	**−1.031628453**	−1.031628453	0	−1.031628453	**−1.031628453**	−1.031628453	0
F_{18}	3	**3**	3	0	3	**3**	3	0

Table 2
Test 10 dimensions of STSA, TSA, and SCA algorithms on benchmark functions.

Function	STSA		TSA		SCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev
F_1	**0**	0	6.14003E−58	9.09095E−58	7.70011E−12	1.85358E−11
F_2	**3.8669E−193**	0	2.0848E−36	1.99062E−36	6.35004E−10	8.20193E−10
F_3	**6.4491E−252**	0	1.38991E−08	2.93509E−08	0.002785518	0.005503978
F_4	**2.7604E−157**	7.9269E−157	1.50146E−11	1.54457E−11	0.001856347	0.004024226
F_5	6.740343539	0.255655294	**3.697345435**	0.967684642	7.469140732	0.454731885
F_6	0.019223256	0.006515981	**0**	0	0.479891223	0.16558056
F_7	**7.69259E−05**	4.74073E−05	0.001514898	0.000721624	0.002072487	0.001565976
F_8	−2949.583932	172.4702049	−3983.383222	260.8319443	**−2124.717376**	160.8814671
F_9	**0**	0	5.51063386	2.876688408	0.58828547	1.883503053
F_{10}	**8.88178E−16**	0	4.32247E−15	6.48634E−16	3.4911E−06	1.61795E−05
F_{11}	**0**	0	0.116339102	0.074541113	0.097309014	0.147813439
F_{12}	0.002158034	0.000992842	**4.71E−32**	0	0.104672303	0.041780972
F_{13}	0.017193327	0.008346033	**1.35E−32**	0	0.31066643	0.088650469
F_{15}	**0.000413852**	0.000243004	0.000531646	9.1999E−05	0.001172708	0.000344145
F_{16}	**−1.031628453**	0	**−1.031628453**	0	**−1.031628453**	0
F_{18}	**3**	0	**3**	0	**3**	0

Table 3
Test 50 dimensions of STSA, TSA, and SCA algorithms on benchmark functions.

Function	STSA		TSA		SCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev
F_1	**2.09873E−11**	3.7835E−11	37.49125793	12.78842918	852.5265976	1182.869122
F_2	**3.03462E−10**	3.17053E−10	0.99747663	0.307036604	0.561064287	0.713184908
F_3	80961.20259	18903.57364	92970.32715	8703.917066	**47938.296**	13823.48928
F_4	76.59891165	19.65233503	84.40999669	5.214241995	**70.87506431**	6.245928564
F_5	**47.8840535**	0.136216766	925500.7482	397054.1905	8723235.182	12457686.86
F_6	**5.543152566**	0.401943163	40.7600341	10.68622681	1100.511257	1220.611516
F_7	**0.025879662**	0.010220925	1.042994019	0.287275626	5.150171173	4.604588886
F_8	−6403.4687741	400.4018495	−7895.3291	977.0255383	**−4949.936664**	414.19178
F_9	**2.838603442**	0	12.40980909	48.26672581	107.1002834	56.44248435
F_{10}	4.081467613	8.282419073	**3.809060875**	0.251617135	14.82285523	7.851793748
F_{11}	**4.31505E−09**	2.31402E−08	1.360207388	0.109355175	9.992810847	11.28879743
F_{12}	**0.353011421**	0.057884073	4251671.434	2764211.925	11230940	18975323.2
F_{13}	**3.471250384**	0.254658731	7841429.597	6012118.736	33802396.17	64739994.94
F_{15}	0.00053232	0.000265155	**0.000488177**	9.90672E−05	0.000947735	0.00032182
F_{16}	**−1.031628453**	0	**−1.031628453**	0	**−1.031628453**	0
F_{18}	**3**	0	**3**	0	**3**	0

4.3. Comparing test of different dimensions of test function

We find that the new algorithm has different optimization effects in different dimensions. In the higher dimensions, the optimization effect of the algorithm is still obvious. We use 24 benchmark functions to test the effect of STSA, TSA, and SCA. We test STSA, TSA and SCA in 10, 50, 100, 200, and 300 dimensions, and we test each dimension 30 times to take its average value. The results are separately shown in Tables 2–6.

From Tables 2–6, we can draw the conclusions: STSA is better than TSA, especially in low-dimensional optimization, and in high dimensions, although STSA is not as effective as SCA, it is still better than TSA. Meanwhile, STSA is more effective in single-mode function optimization. This shows that STSA is still effective than TSA in solving continuous optimization problems.

4.4. Compare the new algorithm with other algorithms

Not only does STSA have made great progress compared with the original algorithm, but also has obvious progress compared with other algorithms like PSO [22], ABC [25]. PSO, and ABC are all heuristic algorithms with good performance. They are widely used. Because of the randomness of meta-heuristics, the results of a single run may not be reliable, so all algorithms are run for 30 times, and we test benchmark functions in 10 dimensions. The results are shown in Table 7.

From Table 7, we can draw the conclusion: the optimization effect of STSA on the benchmark functions are better than that of other algorithms. This shows that we have achieved remarkable results in improving TSA.

The convergence curve of STSA, TSA, SCA, PSO, and ABC on 10, 50, 100, 200, 300 dimensions are separately reported in Figs. 5–9.

From above figures, the global optimization ability of STSA has been significantly improved. This shows that our improvements are valuable.

Table 4
Test 100 dimensions of STSA, TSA, and SCA algorithms on benchmark functions.

