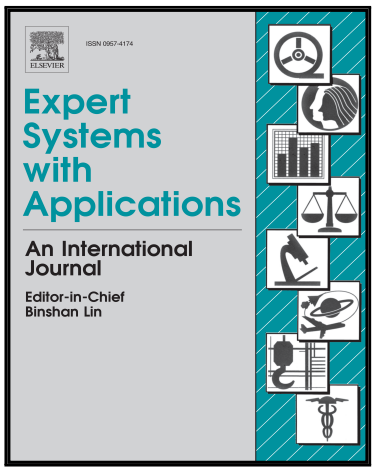


Solving high-dimensional global optimization problems using an improved sine cosine algorithm

Wen Long , Tiebin Wu , Ximing Liang , Songjin Xu

PII: S0957-4174(18)30752-8
DOI: https://doi.org/10.1016/j.eswa.2018.11.032
Reference: ESWA 12331

To appear in: *Expert Systems With Applications*

Received date: 29 May 2018
Revised date: 26 October 2018
Accepted date: 24 November 2018

Please cite this article as: Wen Long , Tiebin Wu , Ximing Liang , Songjin Xu , Solving high-dimensional global optimization problems using an improved sine cosine algorithm, *Expert Systems With Applications* (2018), doi: https://doi.org/10.1016/j.eswa.2018.11.032

Highlights

- Sine cosine algorithm is modified to solve high-dimensional optimization problem.
- A modified position-updating equation is presented to accelerate convergence.
- A novel nonlinear decreasing conversion parameter strategy is designed to balance exploration and exploitation.
- The performance of ISCA is tested on 24 high-dimensional functions and three engineering design problem.
- Comparisons demonstrate that ISCA is a competitive solver for high-dimensional optimization problems.

* Corresponding author.
E-mail address: longwen227@mail.gufe.edu.cn (W. Long), wutiebin81@163.com (T. Wu), dragon638@126.com (X. Liang), trxyxsj_72@163.com (S. Xu).

Solving high-dimensional global optimization problems using an improved sine cosine algorithm

Wen Long[a], Tiebin Wu[b], Ximing Liang[c], Songjin Xu[d*]

[a] *Key Laboratory of Economics System Simulation, Guizhou University of Finance and Economics, Guiyang* 550025, *China*
[b] *School of Energy and Electrical Engineering, Hunan University of Humanities Science and Technology, Loudi Hunan* 417000, *China*
[c] *School of Science, Beijing University Civil Engineering and Architecture, Beijing* 100044, *China*
[d] *School of Big Data, Tongren University, Tongren Guizhou* 554300, *China*

Received 29 May 2018

***Key words*:** Sine cosine algorithm; High-dimensional global optimization; Inertia weight; Engineering design optimization

Abstract The sine cosine algorithm (SCA) is a relatively novel population-based optimization technique that has been proven competitive with other algorithms and it has received significant interest from researchers in different fields. However, similar to other population-based algorithms, SCA tends to be trapped in local optima and unbalanced exploitation. Additionally, to our limited knowledge, the present SCA and its variants have not been applied to the high-dimensional global optimization problems. This paper presents an improved version of the SCA (ISCA) for solving high-dimensional problems. A modified position-updating equation by introducing inertia weight is proposed to accelerate convergence and avoid falling into the local optima. In addition, to balance the exploration and exploitation of the SCA, we present a new nonlinear conversion parameter decreasing strategy based on the Gaussian function. The effectiveness of the proposed ISCA is evaluated using 24 benchmark high-dimensional (D = 30, 100, 500, 1000, and 5000) functions, large-scale global optimization problems from the IEEE CEC2010 competition, and several real-world engineering applications. The comparisons show that the proposed ISCA can better escape from local optima with faster convergence than both the traditional SCA and other population-based algorithms.

1. Introduction

High-dimensional global optimization problems are increasingly encountered in practical applications (Tuo, Zhang, Yong, Yuan, Liu, Xu, & Deng, 2015; Huang, Li, & Zhang, 2018; Lastra, Molina, & Benítez, 2015). They are challenging because the search space is extremely large owing to the high dimensionality of the problem (> 100). Without loss of generality, a high-dimensional global optimization problem is formulated as follows:

$$\min/\max \quad F(X) = f(x_1, x_2, ..., x_n) \tag{1}$$

where $X \subseteq R^n$ denotes a decision space with n dimensions, $X = (x_1, x_2, ..., x_n) \in R^n$ is the decision variable vector, $f : X \to R$ represents the objective function, and n is the number of variables.

Two aspects cause high-dimensional global optimization problems being difficult to solve. First, the size of the search domain increases exponentially with dimensionality. However, the addition of computational resources typically results in only a linear increase in the number of performed evaluations per second. Next, although the Euclidean distance is widely used in high-dimensional global optimization, it may not be appropriate for measuring proximity and neighborhoods whose relationships are non-trivial. Many nature-inspired optimization algorithms, such as genetic algorithm (GA) (Tsai, Liu, & Chou, 2008), particle swarm optimization (PSO) (Li, & Yao, 2012), artificial bee colony (ABC) (Gao, Liu, & Huang, 2013), ant colony optimization (ACO) (Ning, Zhang, Zhang, & Zhang, 2018), differential evolution (DE) (Ali, Awad, & Suganthan, 2015), grey wolf optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014), salp swarm algorithm (SSA) (Mirjalili, Gandomi, Mirjalili, Saremi, Faris, & Mirjalili, 2017), ageist spider monkey optimization (ASMO) (Sharma, Sharma, Panigrahi, Kiran, & Kumar, 2016), polar bear optimization (PBO) (Polap & Woźniak, 2017), and squirrel search (SS) (Jain, Singh, & Rani, 2018), are found in swarm intelligence and evolutionary computation literatures. The primary advantage of these algorithms is their use of the "trial-and-error" principle in searching for solutions. Thus, these algorithms were successfully applied to solve global optimization problems.

* Corresponding author.
E-mail address: longwen227@mail.gufe.edu.cn (W. Long), wutiebin81@163.com (T. Wu), dragon638@126.com (X. Liang), trxyxsj_72@163.com (S. Xu).

In this paper, we focus on the sine cosine algorithm (SCA), which was developed by Seyedali Mirjalili (2016) based on simulating the behaviors of sine and cosine mathematical functions. The SCA has been tested with numerous benchmark functions, achieving good performance in comparison with similar algorithms (Mirjalili, 2016; Elaziz, Oliva, & Xiong, 2017). Recently, the SCA has also been applied for in different real-world problems such as object tracking (Nenavath & Jatoth, 2017), engineering design problems (Rizk-Allah, 2017), multi-objective engineering design problems (Tawhid & Savsani, 2017), feature selection in machine learning (Sindhu, Ngadiran, & Yacob, 2017), parameter optimization (Li, Fang, & Liu, 2018), pairwise local sequence alignment (Issa, Hassanien, Oliva, Helmi, Ziedan, & Alzohairy, 2018), short-term hydrothermal scheduling (Das, Bhattacharya, & Chakraborty, 2017), and wind speed forecasting (Wang, Yang, Du, & Niu, 2018). However, similar to other population-based algorithms, as the search space dimension increases, the SCA also encounters some challenging problems. For example, the convergence speed of the SCA is typically slower than those of representative population-based algorithms such as the whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016) when handling unimodal problems. Furthermore, the SCA can easily become trapped in the local optima when solving complex multimodal problems. Therefore, avoiding the local optima and accelerating convergence speed have become the two most important goals in SCA research. Hence, a number of SCA variants have been developed to achieve these two goals. In (Elaziz, Oliva, & Xiong, 2017), the opposition-based-learning strategy was introduced as a mechanism for a better exploration of the search space, generating more accurate solutions and proposed an improved version of SCA to solve global optimization problems. Meshkat & Parhizgar (2017b) presented a novel SCA to solve global optimization problems. In the proposed algorithm, the position update rule for each search agent was determined by two coefficients, namely the exploration rate and exploitation rate which provided an appropriate balance between the exploration and exploitation phases. Rizk-Allah (2017) proposed a hybrid version of the SCA to solve engineering design problems. The proposed hybrid algorithm operates in two stages. First, the SCA starts the search process to enhance the exploration capability; subsequently, the multi-orthogonal search strategy starts its search from the SCA found thus far to boost the exploitation tendencies.

To our limited knowledge, the present SCA and its variants have not been applied in high-dimensional global optimization problems (dimension > 100). Additionally, to enhance the performance of the SCA, one active research trend is to study its position-updating equation. We found that a well-designed position-updating equation is typically beneficial to enhance the performance of the SCA (Meshkat & Parhizgar, 2017a; Meshkat & Parhizgar, 2017b; Sindhu et al., 2017). The aim of this paper is to present an improved version of the SCA called the ISCA, to solve high-dimensional global optimization problems. In the proposed ISCA, an inertia weight coefficient is introduced to substitute a modified position-updating equation; it can yield more information and produce a promising candidate solution. We use the modified position-updating equation instead of the old one in the SCA. In addition, a new nonlinear conversion parameter decreasing strategy is presented to balance between exploration and exploitation. The proposed ISCA has been extended to solve high-dimensional global optimization problems for dimensions ranging from 30 to 5000 with a constant population size of 30, and a constant iterative number of only 500. The comparisons indicate that in addition to a faster convergence speed, the ISCA achieved the global best with a higher accuracy.

The primary contributions of this paper are summarized as follows:

1) A novel framework of the SCA is proposed that does not affect the configuration of the original SCA.

2) A modified position-updating equation based on inertia weight and a nonlinear conversion parameter strategy based on the Gaussian function is proposed to improve the performance of the SCA.

3) Systematic experiments have shown that the proposed ISCA provides competitive performance on several test suites and real-world engineering applications.

The organization of this paper is as follows. Section 2 reviews the related work on high-dimensional global optimization problems. Section 3 introduces the fundamentals of the SCA. The proposed algorithm is presented in detail in Section 4. The proposed ISCA is substantiated using a wide set of benchmark functions and real-world engineering applications in Section 5. Section 6 concludes this study and presents the future works.