Function	STSA		TSA		SCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev
F_1	**5467.962537**	3438.477572	51565.08976	4810.357842	9250.26223	6073.209931
F_2	**3.633826105**	2.532717612	204.3450684	37.63593441	7.468276657	6.244409597
F_3	509247.4596	56665.83854	398230.1358	36269.64227	**249784.7447**	59236.27654
F_4	96.2380303	0.866394223	95.95709659	1.42110836	**89.46187628**	2.899588535
F_5	**50469533.68**	28436101.11	263741942.3	56196682.18	119258008.1	47430327.46
F_6	**4911.412299**	3454.70114	51815.36649	6176.019791	10926.35155	6206.509727
F_7	**58.0563075**	36.61767141	320.3909558	55.51799156	129.4509679	59.11736272
F_8	−8939.25696	533.4857467	−10368.32823	1204.014387	**−6718.38097**	530.5657556
F_9	**213.4134681**	97.13097828	1169.298674	81.94375627	249.3109602	129.9299456
F_{10}	**17.07525135**	5.655108573	18.22031237	0.428081552	17.85265813	5.195248554
F_{11}	**61.29852316**	38.29718176	468.4535642	47.25803699	101.3852225	61.73895429
F_{12}	**158476922.2**	105589585.6	675330418.1	159508288.6	309013376.1	110442212.1
F_{13}	**288519075.1**	161102281.7	1199863744	288763141.5	460173397.1	221885759.3
F_{15}	**0.00045001**	0.000186862	0.000488557	9.96379E−05	0.0011668	0.00034868
F_{16}	**−1.031628453**	0	**−1.031628453**	0	**−1.031628453**	0
F_{18}	**3**	0	**3**	0	**3**	0

Table 5
Test 200 dimensions of STSA, TSA, and SCA algorithms on benchmark functions.

Function	STSA		TSA		SCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev
F_1	106539.8055	32078.7322	280383.5464	24264.04951	**52249.82609**	23828.13566
F_2	97.65208347	30.24910941	2.74822E+52	1.46845E+53	**26.34557401**	15.22939772
F_3	2131794.722	180021.6476	1540711.149	112226.3211	**1058993.12**	232308.7127
F_4	97.99938188	0.515363119	97.88989453	0.682597149	**96.66564669**	1.08900964
F_5	1105918860	497927415.4	1397123788	230413291.6	**553538179.7**	176614042
F_6	95010.25467	34491.05209	276936.34	24295.68444	**54203.27135**	20295.24166
F_7	2191.033497	640.849558	4135.572784	579.8152006	**1433.083517**	406.5409284
F_8	−12548.39507	629.5425287	−14343.07021	1865.408198	**−9783.007234**	714.0327655
F_9	766.1132969	179.3715474	2799.471358	95.86368742	**586.0654747**	188.5510106
F_{10}	19.68726044	2.200303927	20.80892902	0.058857862	**18.75391317**	3.667143195
F_{11}	820.8076127	297.9606722	2552.040832	265.1953845	**423.0126078**	160.9687748
F_{12}	6203837502	685035823	4926406306	1084225942	**1587977619**	452102461.2
F_{13}	8616989550	3347208545	7807117976	1521013375	2345670145	561994126.6
F_{15}	0.000485675	0.00025835	**0.000482951**	8.11607E−05	0.000961799	0.000316503
F_{16}	**−1.031628453**	0	**−1.031628453**	0	**−1.031628453**	0
F_{18}	**3**	0	**3**	0	**3**	0

Table 6
Test 300 dimensions of STSA, TSA, and SCA algorithms on benchmark functions.

Function	STSA		TSA		SCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev
F_1	225214.432	56109.35087	533513.9013	39739.46323	**86941.87349**	31912.37329
F_2	217.0273672	65.70356239	9.67345E+97	3.04111E+98	**56.35112038**	40.40624911
F_3	4722365.33	392470.5158	3545991.6	261546.5356	**2292870.862**	466658.8776
F_4	98.5043439	0.607144679	98.76008006	0.485173586	**98.09193925**	0.48878671
F_5	4094151970	169338723.5	3666050169	518124942.2	**968884856.3**	222217502.8
F_6	237179.8654	44463.58628	536911.4214	39806.64417	**94313.09494**	32993.36732
F_7	15406.02675	5088.243232	12895.13472	2458.527172	**4105.620027**	1089.146064
F_8	−15595.01229	917.4856184	−17716.13705	3089.604424	**−11922.97675**	822.4241079
F_9	1393.27575	288.5370256	4445.925961	195.2156154	**748.931034**	320.5253141
F_{10}	20.68731967	0.459986188	20.93341125	0.013978674	**19.23194496**	3.520950386
F_{11}	2162.238113	566.8342169	4930.178318	372.2067111	**904.1766518**	388.554181
F_{12}	10100024289	638625398.1	10015910977	637445278.3	**2949377837**	479856939
F_{13}	18513189539	899565205.6	17619707398	1641081435	**4723637462**	1329157586
F_{15}	0.000485569	0.000217932	**0.000465554**	9.5613E−05	0.001102449	0.000375453
F_{16}	**−1.031628453**	0	**−1.031628453**	0	**−1.031628453**	0
F_{18}	**3**	0	**3**	0	**3**	0

Table 7
Comparisons between STSA and other algorithms on 10 dimensions.

Functions	Index	STSA	TSA	SCA	PSO	ABC
F_1	Mean	**0**	6.14003E−58	7.70011E−12	1.34177E−71	3.13494E−05
	St.dev	0	9.09095E−58	1.85358E−11	5.3328E−71	3.19179E−05
F_2	Mean	**3.8669E−193**	2.0848E−36	6.35004E−10	1.57447E−12	6.84443E−06
	St.dev	0	1.99062E−36	8.20193E−10	7.8476E−12	4.99614E−06
F_3	Mean	**6.4491E−252**	1.38991E−08	0.002785518	8.33078E−16	176.5269364
	St.dev	0	2.93509E−08	0.005503978	2.83888E−15	82.67860293
F_4	Mean	**2.7604E−157**	1.50146E−11	0.001856347	1.1284E−12	3.539160325
	St.dev	**7.9269E−157**	1.54457E−11	0.004024226	2.5212E−12	0.935265519
F_5	Mean	6.740343539	**3.697345435**	7.469140732	6.46636785	21.77400175
	St.dev	0.255655294	0.967684642	0.454731885	16.7846427	23.04898671
F_6	Mean	0.019223256	**0**	0.479891223	1.00457E−31	3.60768E−05
	St.dev	0.006515981	0	0.16558056	3.82709E−31	4.09459E−05
F_7	Mean	**7.69259E−05**	0.001514898	0.002072487	0.003044067	0.012534387
	St.dev	4.74073E−05	0.000721624	0.001565976	0.001567805	0.005778564
F_8	Mean	−2949.583932	−3983.383222	**−2124.717376**	−2605.830585	−5.35659E+61
	St.dev	172.4702049	260.8319443	160.8814671	359.6441894	2.9112E+62
F_9	Mean	**0**	5.51063386	0.58828547	13.13344544	30.44898533
	St.dev	0	2.876688408	1.883503053	5.897249862	4.461725323
F_{10}	Mean	**8.88178E−16**	4.32247E−15	3.4911E−06	6.33567E−15	0.025958926
	St.dev	0	6.48634E−16	1.61795E−05	1.8027E−15	0.015607878
F_{11}	Mean	**0**	0.116339102	0.097309014	0.092887273	0.416745077
	St.dev	0	0.074541113	0.147813439	0.049843162	0.08225128
F_{12}	Mean	0.002158034	**4.71E−32**	0.104672303	5.19164E−32	0.000728654
	St.dev	0.000992842	0	0.041780972	1.5417E−32	0.001932377
F_{13}	Mean	0.017193327	**1.35E−32**	0.31066643	0.000732491	0.00417452
	St.dev	0.008346033	0	0.088650469	0.002787584	0.005670945
F_{15}	Mean	**0.000413852**	0.000531646	0.001172708	0.001254674	0.001075483
	St.dev	0.000243004	0.000531646	0.001172708	0.001254674	0.001075483
F_{16}	Mean	**−1.031628453**	**−1.031628453**	**−1.031628453**	**−1.031628453**	**−1.031628453**
	St.dev	0	0	0	0	0
F_{18}	Mean	**3**	**3**	**3**	**3**	**3**
	St.dev	0	0	0	0	0

5. Application of STSA in engineering design problems

This section further verifies the performance and efficiency of the STSA by solving two constrained real engineering design problems: pressure vessel design, and tension/compression spring design. These problems are widely discussed in the literature, and they have been solved well to clarify the effectiveness of the algorithms. We apply the penalty function method to deal with the constraints. In STSA, the population size is 30 and the maximum number of iterations is 1000.

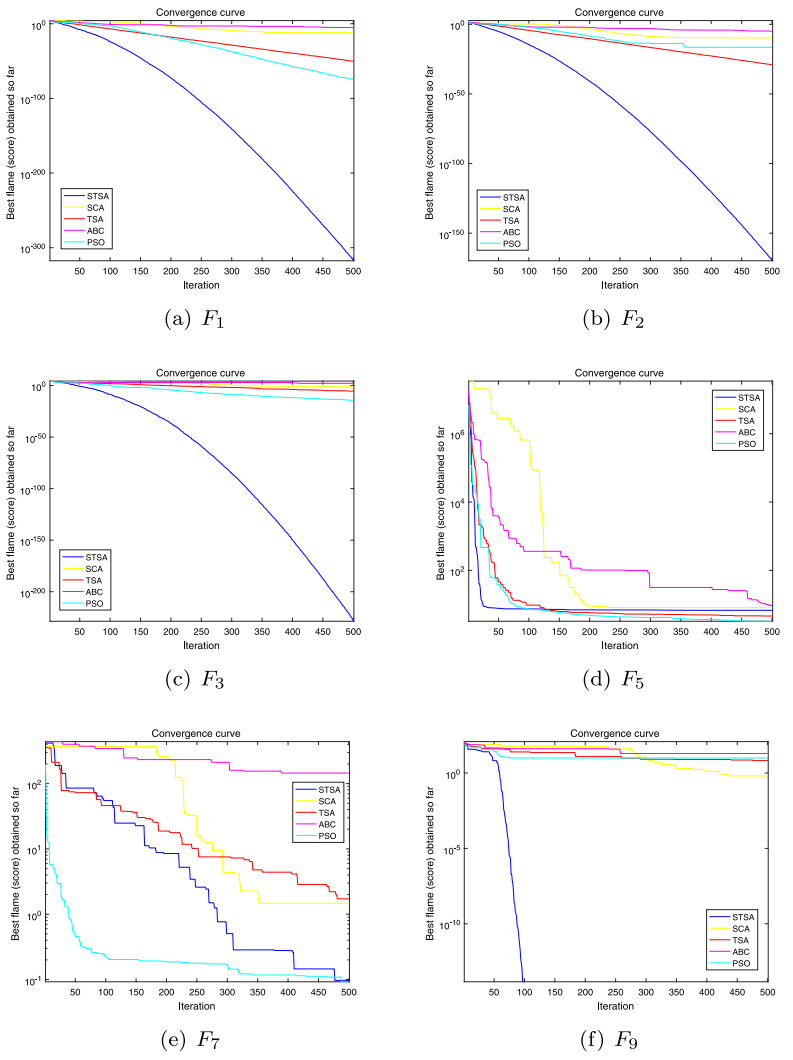
5.1. Pressure Vessel Design (PVD) problem

5.1.1. Introduction to the PVD problem

The aim of the PVD problem is to minimize the whole cost of the cylindrical pressure vessel [38]. This problem has four variables: the thickness of the shell (T_s), thickness of the head (T_h), inner radius (R), and length of the cylindrical section without considering the head (L), as shown in Fig. 10. It is expected to be that T_s and T_h are in multiplies of 0.0625 inches. R and L are continuous values for PVD problem.

The mathematical formulations of pressure vessel design problem are defined as follows:

$$
\begin{aligned}
&\textit{Consider}\quad X = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L]\\
&\textit{Minimize}\quad f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3\\
&\textit{Subjectto}\quad g_1(X) = -x_1 + 0.0193x_3 \le 0\\
&\qquad\qquad g_2(X) = -x_2 + 0.00954 \le 0\\
&\qquad\qquad g_3(X) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0\\
&\qquad\qquad g_4(X) = x_4 - 240.0 \le 0\\
&\qquad\qquad g_5(X) = -x_1 + 1.1 \le 0\\
&\qquad\qquad g_6(X) = -x_2 + 0.6 \le 0\\
&\textit{Where}\quad 1.1 \le x_1 \le 99, 0.6 \le x_2 \le 99, 10 \le x_3 \le 200, 10 \le x_4 \le 240
\end{aligned}
$$



(a) F_1 (b) F_2

(c) F_3 (d) F_5

(e) F_7 (f) F_9

Fig. 5. The convergence curve of STSA and other algorithms on 10 dimensions.