2. Related works

During the past decade, high-dimensional optimization problems have become a popular study area and various population-based algorithms have been applied to solve these problems. Wang and Gao (2014) presented a DE algorithm with a cooperative co-evolutionary selection operator for solving high-dimensional optimization problems. The proposed algorithm designed a local selection operator by decomposing the high-dimensional problem into a number of subproblems and assigning a local fitness function to evaluate each subcomponent. In (Long, Jiao, Liang, & Tang, 2018), an exploration-enhanced grey wolf optimizer (EEGWO) was proposed to solve high-dimensional numerical optimization problems. The proposed EEGWO designed a modified position-updating equation to enhance the exploration and presented a nonlinear control parameter strategy to balance the exploration and exploitation. In (Trivedi, Srinivasan, Biswas, & Reindl, 2015), a novel hybrid GA and differential evolution (hGADE) was proposed to solve high-dimensional unit commitment scheduling problems. In hGADE, a problem specific heuristic initial population generation method and a replacement strategy based on the preservation of in-feasible solutions in the population were incorporated to enhance the search ability of the hybridized variant. Tuo et al. (2015) presented a novel harmony search algorithm based on dynamic dimensionality-reduction adjustment and dynamic fret width strategy for high-dimensional multimodal optimization problems. Li et al. (2015) proposed a hybrid algorithm based on particle swarm and artificial bee colony (PS-ABC) for high-dimensional optimization problems. The proposed PS-ABC combined the local search phase in PSO with two global search phases in ABC for the global optimum. In (Chu, Gao, & Sorooshian, 2011), an innovative evolutionary algorithm called the shuffled complex evolution with principal components analysis-University of California at Irvine (SP-UCI) was introduced to solve high-dimensional global optimization problems. Rodriguez et al. (2013) presented an iterated greedy algorithm to solve large-scale unrelated parallel machines scheduling problems. In (Xue, Zhong, Zhuang, & Xu, 2014), an ensemble evolution algorithm with self-adaptive learning population search techniques (EEA-SLPS) was proposed to solve high-dimensional optimization problems. In the EEA-SLPS, the population was divided into three subpopulations and three sub-algorithms were adopted to evolve the subpopulations, respectively. Liu and Zhou (2014) presented an improved PSO algorithm based on combining the simulated annealing (SA), co-evolution theory, quantum behavior theory, and diversity-guided mutation strategy to solve high-dimensional complex problems. In (Wang, Rahnamayan, & Wu, 2013), a parallel DE with self-adapting control parameters and generalized opposition-based learning was proposed to solve high-dimensional optimization problems. Kytöjoki et al. (2007) proposed an efficient variable neighborhood search heuristic algorithm to solve large-scale vehicle routing problems. In this approach, a strategy reminiscent of the guided local search metaheuristic was used to escape the local minima. In (Imanian, Shiri, & Moradi, 2014), a velocity-based artificial bee colony (VABC) algorithm was presented to solve high-dimensional optimization problems. The VABC applied a new search equation in the onlooker phase, which used the PSO search strategy to guide the search for candidate solutions. In (Mohapatra, Das, & Roy, 2017), a modified competitive swarm optimizer (MCSO) was presented to solve large-scale optimization problems. In the MCSO, two-thirds of the population swarms were being updated by a tri-competitive criterion unlike CSO. Zhang and Chiang (2017) proposed a novel consensus-based PSO-assisted trust-tech methodology to solve large-scale global optimization. The proposed algorithm integrated Trust-Tech methods, consensus-based PSO, and local optimization methods to compute a set of high-quality local optimal solutions. Kong et al. (2015) proposed a new binary harmony search algorithm for solving large-scale multidimensional knapsack problems. In this method, a simple but effective repair operator was embedded to guarantee the availability of harmonies. In (Ali, Awad, & Suganthan, 2015), a multi-population DE with a balanced ensemble of mutation strategies was proposed to solve large-scale global optimization problems. In the proposed algorithm, called *m*DE-*b*ES, the population was divided into independent subpopulations, each with different mutations and update strategies. Li and Yao (2012) presented a novel cooperative coevolving particle swarm optimization (CCPSO) algorithm to solve large-scale optimization problems. The proposed CCPSO adopted a new position update rule that relied on Cauchy and Gaussian distributions to sample new points in the search space. In (Rao, Savsani, & Vakharia, 2012), an efficient

teaching-learning-based optimization (TLBO) algorithm was proposed to solve large-scale nonlinear optimization problems. In (Zhao, Suganthan, & Das, 2011), a self-adaptive DE algorithm based on JADE mutation and a modified multi-trajectory search was presented to solve large-scale optimization problems. Singh and Agrawal (2016) proposed a self-organizing migrating algorithm with quadratic interpolation to solve large-scale global optimization problems.

3. Sine cosine algorithm

The SCA is a newly designed population-based optimization technique presented by Mirjalili (2016), which mimics the behaviors of sine and cosine mathematical functions. The solutions are updated based on the sine or cosine function, as in the following equation (Mirjalili, 2016):

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times | r_3 P_i^t - X_i^t |, & r_4 < 0.5 \\ X_i^t + r_1 \times \cos(r_2) \times | r_3 P_i^t - X_i^t |, & r_4 \geq 0.5 \end{cases} \tag{2}$$

where X_i is the position vector of the current solution, t is the current iteration, P_i is the destination solution, and $|\cdot|$ indicates the absolute value. The parameter r_1 is called the conversion parameter that is responsible for determining the area of the next solution; this area may be either the inside space between X_i and P_i or outside them. It is linearly decreased from a to 0 over the course of iterations to balance the exploration and exploitation as follows:

$$r_1(t) = a \times \left(1 - \frac{t}{t_{\max}}\right) \tag{3}$$

where t indicates the current iteration, $t_{\max}$ indicates the total number of iteration, and $a > 0$ is a constant.

Here, $r_2 \in [0, 2\pi]$ is a random variable that is used to obtain the direction of the movement of the next solution (i.e., if it is towards or away from P_i). Further, r_3 is a random variable that provides random weights for P_i to stochastically emphasize ($r_3 > 1$) or deemphasize ($r_3 < 1$) the effect of desalination in defining the distance. Term r_4 is used to switch between the sine and cosine functions in Eq.(3).

The pseudo code of conventional SCA is presented by **Algorithm 1**.

Algorithm 1 Sine Cosine Algorithm.

1: Initialize a set of search agents (solutions) (X)
2: **repeat**
3: Evaluate each of the search agents by the objective function
4: Update the best solution obtained so far ($P = X^*$)
5: Update the parameters r_1, r_2, r_3, and r_4
6: Update the position of search agents using Eq. (2)
7: **until** (t < maximum number of iterations)
8: Return the best solution obtained so far as the global optimum

4. Proposed algorithm

The traditional SCA suffers from some limitations, such as trapping in the local optima, slow convergence, unbalanced exploitation, and it is time-consuming (Elaziz, Oliva, & Xiong, 2017). In this section, we introduce the proposed ISCA that improves the performance of the traditional SCA. The no-free-lunch theorem (Wolpert & Macready, 1997) mentioned that an optimization algorithm could not be used to solve all the optimization problems. This is the primary motivation to develop the ISCA. The proposed algorithm does not affect the configuration of the SCA, and improves the SCA through two modifications. The two modifications are explained in detail in the following subsections.

4.1 Modified position-updating equation

In the traditional SCA, because all of the other search agents are attracted towards the global optimum (P), they may converge prematurely without sufficient exploration of the search space. This means the traditional SCA is prone to premature convergence. According to the position-updating equation of the SCA described by Eq. (2), the

new candidate solution is generated by moving the old solution towards the global-best solution (*P*). Therefore, the position-updating equation described by Eq. (2) is good for a local search. However, it is easily trapped in the local optima. To improve the performance of the SCA, one active research trend is to study its position-updating equation. Hitherto, various position-updating equations have been suggested (Meshkat & Parhizgar, 2017a; Meshkat & Parhizgar, 2017b; Sindhu et al., 2017).

Particle swarm optimization is a population-based algorithm developed by Kennedy an Eberhart (1995); it is inspired by the social behavior of bird flocking or fish schooling. Empirical studies on PSO with inertia weight (*w*) have shown that a relatively large *w* exhibits better global search ability, while a relatively small *w* results in a faster convergence (Shi & Eberhart, 1998). In the conventional PSO, the values of *w* are linearly decreased over the course of iterations.

Similar to the PSO algorithm, to improve the performance of the SCA (different from other previous works), we herein introduce an inertia weight coefficient (*w*) and modify the position-updating equation described by Eq. (2) as follows:

$$X_i^{t+1} = \begin{cases} w(t) \times X_i^t + r_1 \times \sin(r_2) \times | r_3 P_i^t - X_i^t |, & r_4 < 0.5 \\ w(t) \times X_i^t + r_1 \times \cos(r_2) \times | r_3 P_i^t - X_i^t |, & r_4 \geq 0.5 \end{cases} \tag{4}$$

where $w \in [0,1]$ is the inertia weight coefficient, and *t* is the current iteration. According to the Eq. (4), *w* is an important parameter. The primary reason is that the larger *w* values at the beginning facilitate the "global search" and smaller *w* values subsequently facilitate the "local refinement". Hence, the value of *w* is linearly decreased from a start value (w_{start}) to an end value (w_{end}) according to the following equation:

$$w(t) = w_{\text{end}} + (w_{\text{start}} - w_{\text{end}}) \times \frac{t_{\max} - t}{t} \tag{5}$$

Compared with the position-updating Eq. (2) in the conventional SCA, Eq. (4) can yield more information to the position-updating equation and generate a more promising candidate solution to improve the performance of the proposed SCA algorithm.

4.2 Modified conversion parameter strategy

All population-based optimization algorithms aim to achieve a balance in both the exploration and exploitation to obtain the promising regions of the search space and eventually converge to the global optimum. In the population-based optimization approaches, exploration refers to the ability to investigate various unknown regions in the search space to identify the global optimum. Exploitation refers to the ability to apply the knowledge of the previous satisfactory solution to obtain better solutions (Luo, Wang, & Xiao, 2013). Further, the abilities of exploration and exploitation in every population-based search algorithm are applied with specific operators.