The objective of the PVD problem is to find X values that minimizes the cost of $f(X)$ under g_1, g_2, g_3, g_4, g_5, and g_6 constraints. The other constraints are in design code of PVD problem and detailed explanations can be seen in the algorithm: New optimization techniques in engineering [39].

5.1.2. Application of STSA in PVD problem

To cope with the constraints of the PVD problem, a penalty function is used. If a constraint is violated, a penalty value is added to the objective function [40].

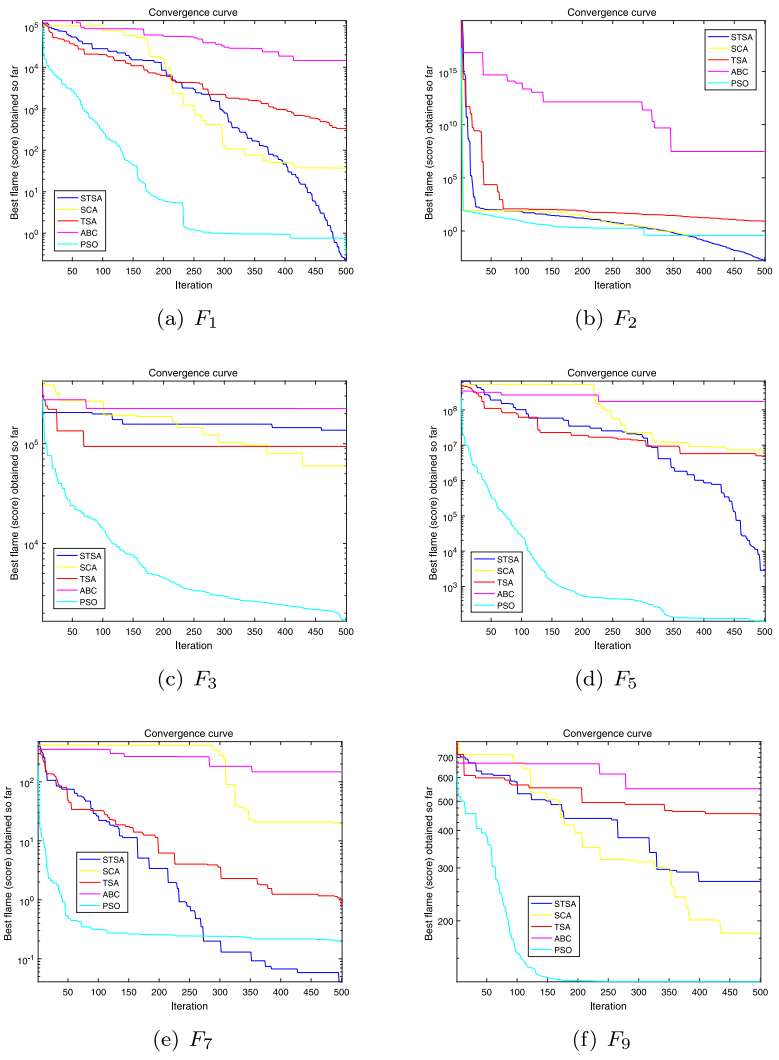
Let be $x_2 = 0.2$, g_6 constraint function is calculated as 0.4, and the constraint is therefore violated. New $f(X)$ is obtained by using Eq. (18).

$$n \times F(X) = f(X) + \sum_{1}^{6} v_i \tag{18}$$

where, v_i is the violation of i_{th} constraint and is calculated as follows:

$$v_i(X) = \begin{cases} k \times g_i(X)^2 & if \quad (g_i(X) > 0) \\ 0 & otherwise \end{cases} \tag{19}$$

where, k is a high positive constant number.



(a) F_1 (b) F_2

(c) F_3 (d) F_5

(e) F_7 (f) F_9

Fig. 6. The convergence curve of STSA and other algorithms on 50 dimensions.

By considering STSA, PVD problem and explanations given above, the pseudo code of the application of STSA to PVD problem is presented in Algorithm 2.

Algorithm 2 The pseudo code of the application of STSA to PVD problem.

Step 1. Initialize of the algorithm
Step 2. UNTIL a termination condition is met
2.1 **FOR** each tree
2.1.1 Produce seeds from the tree by using Eqs. (15), (16) and (17)
2.1.2 Calculate fitness of the seeds by using Eq. (18)
2.1.3 **IF** fitness of the best seeds is better than the fitness of the tree
2.1.4 **THEN**
2.1.5 **Begin**
2.1.5.1 Remove the tree from the stand
2.1.5.2 Add the seed to the stand
END IF
END FOR
2.2 Decide the best tree location
END UNTIL
Step 3. Report the best solution