According to (Mirjalili, 2016), four primary parameters exist in the conversional SCA: r_1, r_2, r_3, and r_4, where r_1 is the most critical parameter that converts from exploration to exploitation using an adaptive range in the sine and cosine functions. A larger conversion parameter r_1 facilitates global exploration, while a smaller conversion parameter (r_1) facilitates local exploitation. A suitable selection of conversion parameter r_1 can provide a balance between global exploration and local exploitation. However, in the conventional SCA, the value of r_1 decreases linearly from *a* to zero using the Eq. (2). In the beginning stage of the optimization process, the conversion parameter r_1 for the linearly decreasing strategy is good at exploration but poor at convergence; in the later stage of the search process, this strategy is good at exploitation but easily trapped in the local optima. Additionally, because the search process of the SCA is nonlinear and highly complicated, the linearly decreasing conversion parameter r_1 cannot truly reflect the actual search process.

Therefore, we herein modify the conversion parameter r_1 equation described by Eq. (3), and present a new conversion parameter r_1 for the nonlinearly decreasing strategy based on the Gaussian function as follows:

$$r_1(t) = (a_{\text{start}} - a_{\text{end}}) \times \exp\left(-\frac{t^2}{(k \times t_{\max})^2}\right) + a_{\text{end}} \tag{6}$$

where t indicates the current iteration, $t_{\max}$ indicates the total number of iterations, k is the nonlinear modulation index, and a_{start} and a_{end} are the initial and final values of constant a, respectively. Figure 1 shows the comparison curve of the linearly decreasing strategy and the Gaussian function decreasing strategy.

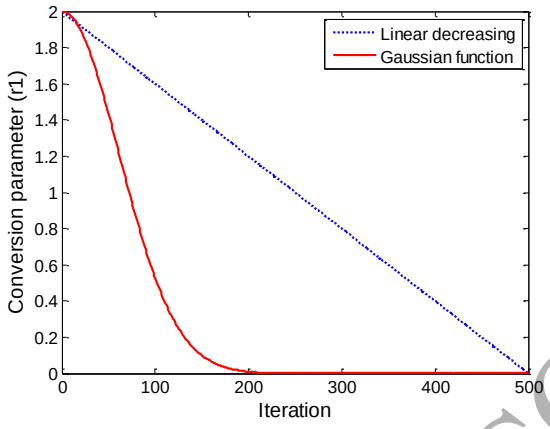


Fig. 1. The conversion parameter r_1 curve of linear decreasing and Gaussian function.

In summary, we combined the proposed modified position-updating equation and the modified conversion parameter strategy and developed the ISCA algorithm. The flowchart of the ISCA is shown in Fig. 2.

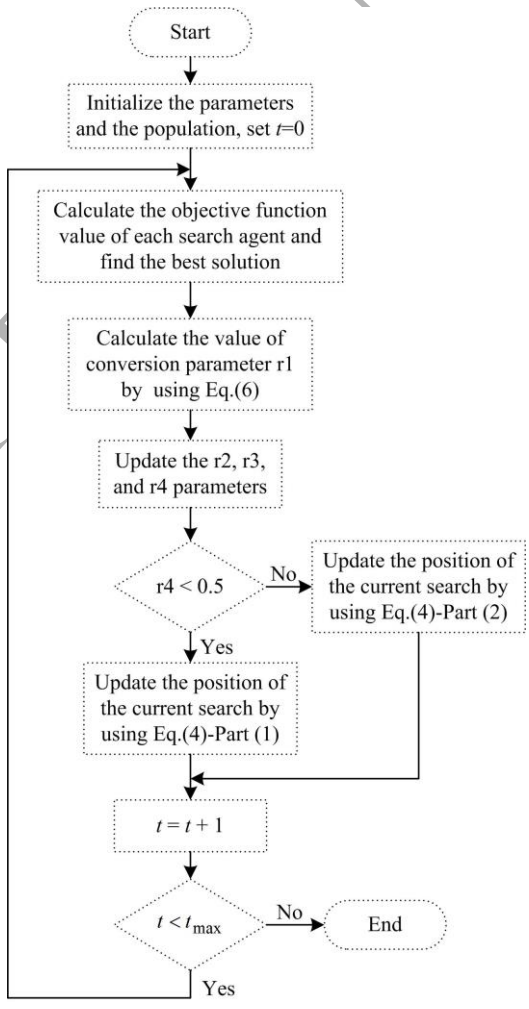


Fig. 2. The flowchart of the proposed ISCA.

4.3 Computational complexity

The time complexities of the SCA and ISCA are described as follows: 1) SCA and ISCA initialization require the $O(N)$ time, where N denotes the population size; 2) The fitness evaluation of each search agent requires $O(N)$ time; 3) The calculation of the control parameters of the SCA and ISCA require the $O(N)$ time; and 4) The search agents' position update steps of the SCA and ISCA require the $O(N)$ time. In summary, the total time complexity becomes the $O(N)$ time per generation. Therefore, the total time complexity of the SCA and ISCA for maximum number of iterations is $O(N \times t_{\max})$, where $t_{\max}$ is the maximum number of iterations.

5. Numerical experiment and analysis

To prove the theoretical descriptions indicated in the previous sections, a variety of experiments are conducted. First, a set of benchmark test functions with high dimensions (up to 5000) is employed to obtain if the ISCA is better than other population-based algorithms. Next, two test suites from IEEE CEC 2010 are used to further investigate the performance and efficiency of the ISCA. Finally, several challenging real-world engineering optimization problems are utilized to verify the search ability of the ISCA. The following subsections provide the experimental setup for each of the experimental phases, present the results and discuss them in detail.

5.1 Benchmark test functions and parameter settings

The 24 well-known benchmark test functions of dimensions D = 30, 100, 500, 1000, and 5000 collected from several references (Gao, Liu, & Huang, 2013; Li, Wang, Yan, & Li, 2015; Mirjalili, 2016) were applied to verify the performance and efficiency of the ISCA. The selected test functions were summarized in Table 1, which $f_{\min}$ is the global optimal value.

Table 1

Test functions used in the experiments of this paper.

Function name	Function formula	Search range	$f_{\min}$
Sphere	$f_1(x)=\sum_{i=1}^{n} x_i^2$	[−100,100]	0
Sum-Square	$f_2(x)=\sum_{i=1}^{n} ix_i^2$	[−10,10]	0
Schwefel's 2.22	$f_3(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[−10,10]	0
Rotated hyper-ellipsoid	$f_4(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	[−100,100]	0
Schwefel's 2.21	$f_5(x)=\max_i\{\lvert x_i\rvert,1\le x_i\le n\}$	[−100,100]	0
Rosenbrock	$f_6(x)=\sum_{i=1}^{n}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	[−30,30]	0
Step	$f_7(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	[−100,100]	0
Quartic	$f_8(x)=\sum_{i=1}^{n} ix_i^4$	[−1.28,1.28]	0
Noise	$f_9(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	[−1.28,1.28]	0
Sum-Power	$f_{10}(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{(i+1)}$	[−1,1]	0
Rastrigin	$f_{11}(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	[−5.12,5.12]	0
Ackley	$f_{12}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	[−32,32]	0
Griewank	$f_{13}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[−600,600]	0
Levy	$f_{14}(x)=\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\sin^2(3\pi x_1)+\lvert x_n-1\rvert[1+\sin^2(3\pi x_n)]$	[−10,10]	0

Alpine	$f_{15}(x)=\sum_{i=1}^{n}\lvert x_i\cdot\sin(x_i)+0.1\cdot x_i\rvert$	[−10,10]	0
Inverted Cosine Mixture	$f_{16}(x)=0.1n-\left(0.1\sum_{i=1}^{n}\cos(5\pi x_i)-\sum_{i=1}^{n}x_i^2\right)$	[−1,1]	0
Zakharov	$f_{17}(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$	[−5,10]	0
Pathological	$f_{18}(x)=\sum_{i=2}^{n}0.5+\frac{\sin^2(\sqrt{100x_{i-1}^2+x_i^2})-0.5}{1+0.001(x_{i-1}^2-2x_{i-1}x_i+x_i^2)^2}$	[−100,100]	0
Levy and Montalo	$f_{19}(x)=0.1(\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2(1+\sin^2(3\pi x_{i+1}))+(x_n-1)^2(1+\sin^2(2\pi x_n)))$	[−5,5]	0
Elliptic	$f_{20}(x)=\sum_{i=1}^{n}(10^6)^{(i-1)/(n-1)}x_i^2$	[−100,100]	0
Easom	$f_{21}(x)=(-1)^{n+1}\prod_{i=1}^{n}\cos(x_i)\cdot\exp[-\sum_{i=1}^{n}(x_i-\pi)^2]$	[−100,100]	0
Salomon	$f_{22}(x)=1-\cos\left(2\pi\sqrt{\sum_{i=1}^{n}x_i^2}\right)+0.1\sqrt{\sum_{i=1}^{n}x_i^2}$	[−100,100]	0
Schaffer	$f_{23}(x)=0.5+\frac{\sin^2\left(\sqrt{\sum_{i=1}^{n}x_i^2}\right)-0.5}{\left(1+0.001\left(\sum_{i=1}^{n}x_i^2\right)\right)^2}$	[−100,100]	0
Stretched V-sine	$f_{24}(x)=\sum_{i=1}^{n-1}(x_i^2+2x_{i+1}^2)^{0.25}\cdot((\sin 50(x_i^2+x_{i+1}^2)^{0.1})^2+1)$	[−10,10]	0

For a fair comparison, we set the same common control parameters for the SCA and ISCA. For each function, the population size (N) was fixed to 30, corresponding to the dimensions 30, 100, 500, 1000, and 5000, and the maximum number of iterations (t_{max}) was 500 on all the simulations. This meant the maximum number of fitness function evaluations (FFEs) was 15,000. The other specific parameters of the ISCA were set as follows: a_{start} and a_{end} were set to 0.1 and 0, respectively; the nonlinear modulation index k = 15; and w_{start} and w_{end} were set to 2 and 0, respectively. The ISCA and conventional SCA were coded in Matlab R2010b, and all the experiments were performed on a computer with Intel(R), Core(TM)2, Quad CPU Q8300 @ 2.50 GHz and 4.00 GB RAM in the Windows 8 environment.

5.2 Comparison with conventional SCA

First, it is necessary to compare the conventional SCA with the proposed ISCA in terms of convergence speed, accuracy, and robustness. We tested each function 30 times. Four criteria were used for two algorithms: best, mean, worst, and standard deviation. The comparison results of the SCA with ISCA are reported in Table 2 to solve the 24 benchmark functions with D = 30 that are previously defined in Table 1. In Table 2, "best" refers to the best value, "mean" means the average best values, "worst" denotes the worst value, and the "st.dev" represents the standard deviation value.