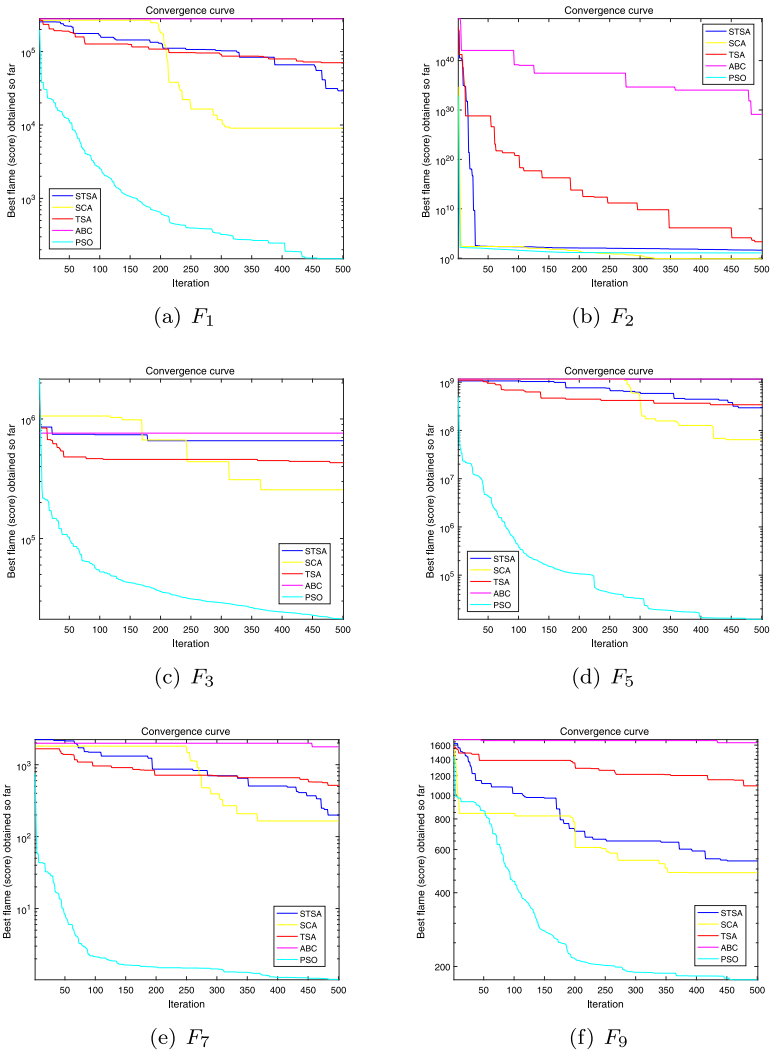


Fig. 7. The convergence curve of STSA and other algorithms on 100 dimensions.

Table 8
The experimental results of STSA and other algorithms for PVD problem.

Algorithm	Optimal values for variables				Mean cost	Std.dev
	T_s	T_h	R	L		
STSA	1.173757082	0.602273164	60.67453944	31.7949056	7191.8	64.5232
TSA	1.1039	0.6284	56.3499	55.2178	7146.0100	51.6371
SCA	1.1479	0.6250	57.0737	55.2968	7534.1766	188.0822
PSO	1.1257	0.6257	58.0142	45.5515	7236.9200	21.9913
ABC	1.1250	0.6250	58.0728	45.1283	7221.7900	15.3953

5.1.3. Application results of STSA and other algorithms

STSA is applied to solve the pressure vessel design problem. We compare it with 4 optimization algorithms which are reported in previews works as shown in Table 8. In order to analyze the performance of STSA on PVD problem, the method is run 30 times with random initialization for each test case and optimal values for variables(T_s, T_h, R, and L), mean cost, and standard deviation are reported.

From Table 8, compared with SCA, PSO, and ABC, the STSA provided a better result for the PVD problem. Compared with tsa, the results of STSA are slightly worse, but the difference is not significant.

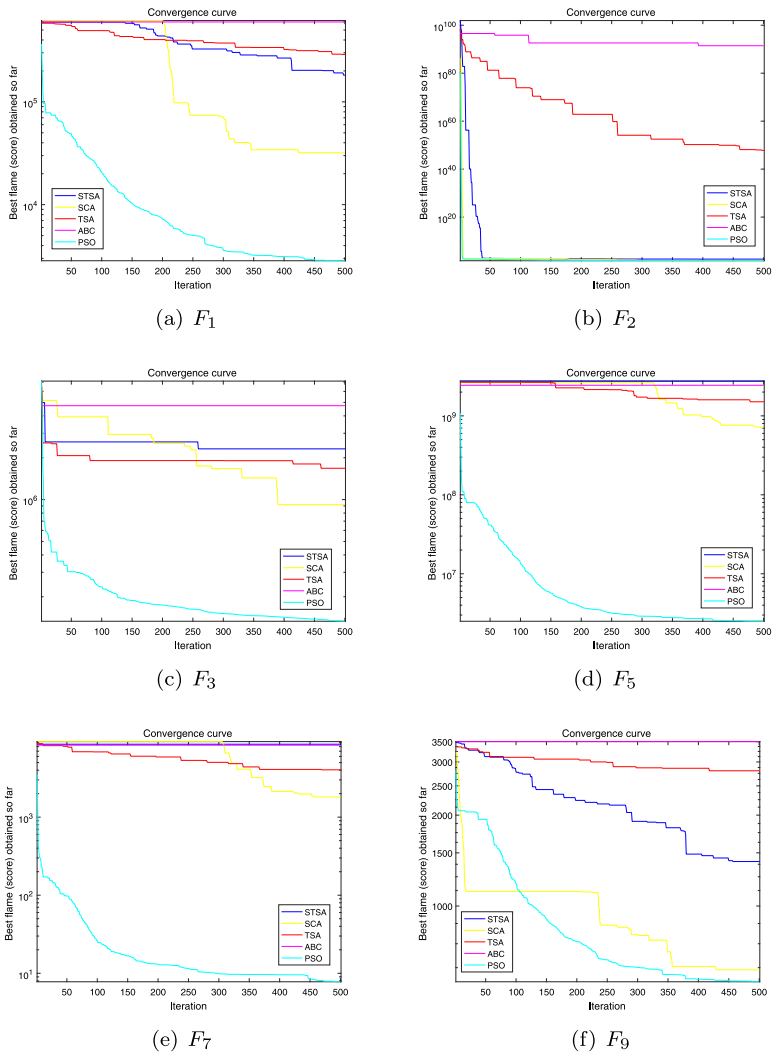


Fig. 8. The convergence curve of STSA and other algorithms on 200 dimensions.

5.2. Tension/Compression Spring Design (T/CSD) problem

The T/CSD problem is shown in Fig. 11, which deals with the minimization of the weight of the tension/compression spring. The aim of this problem is to minimize the weight of the tension/compression spring by determining the optimal value of three variables: the mean coil diameter (D), the number of active coils (N), and wire diameter (d). The problem is formulated as:

$$
\begin{aligned}
&\textit{Consider}\ \ X = [x_1, x_2, x_3] = (d, D, P)\\
&\textit{Minimize}\ f(X) = (x_3 + 2)x_2x_1^2\\
&\textit{Subjectto}\ g_1(X) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0\\
&\qquad g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0\\
&\qquad g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0\\
&\qquad g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \le 0\\
&\textit{Where}\quad 0.05 \le x_1 \le 2.00, 0.25 \le x_2 \le 1.30, 2.00 \le x_3 \le 15.00
\end{aligned}
$$

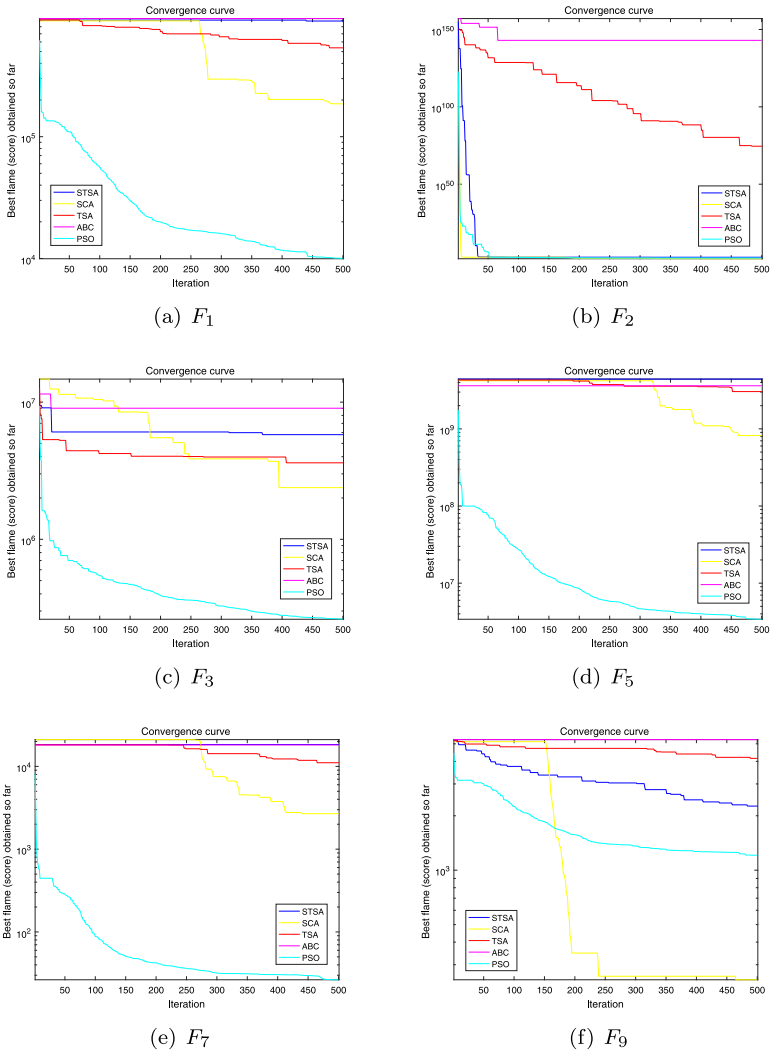


Fig. 9. The convergence curve of STSA and other algorithms on 300 dimensions.

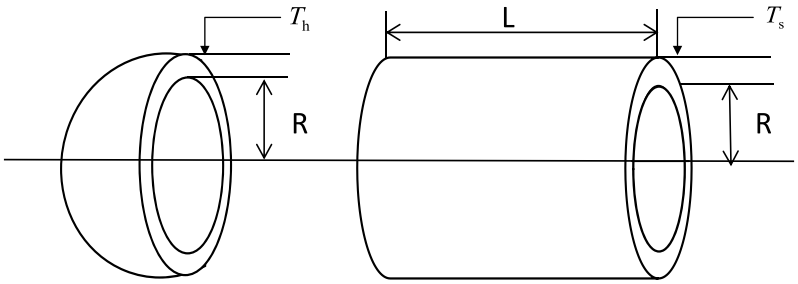


Fig. 10. Schematic view of PVD problem.

The optimization results obtained by the proposed STSA for this problem are evaluated by comparing it with TSA, SCA, PSO, and ABC, as shown in Table 9. In order to analysis the performance of STSA on T/CSD problem, the method is run 30 times with random initialization for each test case. Optimal values for variables(d, D, and N), mean cost, and standard deviation are reported. From Table 9, compared with other algorithms, the STSA obtain better results for the T/CSD problem.

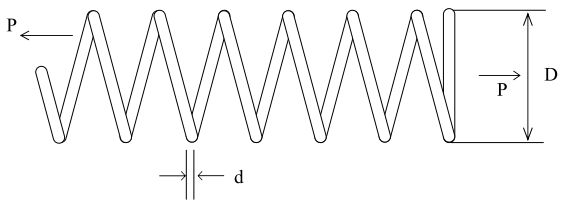


Fig. 11. Schematic view of T/CSD problem.

Table 9
The experimental results of STSA and other algorithms for pressure vessel design problem.

Algorithm	Optimal values for variables			Mean cost	Std.dev
	d	*D*	*N*		
STSA	0.050064347	0.480649917	4.068127065	0.00734579	5.44585E−05
TSA	0.090476838	0.913401821	8.1	0.020819027	4.33E−03
SCA	0.0604564	0.602142333	7.05075	0.0170221	3.01E−03
PSO	0.056971033	0.507188333	6.8394	0.0134377	1.10E−03
ABC	0.052481	0.375246	11.49661667	0.013254933	1.73E−04

Table 10
Unimodal benchmark functions.

Function	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	10	[−100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[−100,100]	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	10	[−100,100]	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	10	[−100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2) + (x_i - 1)^2]$	10	[−100,100]	0
$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	10	[−100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	10	[−100,100]	0

Table 11
Multimodal benchmark functions.

Function	Dim	Range	*f_min*
$f_8 = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	20	[−500,500]	−418.98295
$f_9 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	20	[−5.12,5.12]	0
$f_{10} = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	20	[−32,32]	0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	20	[−600,600]	0
$f_{12} = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	20	[−50,50]	0
$f_{13} = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [\sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	20	[−50,50]	0

6. Discussion

Through all the above experiments and results, the following analysis can be obtained.

6.1. Analysis of seed production mechanism

It is unreasonable when the number of seeds is random. Studies have shown that the results are optimal when there are an extreme number of seeds. However, the number of seeds is so large that it falls into the local optimum and ignore

Table 12
Composite benchmark functions.