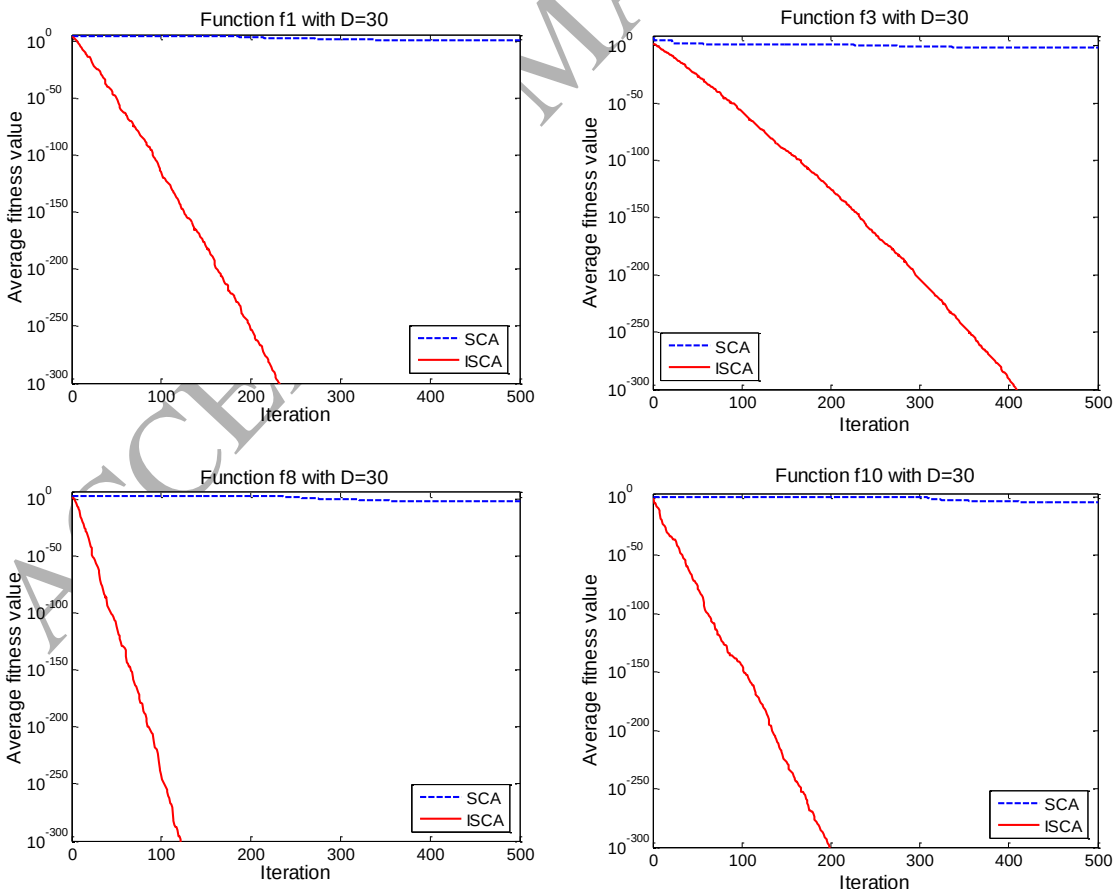
Table 2
Comparison results of SCA and ISCA for 24 benchmark test functions with D=30.

Function	SCA				ISCA			
	Best	Mean	Worst	St.dev	Best	Mean	Worst	St.dev
f_1	3.24E-01	9.11E+00	3.89E+01	1.63E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_2	4.09E-03	2.21E+00	8.72E+00	3.70E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_3	2.33E-04	1.72E-02	4.70E-02	2.13E-02	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_4	8.68E+02	5.06E+03	8.69E+03	2.94E+03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_5	9.07E+00	2.87E+01	4.31E+01	1.39E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_6	3.96E+02	1.94E+03	4.07E+03	1.68E+03	2.81E+01	2.88E+01	2.90E+01	3.79E-01
f_7	0.00E+00	1.00E+01	2.60E+01	1.11E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	1.66E-06	4.49E-03	1.51E-02	6.23E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00

f_9	2.57E-02	1.28E-01	3.57E-01	1.34E-01	7.95E-06	1.76E-04	3.13E-04	1.20E-04
f_{10}	8.05E-09	1.44E-05	8.49E-05	3.71E-05	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{11}	1.27E+01	4.06E+01	8.09E+01	3.29E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{12}	20.1657	20.2487	20.2908	4.91E-02	8.88E-16	8.88E-16	8.88E-16	0.00E+00
f_{13}	6.87E-01	1.17E+00	2.00E+00	4.95E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{14}	1.49E-02	4.12E+00	1.14E+01	5.36E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{15}	1.04E-02	2.69E+00	1.02E+01	4.51E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{16}	1.73E-04	1.29E-03	3.76E-03	1.49E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{17}	3.50E+00	4.32E+01	1.06E+02	3.92E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{18}	9.10E+00	9.41E+00	1.00E+01	4.57E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{19}	9.69E-06	7.12E-03	2.33E-02	9.52E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{20}	1.89E+00	6.94E+02	3.13E+03	1.36E+03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{21}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{22}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{23}	9.22E-02	1.88E-01	4.15E-01	1.30E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{24}	4.99E-01	1.02E+00	1.68E+00	4.43E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00

From Table 2, for 21 low-dimension (D = 30) benchmark test functions (i.e., f_1–f_5, f_7, f_8, f_{10}, f_{11}, f_{13}–f_{24}), the ISCA obtained the theoretical optima (0) with a fixed number of fitness function evaluations of 15,000. However, the conventional SCA only obtained the theoretical optima (0) for two functions (f_{21} and f_{22}). Moreover, compared with the conventional SCA, the ISCA obtained better results for 22 test functions (f_1–f_{20}, f_{23}, and f_{24}) and similar results for two test functions (f_{21} and f_{22}) with the same number of FFEs (15,000).

For further illustration, the convergence curves of the SCA and ISCA with D = 30 on eight representative test functions are shown in Fig. 3. From Fig. 3, we conclude that the proposed ISCA is effective and efficient in obtaining the global optimal solutions of optimization problems with a fast convergence rate. This shows the advantages of the modified position-updating equation, and that the modified conversion parameter strategy can yield a faster search.



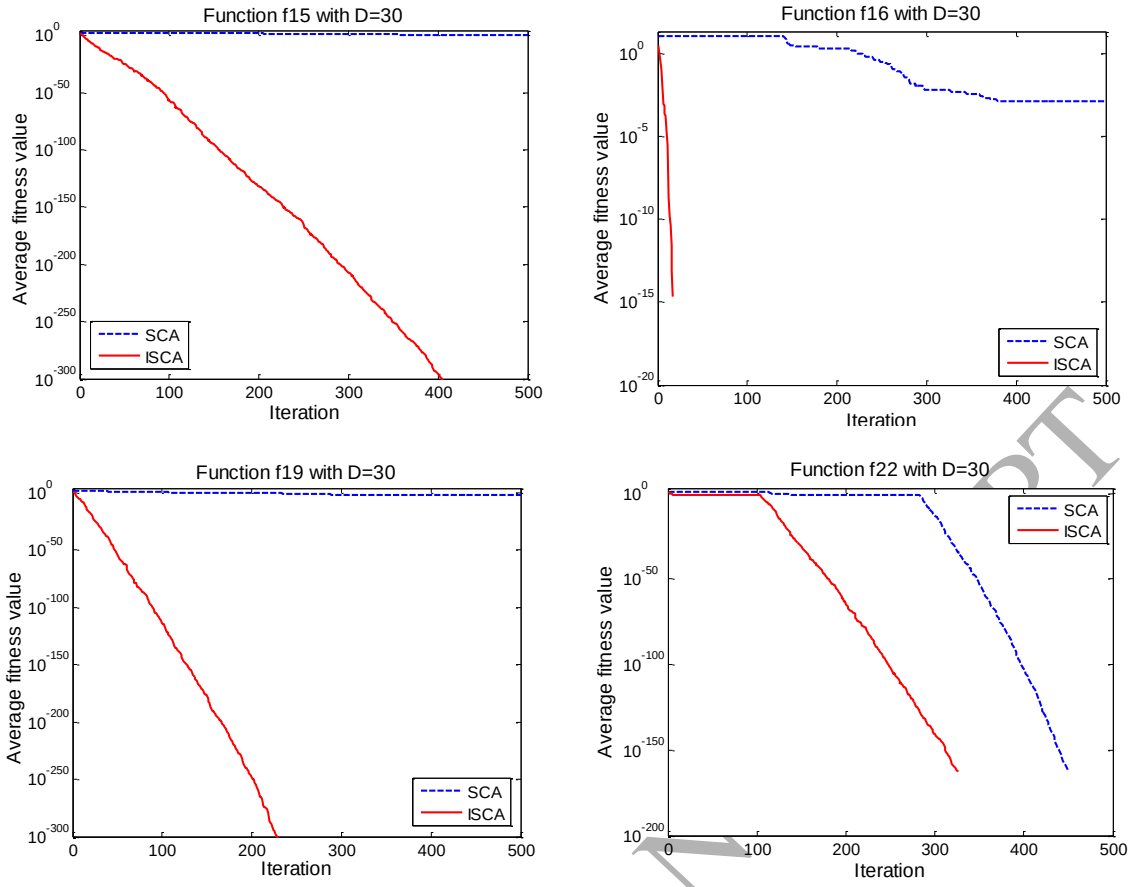


Fig. 3. Convergence curves of SCA and ISCA on eight representative test functions with D=30.

To further verify the scalability of the proposed method, the ISCA was tested with higher functions (D = 100, 500, 1000, and 5000). In this experiment, 24 benchmark test functions from Table 1 were used. Table 3 reports the mean results (mean) and the standard deviation results (st.dev) for 30 independent runs on the 24 benchmark test functions. We used the same parameter settings as in the experiments above, and maintain the population size or number of fitness function evaluations.

Table 3

Results obtained by ISCA over 30 independent runs on dimensions 100, 500, 1000, and 5000 with 15, 000 FFEs.