Function	Dim	Range	f_{min}
$f14(CF1)$:			
$f_1, f_2, f_3, \ldots, f_10 = SphereFunction$	10	$[-5,5]$	0
$[\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_10] = [1, 1, 1, \ldots, 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_10] = [5/100, 5/100, 5/100, \ldots, 5/100]$			
$f15(CF2)$:			
$f_1, f_2, f_3, \ldots, f_{10} = Griewank'sFunction$	10	$[-5,5]$	0
$[\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_10] = [1, 1, 1, \ldots, 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_10] = [5/100, 5/100, 5/100, \ldots, 5/100]$			
$f16(CF3)$:			
$f_1, f_2, f_3, \ldots, f_{10} = Griewank'sFunction$	10	$[-5,5]$	0
$[\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_10] = [1, 1, 1, \ldots, 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_10] = [1, 1, 1, \ldots, 1]$			
$f17(CF4)$:			
$f_1, f_2 = Ackley'sFunction$			
$f_3, f_4 = Rastrigin'sFunction$			
$f_5, f_6 = WeierstrassFunction$	10	$[-5,5]$	0
$f_7, f_8 = Griewank'sFunction$			
$f_9, f_{10} = SphereFunction$			
$[\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_10] = [1, 1, 1, \ldots, 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_10] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$			
$f18(CF5)$:			
$f_1, f_2 = Rastrigin'sFunction$			
$f_3, f_4 = WeierstrassFunction$			
$f_5, f_6 = Griewank'sFunction$	10	$[-5,5]$	0
$f_7, f_8 = SphereFunction$			
$f_9, f_{10} = Ackley'sFunction$			
$[\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_10] = [1, 1, 1, \ldots, 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_10] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$			
$f19(CF6)$:			
$f_1, f_2 = Rastrigin'sFunction$			
$f_3, f_4 = WeierstrassFunction$			
$f_5, f_6 = Griewank'sFunction$			
$f_7, f_8 = Ackley'sFunction$	10	$[-5,5]$	0
$f_9, f_{10} = SphereFunction$			
$[\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_10] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_10] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5,$			
$0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$			

the global optimum. The number of seeds is too small to find the optimal solution. The seed renewal mechanism proposed in this paper effectively solves this problem. The number of seeds generates by the renewal changes from more to less varying with the increase of iteration times, and finally reaches a good state.

6.2. Analysis of balance mechanism

Exploration and exploitation are mentioned in TSA, but there is no corresponding mechanism to balance exploration and exploitation, they have a conflict in that the realization of one means the sacrifice of the other. Whether the exploration is too strong or the development ability is too strong, it is not conducive to finding the optimal solution. Inspired by the equilibrium parameters in SCA, we propose the balance parameter k to balance exploration and exploitation. When the exploration ability is strong, the development ability should be improved appropriately. On the contrary, when the exploration ability is strong, the exploration ability should be improved so as to make the process of finding the optimal solution more smoothly.

6.3. Analysis of the shortcomings of STSA

STSA uses 24 benchmark functions to test, including single-mode benchmark functions and multi-mode benchmark functions. According to the experimental results, we can find the shortcomings of STSA: the results of optimization of multi-mode functions are not satisfactory. In the same dimension, the optimization effect is not as good as that of single-mode function.

In addition, STSA is better than TSA in solving high-dimensional complex optimization problems, but not as good as SCA. This shows that we still need further exploration and improvement in high-dimensional complex optimization.

7. Conclusion and future works

TSA is a continuous optimization algorithm with good performance [20,34–36]. However, TSA has two problems. For example, in TSA, seed production mechanism has a certain effect on the experimental results. The number of seeds produced randomly makes the results very uncertain. As assumed in this paper, seed production mechanism should be redefined and redesigned. In addition, exploration and exploitation play a very important role in global search, and they conflict with each other. But TSA lacks a balancing mechanism for exploration and exploitation.

Inspired by SCA, two improvements are proposed, so that the optimal solution can be obtained more effectively. As shown in Table 1, after the above two improvements, the optimization effect of STSA has been significantly improved. Hence, there are some findings in this paper.

- Adaptive seeds generation mechanism will help to find the global optimal solution in TSA.
- The addition of regulate mechanism k linearly varying with iteration times has a positive effect on finding the global optimal solution.

This paper proposes two methods to improve the optimization capability of TSA. However, the algorithm still has some inadequacy, such as when dealing with complex continuous optimization problems, the effect is not obvious and needs to be improved. In fact, from this perspective, more improvements can be presented because more heuristic methods can be found by seed evaluation. It is necessary for us to carry out further research.