Function	Mean ± St.dev (D=100)	Mean ± St.dev (D=500)	Mean ± St.dev (D=1000)	Mean ± St.dev (D=5000)
f_1	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_2	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_3	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_4	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_5	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	2.45E−50 ± 3.54E−50
f_6	9.89E+01 ± 1.11E−01	4.99E+02 ± 2.14E−02	9.99E+02 ± 2.03E−02	5.00E+03 ± 5.91E−03
f_7	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_8	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_9	1.91E−04 ± 1.42E−04	2.13E−04 ± 1.65E−04	2.34E−04 ± 1.83E−04	2.58E−04 ± 1.97E−04
f_{10}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{11}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{12}	8.88E−16 ± 0.00E+00	8.88E−16 ± 0.00E+00	8.88E−16 ± 0.00E+00	8.88E−16 ± 0.00E+00
f_{13}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{14}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{15}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{16}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{17}	1.20E−273 ± 0.00E+00	3.80E−106 ± 8.60E−106	9.08E−27 ± 2.03E−26	3.29E+03 ± 3.52E+03
f_{18}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{19}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00

f_{20}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{21}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{22}	0.00E+00 ± 0.00E+00	3.06E−89 ± 6.84E−89	1.37E−45 ± 3.06E−45	2.00E−02 ± 4.476E−02
f_{23}	0.00E+00 ± 0.00E+00	9.72E−03 ± 0.00E+00	9.72E−03 ± 0.00E+00	9.72E−03 ± 0.00E+00
f_{24}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00

From Table 3, except for the functions f_5 (D = 5000), f_6, f_{17}, f_{22} (D = 500, 1000, and 5000) and f_{23} (D = 500, 1000, and 5000), we noticed that the ISCA showed excellent scalability to the search dimension for the other functions, meaning the performance did not deteriorate seriously as the dimension increased. It must be emphasized that the optimization of problems of 1000 and 5000 dimensions is highly challenging for the SCA because it does not use any particular operators tailored to solve high-dimensional optimization, such as divide-and-conquer operators (Cheng & Jin, 2014). Furthermore, the convergence curves of two typical separable functions (f_1 and f_5) and two typical nonseparable functions (f_{15} and f_{20}) with different dimensions are plotted in Fig. 4.

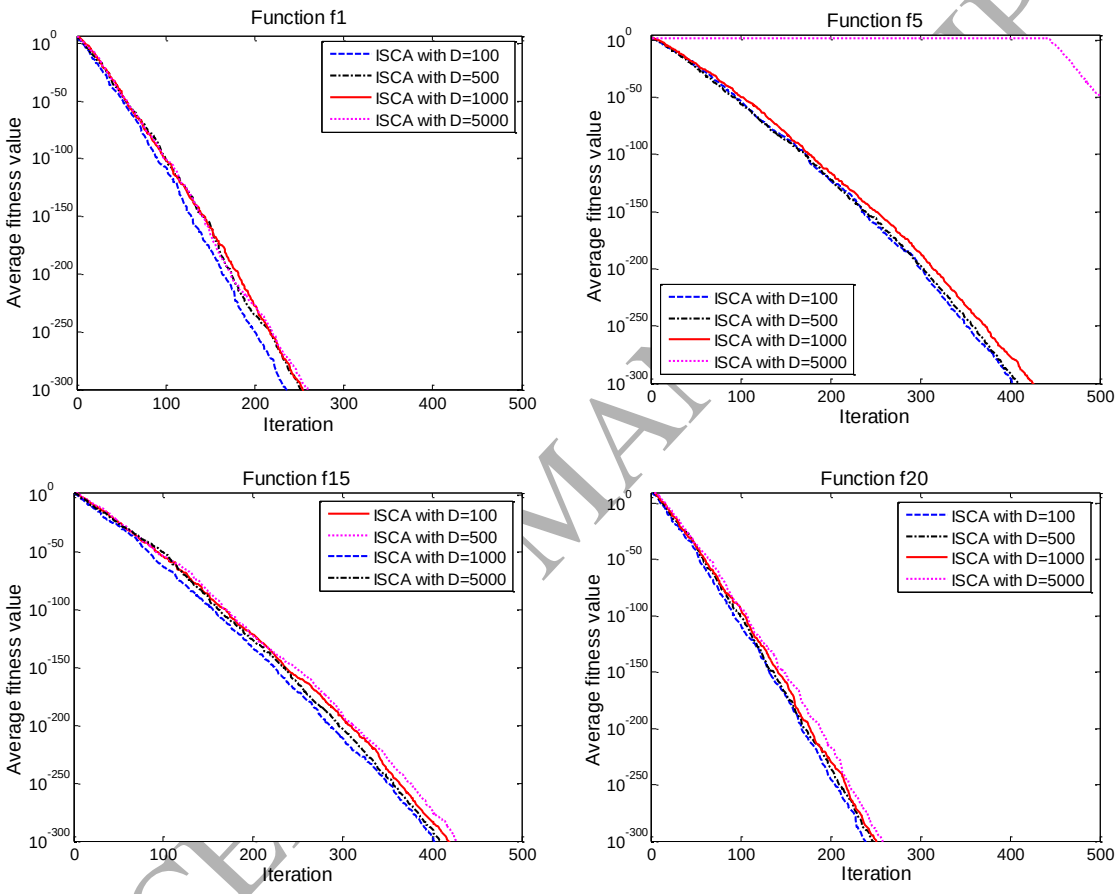


Fig. 4. Convergence curves of EEGWO with D=100, D=500, D=1000, and D=5000 on some typical functions.

For a better evaluation of the population-based optimization algorithms, not only the solution quality and the computational times should be investigated (Yurtkuran and Emel, 2015). Table 4 shows the mean CPU execution time results (in seconds) between the SCA on 30D, and the ISCA on 30D, 100D, 500D, 1000D, and 5000D benchmark test functions.

Table 4

The mean execution time (in seconds) obtained by ISCA and SCA on the 24 benchmark functions.

Function	SCA (D=30)	ISCA (D=30)	ISCA (D=100)	ISCA (D=500)	ISCA (D=1000)	ISCA (D=5000)
f_1	0.4987	0.4786	0.8082	2.8329	5.1952	25.2264
f_2	0.5051	0.5228	0.8776	2.8761	5.3209	25.8026
f_3	0.5070	0.5148	0.8373	2.6872	4.9045	23.4904
f_4	1.8384	1.8602	5.4699	30.0378	74.1501	335.2889
f_5	0.6395	0.6232	0.9307	2.8583	4.9336	23.6751

f_6	0.6423	0.6491	1.0056	2.9058	5.2998	24.8357
f_7	0.4798	0.4816	0.8128	2.8011	4.9304	24.9703
f_8	0.6184	0.6043	1.1598	4.3012	7.9124	30.8057
f_9	0.6181	0.6170	1.2027	4.3791	8.1840	32.0598
f_{10}	0.6363	0.6348	1.1160	4.0496	7.5552	30.0960
f_{11}	0.6219	0.6015	0.9408	2.8225	5.1038	25.2083
f_{12}	0.7084	0.6591	1.0139	2.9044	5.1406	25.3207
f_{13}	0.6515	0.5796	0.9726	3.0325	5.5271	26.5492
f_{14}	0.6961	0.7043	1.0628	3.3429	6.1181	28.0608
f_{15}	0.6272	0.5867	0.9484	2.8790	5.2957	25.9911
f_{16}	0.5737	0.5432	0.9214	2.7678	4.9506	25.3465
f_{17}	0.6063	0.6284	0.9686	2.7814	5.1227	23.7338
f_{18}	0.8319	0.7883	1.2058	3.3151	5.9178	27.9108
f_{19}	0.5970	0.6025	1.0110	3.2778	6.0837	27.1051
f_{20}	0.6382	0.6211	1.2517	4.6542	8.7002	32.0041
f_{21}	0.7493	0.7177	1.1469	3.5035	6.1580	27.1105
f_{22}	0.6557	0.7021	1.0067	2.8315	5.0567	23.5731
f_{23}	0.6617	0.6426	0.9567	2.7899	5.0708	23.8947
f_{24}	0.9408	0.9217	1.8242	6.8271	13.1321	44.3149

From Table 4, for the 24 benchmark test functions with D = 30, the mean CPU execution times of the ISCA and SCA were very close to each other. In other words, there was no significant difference between the ISCA and SCA on all of the functions in terms of the CPU execution times. For the other dimensions (D = 100, 500, 1000, and 5000) functions, the mean CPU execution times taken by the ISCA were small.

5.3 Comparison with other population-based algorithms

To further show the superiority of EEGWO for solving high-dimensional global optimization problems, we compared it with eight well-known population-based optimization algorithms: ABC (Karaboga & Basturk, 2007), PSO (Shi & Eberhart, 1998), DE (Storn & Price, 1997), salp swarm algorithm (SSA) (Mirjalili, Gandomi, Mirjalili, Saremi, Faris, & Mirjalili, 2017), teaching-learning-based optimization (TLBO) (Rao, Savsani, & Vakharia, 2012), whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016), evolution strategy with (CMA-ES) (Hansen, Müller, & Koumoutsakos, 2003), and exploration-enhanced grey wolf optimizer (EEGWO) (Long, Jiao, Liang, & Tang, 2018). The dimension of each test function was set to 1000 and each algorithm was independently executed 30 times on each function. In this experiment, the nine algorithms executed for a specified maximum number of iterations. The parameters of nine algorithms were set as follows: the population sizes of ABC, PSO, DE, SSA, TLBO, WOA, CMA-ES, EEGWO, and ISCA were 100, 100, 100, 30, 100, 30, 240, 30, and 30, respectively; the maximum number of iterations of ABC, PSO, DE, SSA, TLBO, WOA, CMA-ES, EEGWO, and ISCA were set to 1000, 1000, 1000, 1000, 500, 500, 500, 500, 500, respectively. Tables 5 and 6 show the average values (mean), and the standard deviation (st.dev) values obtained by nine algorithms for 24 benchmark test functions with D = 1000 from Table 1, the results of the Friedman's test, the results of the multiple-problem Wilcoxon's test, and the results of the Wilcoxon's rank sum test, respectively. For emphasis, the best results are marked in boldface in Table 5. It is noteworthy that "N/A" denotes the results are not available in Table 5.

Table 5
Results obtained by ISCA and other eight algorithms for 24 benchmark functions with 1000 dimensions.