References

[1] S. Kirkpatrick, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680, http://dx.doi.org/10.1142/9789812799371_0035.
[2] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.
[3] J. Du, Y. Yuan, T. Si, J. Lian, H. Zhao, Customized optimization of metabolic pathways by combinatorial transcriptional engineering, Nucleic Acids Res. 40 (18) (2012) 177–209, http://dx.doi.org/10.1093/nar/gks549.
[4] S. Raghavan, C. Woon, A. Kraus, K. Megerle, H. Pham, J. Chang, Optimization of human tendon tissue engineering: synergistic effects of growth factors for use in tendon scaffold repopulation, Plast. Reconstr. Surg. 129 (2) (2012) 479–489, http://dx.doi.org/10.1097/PRS.0b013e31823aeb94.
[5] R. Bodade, C. Khobragade, S. Arfeen, Optimization of culture conditions for glucose oxidase production by a penicillium chrysogenum SRT 19 strain, Eng. Life Sci. 10 (1) (2010) 35–39, http://dx.doi.org/10.1002/elsc.200900030.
[6] N. Chen, W. Lu, R. Chen, C. Li, P. Qin, Chemometric methods applied to industrial optimization and materials optimal design, Chemometr. Intell. Lab. Syst. 45 (1) (1999) 329–333, http://dx.doi.org/10.1016/s0169-7439(98)00139-7.
[7] S. Jang, H. Ryoo, S. Ahn, J. Kim, G. Rim, Development and optimization of high-voltage power supply system for industrial magnetron, IEEE Trans. Ind. Electron. 59 (3) (2012) 1453–1461, http://dx.doi.org/10.1109/tie.2011.2163915.
[8] J. Gordon, A. Rabl, Design, analysis and optimization of solar industrial process heat plants without storage, Sol. Energy 28 (6) (1982) 519–530, http://dx.doi.org/10.1016/0038-092X(82)90323-1.
[9] A. Arvay, J. French, J. Wang, X. Peng, A. Kannan, Nature inspired flow field designs for proton exchange membrane fuel cell, Int. J. Hydrogen Energy 38 (9) (2013) 3717–3726, http://dx.doi.org/10.1016/j.ijhydene.2012.12.149.
[10] K. Lee, Z. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Engrg. 194 (3638) (2005) 3902–3933, http://dx.doi.org/10.1016/j.cma.2004.09.007.
[11] M. Nawaz, E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, Omega-Int. J. Manage. Sci. 11 (1) (1983) 91–95, http://dx.doi.org/10.1016/0305-0483(83)90088-9.
[12] J. Holland, J. Reitman, Cognitive systems based on adaptive algorithms, ACM SIGART Bull. (1977) 49, http://dx.doi.org/10.1145/1045343.1045373.
[13] M. Moradi, S. Parsa, An evolutionary method for community detection using a novel local search strategy, Physica A 523 (2019) 457–475, http://dx.doi.org/10.1016/j.physa.2019.01.133.
[14] Y. Yang, L. Tu, K. Li, T. Guo, Optimized inter-structure for enhancing the synchronizability of interdependent networks, Physica A 521 (2019) 310–318, http://dx.doi.org/10.1016/j.physa.2019.01.082.
[15] B. Yang, S. Yang, J. Zhang, D. Li, Optimizing random searches on three-dimensional lattices, Physica A 501 (2018) 120–125, http://dx.doi.org/10.1016/j.physa.2018.02.100.
[16] Y. Wang, H. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247, http://dx.doi.org/10.1016/j.asoc.2014.01.038.
[17] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66, http://dx.doi.org/10.1109/TEVC.2010.2087271.
[18] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inform. Sci. 185 (2012) 153–177, http://dx.doi.org/10.1016/j.ins.2011.09.001.
[19] D. Simon, D. Du, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713, http://dx.doi.org/10.1109/TEVC.2008.919004.
[20] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Engrg. 186 (2) (2000) 311–338, http://dx.doi.org/10.1016/s0045-7825(99)00389-8.

[21] M. Dorigo, L. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66, http://dx.doi.org/10.1109/4235.585892.
[22] C. Tsai, K. Huang, c. Yang, M. Chiang, A fast particle swarm optimization for clustering, Soft Comput. 19 (2) (2015) 321–338, http://dx.doi.org/10.1007/s00500-014-1255-3.
[23] J. Tang, R. Zhang, Y. Yao, F. Yang, Z. Zhao, R. Hu, Y. Yuan, Identification of top-k influential nodes based on enhanced discrete particle swarm optimization for influence maximization, Physica A 513 (2019) 477–496, http://dx.doi.org/10.1016/j.physa.2018.09.040.
[24] H. Marouani, Y. Fouad, Particle swarm optimization performance for fitting of levy noise data, Physica A 514 (2019) 708–714, http://dx.doi.org/10.1016/j.physa.2018.09.137.
[25] D. Karaboga, C. Ozturk, A novel clustering approach: Artificial Bee Colony (ABC) algorithm, Appl. Soft Comput. 11 (1) (2011) 652–657, http://dx.doi.org/10.1016/j.asoc.2009.12.025.
[26] J. Jiang, Y. Feng, J. Zhao, K. Li, DateABC: a fast ABC based energy-efficient live VM consolidation policy with data-intesive energy evalation model, Future Gener. Comput. Syst. 74 (2017) 132–141, http://dx.doi.org/10.1016/j.future.2016.05.013.
[27] X. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-inspir. Comput. 2 (2) (2010) 78–84, http://dx.doi.org/10.1504/IJBIC.2010.032124.
[28] E. Rashedi, H. Nezamabadipour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.
[29] A. Kaveh, V. Mahdavi, A hybrid CBO-PSO algorithm for optimal design of truss structures with dynamic constraints, Appl. Soft Comput. 34 (2015) 260–273, http://dx.doi.org/10.1016/j.asoc.2015.05.010.
[30] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inform. Sci. 222 (2013) 175–184, http://dx.doi.org/10.1016/j.ins.2012.08.023.
[31] A.H. Kashan, League Championship Algorithm (LCA): an algorithm for global optimization inspired by sport championships, Appl. Soft Comput. 16 (2014) 171–200.
[32] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems, Inform. Sci. 13 (2013) 2592–2612, http://dx.doi.org/10.1016/j.asoc.2012.11.026.
[33] R. Rao, V. Savsani, D. Vakharia, Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (3) (2011) 303–315, http://dx.doi.org/10.1016/j.cad.2010.12.015.
[34] M. Kiran, TSA: tree-seed algorithm for continuous optimization, Expert Syst. Appl. 42 (19) (2015) 6686–6698, http://dx.doi.org/10.1016/j.eswa.2015.04.055.
[35] A. Babalik, A. Cinar, M. Kiran, A modification of tree-seed algorithm using Deb's rules for constrained optimization, Appl. Soft Comput. 63 (2018) 289–305.
[36] M. Aslan, M. Beskirli, H. Kodaz, M. Kiran, An improved tree seed algorithm for optimization problems, Int. J. Mach. Learn. Comput. 8 (1) (2018) 20–25.
[37] M. Seyedali, SCA: a sine cosine algorithm for solving optimization problems, Expert Syst. Appl. 96 (2015) 120–133, http://dx.doi.org/10.1016/j.knosys.2015.12.022.
[38] W. Long, T. Wu, X. Liang, S. Xu, Solving high-dimensional global optimization problems using an improved sine cosine algorithm, Expert Syst. Appl. 123 (2019) 108–126, http://dx.doi.org/10.1016/j.eswa.2018.11.032.
[39] G. Onwubolu, B. Babu, New Optimization Techniques in Engineering, Springer Berlin Heidelberg, 2004, pp. 150–153, http://dx.doi.org/10.1007/978-3-540-39930-8.
[40] M. Kıran, An Implementation of Tree-Seed Algorithm (TSA) for Constrained Optimization, 2016, pp. 189–197, http://dx.doi.org/10.1007/978-3-319-27000-5_15.