Fun	Index	ABC	PSO	DE	SSA	TLBO	WOA	CMA-ES	EEGWO	ISCA
f_1	Mean	3.08E+06	3.52E+04	7.34E+05	1.92E+05	7.21E−71	1.44E−69	9.75E+03	**0.00E+00**	**0.00E+00**
	St.dev	4.60E+04	4.83E+03	2.49E+04	1.02E+04	4.12E−71	3.22E−69	6.64E+02	**0.00E+00**	**0.00E+00**
f_2	Mean	1.52E+07	1.43E+05	2.51E+06	8.56E+05	6.53E−70	2.04E−66	1.40E+05	**0.00E+00**	**0.00E+00**
	St.dev	2.08E+05	1.51E+04	6.44E+04	6.48E+05	3.44E−70	4.56E−66	7.42E+03	**0.00E+00**	**0.00E+00**
f_3	Mean	N/A	N/A	N/A	1.11E+03	N/A	3.16E−49	N/A	**0.00E+00**	**0.00E+00**
	St.dev	N/A	N/A	N/A	1.52E+01	N/A	6.79E−49	N/A	**0.00E+00**	**0.00E+00**
f_4	Mean	3.12E+06	3.50E+04	7.26E+05	4.36E+06	6.86E−71	1.25E+08	1.00E+04	**0.00E+00**	**0.00E+00**
	St.dev	4.51E+04	3.28E+03	2.22E+04	1.45E+06	3.45E−71	3.82E+07	9.58E+02	**0.00E+00**	**0.00E+00**
f_5	Mean	9.93E+01	3.52E+01	9.94E+01	4.27E+01	7.64E−28	8.37E+01	8.05E+01	**0.00E+00**	**0.00E+00**

	St.dev	2.07E−01	1.43E+00	9.30E−02	1.47E+00	2.79E−28	1.28E+01	1.44E+00	**0.00E+00**	**0.00E+00**
f_6	Mean	1.46E+10	7.00E+06	3.18E+09	7.85E+07	9.95E+02	**9.93E+02**	8.03E+05	9.99E+02	9.99E+02
	St.dev	1.64E+08	9.97E+05	1.78E+08	6.39E+07	6.37E−01	**7.93E−01**	1.31E+05	2.12E−02	2.03E−02
f_7	Mean	3.08E+06	6.14E+04	7.32E+05	2.21E+05	**0.00E+00**	**0.00E+00**	9.81E+03	**0.00E+00**	**0.00E+00**
	St.dev	3.70E+04	7.80E+03	8.81E+03	1.14E+04	**0.00E+00**	**0.00E+00**	1.81E+02	**0.00E+00**	**0.00E+00**
f_8	Mean	2.38E+05	7.54E+01	3.74E+04	1.08E+03	4.60E−142	8.44E−112	5.33E+01	**0.00E+00**	**0.00E+00**
	St.dev	3.80E+03	1.79E+01	1.45E+03	8.34E+01	3.60E−142	1.89E−111	1.02E+01	**0.00E+00**	**0.00E+00**
f_9	Mean	2.38E+05	1.28E+02	3.65E+04	1.23E+03	1.37E−03	2.06E−03	5.73E+01	**1.51E−04**	2.34E−04
	St.dev	2.60E+03	3.60E+01	6.21E+02	9.34E+01	2.82E−04	2.05E−03	4.09E+00	**1.47E−04**	1.83E−04
f_{10}	Mean	2.29E+00	6.73E−36	2.43E+00	5.69E−07	5.48E−192	6.09E−112	5.27E+00	**0.00E+00**	**0.00E+00**
	St.dev	3.55E−01	1.21E−35	4.39E−01	2.06E−06	0.00E+00	1.21E−111	1.74E+00	**0.00E+00**	**0.00E+00**
f_{11}	Mean	1.77E+04	3.57E+03	1.35E+04	6.32E+03	**0.00E+00**	3.64E−13	1.01E+04	**0.00E+00**	**0.00E+00**
	St.dev	1.13E+02	1.25E+02	8.59E+01	1.41E+02	**0.00E+00**	8.14E−14	1.30E+02	**0.00E+00**	**0.00E+00**
f_{12}	Mean	2.11E+01	9.74E+00	1.87E+01	1.43E+01	6.66E−15	5.15E−15	5.65E+00	3.73E−15	**8.88E−16**
	St.dev	6.61E−03	1.54E−01	5.55E−02	1.24E−01	0.00E+00	1.59E−15	8.38E−02	1.59E−15	**0.00E+00**
f_{13}	Mean	2.77E+04	2.89E+02	6.44E+03	1.67E+03	1.11E−16	**0.00E+00**	8.68E+01	**0.00E+00**	**0.00E+00**
	St.dev	6.32E+02	2.95E+01	1.33E+02	4.52E+01	0.00E+00	**0.00E+00**	5.46E+00	**0.00E+00**	**0.00E+00**
f_{14}	Mean	1.87E+05	3.54E+03	9.38E+04	9.34E+03	5.16E+02	4.69E−68	2.14E+04	**0.00E+00**	**0.00E+00**
	St.dev	4.61E+03	4.63E+02	1.12E+03	1.56E+03	3.24E+02	9.43E−68	1.44E+03	**0.00E+00**	**0.00E+00**
f_{15}	Mean	2.84E+03	2.41E+02	1.83E+03	6.36E+02	7.76E−37	9.46E−50	3.71E+02	5.90E−214	**0.00E+00**
	St.dev	1.83E+01	1.86E+01	2.63E+01	1.61E+01	1.86E−37	1.84E−49	1.81E+01	0.00E+00	**0.00E+00**
f_{16}	Mean	4.10E+02	3.97E+01	1.84E+02	9.08E+01	7.17E−75	**0.00E+00**	3.39E+01	**0.00E+00**	**0.00E+00**
	St.dev	1.33E+00	2.01E+01	2.29E+00	2.06E+00	5.81E−75	**0.00E+00**	1.64E+00	**0.00E+00**	**0.00E+00**
f_{17}	Mean	9.69E+06	9.67E+03	2.05E+04	1.98E+04	3.18E+03	1.56E+04	3.63E+04	**0.00E+00**	9.08E−27
	St.dev	4.76E+06	5.22E+03	5.51E+02	6.65E+02	2.15E+02	4.39E+02	8.55E+03	**0.00E+00**	2.03E−26
f_{18}	Mean	4.92E+02	4.32E+02	4.72E+02	4.30E+02	2.84E+02	1.08E+00	4.91E+02	**0.00E+00**	**0.00E+00**
	St.dev	6.39E−01	1.14E+01	4.73E−01	2.65E+00	1.13E+00	1.49E+00	7.58E−01	**0.00E+00**	**0.00E+00**
f_{19}	Mean	1.20E+03	2.31E+01	3.62E+02	1.03E+02	9.22E−74	1.32E−73	7.20E+01	**0.00E+00**	**0.00E+00**
	St.dev	1.75E+01	2.43E+00	1.42E+01	7.96E+00	4.26E−74	2.95E−73	7.18E+00	**0.00E+00**	**0.00E+00**
f_{20}	Mean	2.05E+11	2.84E+08	1.12E+09	4.10E+09	2.40E−66	4.17E−65	1.36E+10	**0.00E+00**	**0.00E+00**
	St.dev	5.94E+09	2.83E+07	4.76E+07	7.55E+08	1.42E−66	8.23E−65	4.09E+08	**0.00E+00**	**0.00E+00**
f_{21}	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{22}	Mean	1.77E+02	2.64E+01	9.91E+01	5.29E+01	3.00E−01	1.40E−01	1.42E+01	3.00E−01	**1.37E−45**
	St.dev	1.31E+00	5.81E−01	9.80E−01	7.44E−01	0.00E+00	8.94E−02	8.23E−01	1.64E−04	**3.06E−45**
f_{23}	Mean	5.00E−01	5.00E−01	5.00E−01	5.00E−01	7.82E−02	1.88E−02	5.00E−01	**7.78E−03**	9.72E−03
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.48E−07	1.73E−02	0.00E+00	**4.35E−03**	0.00E+00
f_{24}	Mean	3.92E+03	1.37E+03	3.10E+03	1.83E+03	1.24E−16	2.60E−30	1.19E+03	**0.00E+00**	**0.00E+00**
	St.dev	2.14E+01	2.54E+01	1.49E+03	3.61E+01	2.05E−17	5.21E−30	6.23E+00	**0.00E+00**	**0.00E+00**
Average ranking		8.125	5.2917	7.25	6.2083	3.125	3.375	5.625	1.25	1.2083
Total ranking		9	5	8	7	3	4	6	2	1

Table 6

Results of the multiple-proble Wilcoxon's test and Wilcoxon's rank sum test for ISCA and other eight selected algorithms on 24 test functions with D = 1000.

Algorithm	Better	Equal	Worst	R^+	R^-	p-value	α = 0.1	α = 0.05
ISCA versus ABC	23	1	0	288	12	1.5813E−08	Yes	Yes
ISCA versus PSO	23	1	0	288	12	4.7459E−08	Yes	Yes
ISCA versus DE	23	1	0	288	12	1.7928E−08	Yes	Yes
ISCA versus SSA	23	1	0	288	12	2.5913E−08	Yes	Yes
ISCA versus TLBO	20	3	1	248.5	51.5	3.3537E−04	Yes	Yes
ISCA versus WOA	19	4	1	238	62	7.4478E−04	Yes	Yes
ISCA versus CMA-ES	23	1	0	288	12	2.9273E−08	Yes	Yes
ISCA versus EEGWO	3	18	3	149.5	150.5	0.9567	No	No

From Table 5, an interesting result was that all the compared algorithms reliably obtained the minimum of f_{21}. Compared with ABC, PSO, DE, SSA, and CMA-ES, the ISCA obtained better results for all of the test functions with D = 1000, except for f_{21}. With respect to the TLBO, the ISCA obtained better results on 20 test functions, and

similar results on three test functions (f_7, f_{11}, and f_{21}). However, TLBO obtained a better result in function f_6. Compared with the WOA, the ISCA obtained better and similar results for nineteen functions and four functions (f_7, f_{13}, f_{16}, and f_{21}), respectively. However, a better result was obtained by the WOA for f_6. EEGWO is a novel GWO variant and its performance is highly competitive (Long, Jiao, Liang, & Tang, 2018). Compared to the EEGWO algorithm, the ISCA offered better and similar results on three functions (f_{12}, f_{15}, and f_{22}) and eighteen functions, respectively. However, for f_9, f_{17}, and f_{23}, better results were obtained by the EEGWO algorithm. With respect to the Friedman's test, the ISCA achieved the first rank followed by EEGWO.

Regarding the computational complexity (the number of fitness function evaluations), the WOA, EEGWO, and ISCA appeared to incur the minimum computational cost (15,000 FFEs) for all of the test functions, while the CMA-ES incurred a considerable computational cost (120,000 FFEs). The cost required by ABC, PSO, DE, SSA, and TLBO were moderated among in comparison the population-based optimization algorithm.

As shown in Table 6, regarding the multiple-problem Wilcoxon's test, the ISCA provides higher R^+ values than R^- values in all cases except for ISCA vs. EEGWO. Moreover, according to the Wilcoxon's rank sum test, the *p*-values are less than 0.1 and 0.05 in all the cases except for ISCA vs. EEGWO. This means the performance of the ISCA is significantly better than all of the algorithms (except EEGWO) for high-dimensional problems.

Table 7 provides the mean CPU execution time results (in seconds) between the ISCA and ABC, PSO, DE, SSA, TLBO, WOA, CMA-ES, and EEGWO on 1000*D* benchmark test functions. From Table 7, compared with the ABC, PSO, DE, and TLBO algorithms, the ISCA requires less CPU execution time for all of the test functions except for f_4. With respect to the SSA, WOA, CMA-ES and EEGWO algorithms, the ISCA requires less CPU execution time on all the functions.

Table 7

The mean execution time (in seconds) obtained by ISCA and other eight algorithms on the 24 functions with $D = 1000$.

Function	ABC	PSO	DE	SSA	TLBO	WOA	CMA-ES	EEGWO	ISCA
f_1	26.339	28.5978	49.4943	8.7123	11.5501	8.5068	2388.372	107.1346	5.1952
f_2	26.31	29.6092	51.875	9.2064	11.7721	8.4091	2319.839	106.2102	5.3209
f_3	27.1314	30.7476	49.82	8.9039	10.8328	8.6277	2364.298	104.4987	4.9045
f_4	28.3165	30.9824	56.7517	146.8803	12.2178	79.437	2347.795	174.9405	74.1501
f_5	28.0231	31.0569	56.8041	10.3273	13.1388	8.7879	2338.49	123.6104	4.9336
f_6	31.5167	32.7822	57.4354	10.7571	14.3411	9.1005	2527.814	103.9906	5.2998
f_7	29.9265	32.2708	58.7928	9.5937	11.8667	8.6795	2321.249	105.1827	4.9304
f_8	75.9452	54.8498	83.4745	16.3409	34.862	12.3708	2317.381	107.51	7.9124
f_9	74.3739	54.5177	83.4956	16.4254	34.4039	11.8858	2388.521	107.9603	8.184
f_{10}	74.7718	52.0548	79.1715	15.5782	29.8485	11.4902	2345.747	106.899	7.5552
f_{11}	47.706	37.5551	62.9479	12.1979	14.2331	9.1928	2430.214	104.6202	5.1038
f_{12}	60.9516	49.2548	74.554	12.8675	20.1444	9.3304	2458.195	104.994	5.1406
f_{13}	48.4572	40.155	66.984	13.2967	15.0733	9.2497	2437.834	105.9267	5.5271
f_{14}	48.8864	39.5067	67.9341	12.5109	17.2475	8.9998	2448.106	107.0471	6.1181
f_{15}	42.9589	35.5325	57.5255	11.2896	13.1888	8.7884	2375.522	125.6485	5.2957
f_{16}	44.1149	35.4687	59.4052	11.8603	13.906	8.9485	2428.575	105.2835	4.9506
f_{17}	29.9528	31.8035	51.7824	12.6404	12.6926	8.8107	2302.702	106.5417	5.1227
f_{18}	65.8393	48.7501	70.1403	15.0266	28.5936	10.8607	2373.08	105.5272	5.9178
f_{19}	48.994	41.8782	59.1858	12.8913	15.086	8.8158	2373.752	107.3758	6.0837
f_{20}	76.7828	56.0228	73.7229	16.4786	36.2047	11.8357	2377.44	109.7448	8.7002
f_{21}	51.1158	41.2878	61.1583	12.7073	22.5402	10.0601	2354.149	105.4573	6.158
f_{22}	31.9509	31.6798	52.3295	9.6309	13.9354	8.9477	2349.071	107.3255	5.0567
f_{23}	29.2466	31.3259	50.3137	9.6722	12.4835	8.8728	2360.729	104.4511	5.0708
f_{24}	151.4444	91.8571	48.2528	27.0738	67.8461	16.2435	2356.636	114.3736	13.1321

5.4 ISCA for IEEE CEC2010 large-scale global optimization problems

To further analyze and compare the performance of the proposed ISCA on high-dimensional problems, 20 benchmark test functions from the IEEE CEC2010 large-scale global optimization (LSGO) competition are used. A detailed description of these 20 LSGO problems is shown in Tang et al. (2010). The source codes for these functions

are available in http://www3.ntu.edu.sg/home/EPNSugan/, and the dimensions of these functions are 1000. For the analysis, four widely accepted population-based optimization techniques were selected as the algorithms for comparison. Multilevel cooperative co-evolution (MLCC) (Yang, et al. 2008) is a modified variant of the cooperative co-evolution algorithm. Cooperative PSO (CPSO-H) (Van den Bergh & Engelbrecht, 2004) is an improved version of PSO. Variance priority based on cooperative co-evolution differential evolution (DECC-VP) (Wang et al. 2009) is an improved version of DE. MA-SW-Chains (Molina et al., 2010) is a modified version of the memetic algorithm. These four algorithms were selected for comparison based on the following reasons: 1) MLCC, CPSO-H, DECC-VP, and MA-SW-Chains represent state-of-the-art LSGO techniques. 2) Their numerical performances are highly competitive. To ensure a fair comparison, the same maximum number of fitness function evaluations (FFEs) was set to 3.00E+06, which is the stopping criterion for five algorithms. Tables 8 and 9 summarize the average and standard deviation of the function values, results of the Friedman's test, and results of the multiple- problem Wilcoxon's test, respectively. It is noteworthy that the results of other algorithms in Table 8 were directly extracted from their original papers.

Table 8

Comparisons of five algorithms on 20 test functions with D = 1000 from IEEE CEC2010

Func	MLCC		CPSO-H		DECC-VP		MA-SW-Chains		ISCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev
fun1	8.53E−23	4.64E−23	1.50E+04	3.07E+04	7.61E−04	1.52E−04	2.10E−14	1.99E−14	1.55E+11	1.41E+09
fun2	4.97E−01	8.46E−01	4.15E+00	1.67E+00	6.60E−06	9.17E−07	8.10E+02	5.88E+01	1.64E+00	6.26E−01
fun3	2.01E−12	1.90E−13	1.27E−01	5.92E−02	3.82E−05	2.15E−06	7.28E−13	3.40E−13	3.10E−01	1.36E−01
fun4	9.79E+12	3.58E+12	1.55E+13	3.42E+12	5.44E+14	2.11E+14	3.53E+11	3.12E+10	1.37E+12	4.95E+11
fun5	3.46E+08	1.18E+08	6.30E+08	9.41E+07	5.47E+08	6.79E+08	1.68E+08	1.04E+08	3.27E+08	6.36E+07
fun6	1.37E+07	5.34E+06	1.98E+07	3.68E+04	1.96E+07	1.05E+05	8.14E+04	2.84E+05	1.29E+05	1.41E+04
fun7	5.81E+05	4.97E+05	2.55E+10	7.35E+09	1.37E+11	3.72E+11	1.03E+02	8.70E+01	3.71E+10	7.78E+09
fun8	3.84E+07	2.77E+07	1.21E+08	4.29E+07	2.60E+12	5.37E+12	1.41E+07	3.68E+07	1.59E+13	4.24E+11
fun9	1.84E+11	6.28E+04	1.72E+08	3.57E+07	2.67E+09	3.75E+08	1.41E+07	1.15E+06	1.66E+11	7.08E+08
fun10	3.14E+03	9.15E+02	7.00E+03	3.81E+02	6.91E+03	2.78E+02	2.07E+03	1.44E+02	2.96E+03	2.19E+02
fun11	8.56E+12	2.57E+12	1.98E+02	4.95E−01	2.01E+02	7.36E+00	3.80E+01	7.35E+00	1.89E+02	1.82E−01
fun12	1.50E+07	2.76E+01	3.18E+05	4.71E+04	1.28E+06	4.55E+04	3.62E−06	5.92E−07	1.33E+06	4.91E+03
fun13	9.41E+10	1.21E+04	2.29E+03	2.37E+03	1.27E+06	1.60E+05	1.25E+03	5.72E+02	6.30E+10	8.49E+08
fun14	3.17E+08	2.25E+07	4.15E+08	2.39E+07	4.03E+09	4.92E+08	3.11E+07	1.93E+06	2.29E+08	6.36E+06
fun15	6.78E+03	1.85E+03	1.39E+04	4.78E+02	1.38E+04	5.28E+02	2.74E+03	1.22E+02	1.64E+03	2.32E+01
fun16	3.86E+02	7.77E−02	3.97E+02	2.80E−01	4.11E+02	8.27E+00	9.98E+01	1.40E+01	3.18E+02	5.44E−01
fun17	8.68E+05	2.36E+02	6.83E+05	9.50E+04	2.48E+06	1.48E+05	1.24E+00	1.25E−01	3.27E+06	6.08E+05
fun18	1.45E+04	4.01E+03	5.57E+03	4.84E+03	1.41E+05	2.61E+04	1.30E+03	4.36E+02	1.41E+04	1.05E+03
fun19	7.74E+07	1.27E+03	5.19E+07	7.02E+07	6.84E+06	9.19E+05	2.85E+05	1.78E+04	5.90E+06	1.95E+05
fun20	2.27E+11	8.67E+03	3.47E+11	4.36E+08	1.22E+05	1.67E+04	1.07E+03	7.29E+01	1.59E+11	9.33E+08
Average ranking	3.30		3.60		3.75		1.30		3.05	
Total ranking	3		4		5		1		2	

From Table 8, it can be seen that ISCA outperformed MLCC, CPSO-H, DECC-VP, and MA-SW-Chains on 14, 11, 12, and two test functions, respectively. In contrast, MLCC, CPSO-H, DECC-VP, and MA-SW-Chains performed better than the ISCA on 6, 9, 8, and 18 test functions, respectively. In terms of the Friedman's test, MA-SW-Chains achieved the first rank followed by the ISCA. According to the multiple-problem Wilcoxon's test in Table 9, the R^+ values were larger than the R^- values in all cases except for the ISCA vs. MA-SW-Chains. In summary, the performance of MA-SW-Chains was the best.

Table 9

Results of the multiple-proble Wilcoxon's test and Wilcoxon's rank sum test for ISCA and other algorithms on 20 test functions

Algorithm	Better	Equal	Worst	R^+	R^-	p-value	α = 0.1	α = 0.05
ISCA vs MLCC	14	0	6	147	63	0.9031	No	No
ISCA vs CPSO-H	11	0	9	108	92	0.4570	No	No
ISCA vs DECC-VP	12	0	8	106	94	0.4094	No	No
ISCA vs MA-SW-Chains	2	0	18	10	200	0.0133	Yes	Yes

Increasing the dimensionality to millions of variables is a challenging issue in terms of computational performance and heuristic-driven searches on huge spaces (Cano, García-Martínez, & Ventura, 2017; Cano & García-Martínez, 2016). In fact, researchers typically discuss the effectiveness of heuristics on such extremely high-dimensional problems (Cano, García-Martínez, & Ventura, 2017; Lastra, Molina, & Benítez, 2015). We included random search (Cano, García-Martínez, & Ventura, 2017), GPU-MA-SW-Chains (Lastra, Molina, & Benítez, 2015), and MR-MA-SSW-Chains (Cano, García-Martínez, & Ventura, 2017) in our experiment to measure the effectiveness of the ISCA when the search space is that huge. Specifically, 9 benchmark test functions from IEEE CEC2010 were scaled up to 1,000,000 dimensions. To ensure a fair comparison, the same maximum number of FFEs was set to 5.00E+06. Table 10 summarizes the average and standard deviation of the function values, and the results of the Friedman's test, respectively. It is noteworthy that the results of other algorithms in Table 10 were directly extracted from their original papers.

Table 10

Results for 9 benchmark test functions with 1,000,000*D* from IEEE CEC2010

Function	Random search		GPU-MA-SW-Chains		MR-MA-SSW-Chains		ISCA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev
fun1	4.46E+14	3.29E+11	2.60E+14	1.20E+12	1.40E+14	8.40E+11	3.90E+14	4.84E+12
fun2	2.55E+07	7.10E+03	2.00E+07	3.60E+04	1.57E+07	3.24E+04	1.69E+04	2.72E+01
fun3	2.16E+01	2.97E−04	2.20E+01	2.80E−03	2.09E+01	1.04E−02	2.10E+01	1.32E−02
fun4	1.78E+15	1.42E+14	2.80E+14	9.90E+11	1.51E+14	3.61E+12	2.99E+15	7.27E+13
fun6	2.07E+07	5.19E+04	2.20E+01	6.40E−01	2.09E+01	5.11E−03	2.05E+07	5.72E+04
fun8	5.63E+16	7.55E+15	1.10E+08	7.80E+07	2.60E+09	6.21E+07	4.97E+16	4.15E+15
fun13	4.54E+15	3.81E+12	2.00E+15	4.10E+12	7.59E+14	2.56E+13	3.89E+15	8.93E+12
fun18	9.11E+15	4.24E+12	4.20E+15	7.50E+12	1.70E+15	3.94E+13	3.46E+15	7.39E+12
fun20	9.30E+15	1.07E+12	4.20E+15	1.20E+13	1.77E+15	2.32E+13	3.63E+15	9.98E+12
Average ranking	3.7778		2.4444		1.2222		2.5556	
Total ranking	4		2		1		3	

From Table10, it can be seen that the ISCA outperformed random search, GPU-MA-SW-Chains, and MR-MA-SSW-Chains on eight, four, and one test function, respectively. In contrast, random search, GPU-MA-SW-Chains, and MR-MA-SSW-Chains perform better than the ISCA on one, five, and eight test functions, respectively. In terms of the Friedman's test, MR-MA-SW-Chains achieved the first rank followed by GPU-MA-SW-Chains.

5.5 The effectiveness of the two components in ISCA

As mentioned previously, the ISCA contains two primary components: the modified conversion parameter strategy and the modified position-updating equation. The purpose of this subsection is to verify the effectiveness of these two components. Hence, two additional experiments were performed on 24 traditional benchmark test functions (D = 30), as shown in Table 1. In the first experiment, the ISCA only adopts the modified position-updating equation (Eq. (4)) and the modified conversion parameter equation was not used (ISCA−1). In this case, similar to Mirjalili (2016), the conversion parameter equation (Eq (3)) was used. Additionally, in the second experiment, the ISCA only adopts the modified conversion parameter strategy (Eq. (6)) and the modified position-updating equation is ignored (ISCA−2). In this case, similar to Mirjalili (2016), the position-updating Eq. (2) was used. For each test function, 30 independent runs were implemented and the maximum FEs was set to 15,000. That is, the population size was set to 30 and the maximum number of iteration was set 500. The "mean" and "st.dev" results of ISCA−1, ISCA−2, and ISCA are shown in Table 11. Wilcoxon's rank sum test at a 0.05 significance level was performed between the ISCA and each of ISCA−1 and ISCA−2. Terms "−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, or similar to that of the ISCA, respectively.

Table 11

Experimental results of ISCA, ISCA−1, and ISCA−2 on 23 functions (D=30) with 15,000 FEs.

Function	ISCA−1 (Mean ± St.dev)	ISCA−2 (Mean ± St.dev)	ISCA (Mean ± St.dev)
f_1	0.00E+00 ± 0.00E+00 ≈	6.22E−14 ± 1.39E−13 −	0.00E+00 ± 0.00E+00
f_2	0.00E+00 ± 0.00E+00 ≈	7.98E−12 ± 1.72E−11 −	0.00E+00 ± 0.00E+00
f_3	1.74E-197 ± 0.00E+00 −	7.26E−14 ± 1.45E−13 −	0.00E+00 ± 0.00E+00
f_4	5.21E-304 ± 0.00E+00 −	4.13E+03 ± 2.85E+03 −	0.00E+00 ± 0.00E+00
f_5	1.81E-173 ± 0.00E+00 −	1.83E+01 ± 1.39E+01 −	0.00E+00 ± 0.00E+00
f_6	2.86E+01 ± 4.34E−01 +	2.81E+01 ± 4.66E−01 +	2.88E+001 ± 3.79E−01
f_7	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00
f_8	0.00E+00 ± 0.00E+00 ≈	6.22E−12 ± 1.31E−11 −	0.00E+00 ± 0.00E+00
f_9	4.80E−05 ± 5.38E−05 +	2.64E−02 ± 1.52E−02 −	1.76E−04 ± 1.20E−04
f_{10}	0.00E+00 ± 0.00E+00 ≈	9.38E−06 ± 2.10E−05 −	0.00E+00 ± 0.00E+00
f_{11}	0.00E+00 ± 0.00E+00 ≈	2.79E+01 ± 5.16E+01 −	0.00E+00 ± 0.00E+00
f_{12}	8.88E−16 ± 0.00E+00 ≈	9.43E−07 ± 2.06E−06 −	8.88E−16 ± 0.00E+00
f_{13}	0.00E+00 ± 0.00E+00 ≈	3.66E−01 ± 3.55E−01 −	0.00E+00 ± 0.00E+00
f_{14}	0.00E+00 ± 0.00E+00 ≈	9.30E−09 ± 2.08E−08 −	0.00E+00 ± 0.00E+00
f_{15}	6.39E-190 ± 0.00E+00 −	1.92E−04 ± 2.66E−04 −	0.00E+00 ± 0.00E+00
f_{16}	0.00E+00 ± 0.00E+00 ≈	3.63E−15 ± 7.87E−15 −	0.00E+00 ± 0.00E+00
f_{17}	8.46E-222 ± 0.00E+00 −	1.69E+01 ± 1.56E+01 −	0.00E+00 ± 0.00E+00
f_{18}	2.49E-01 ± 5.57E-01 −	9.74E+00 ± 4.15E-01 −	0.00E+00 ± 0.00E+00
f_{19}	0.00E+00 ± 0.00E+00 ≈	1.03E−14 ± 1.65E−14 −	0.00E+00 ± 0.00E+00
f_{20}	0.00E+00 ± 0.00E+00 ≈	6.82E−09 ± 1.51E−08 −	0.00E+00 ± 0.00E+00
f_{21}	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00
f_{22}	9.08E−80 ± 2.03E−79 −	1.80E−01 ± 4.48E−02 −	0.00E+00 ± 0.00E+00
f_{23}	0.00E+00 ± 0.00E+00 ≈	4.09E−02 ± 2.45E−02 −	0.00E+00 ± 0.00E+00
f_{24}	0.00E+00 ± 0.00E+00 ≈	2.20E−05 ± 3.33E−05 −	0.00E+00 ± 0.00E+00
Total −/+/≈	7/2/15	21/1/2	/

From Table 11, compared with ISCA, ISCA-1 showed a similar performance on 15 test functions and demonstrated a better performance on only two test functions. However, the ISCA surpassed ISCA-1 on seven functions. We attribute the above phenomenon to the fact the modified position-updating equation is more effective for balancing between exploration and exploitation in the evolution process.

In addition, the ISCA surpassed ISCA-2 on 21 test functions and yielded similar results on two test functions. This is easy to comprehend because the modified conversion parameter equation is not used dependently in the search process. Consequently, it can adapt to the search according to different landscapes. Thus, the performance differences between ISCA and ISCA-2 are significant.

From Table 11, we conclude that the two components above can benefit each other to enhance the performance of the SCA.

5.6 Sensitivity analysis of the inertia weight (*w*) *value*

In the ISCA, the inertia weight (*w*) is an important parameter. As shown in Eq. (4), *w* is critical for generating a more promising candidate solution. To investigate the sensitivity of the parameter above, we tested the ISCA with different *w*: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. Further, 24 benchmark functions from Table 1 were selected to test the performance of the ISCA with different values of *w*. The dimension was set to 100 for all the test functions and the maximum number of FFEs was set to 15, 000. That is, the population size was set to 30 and the maximum number of iterations was set to 500. Thirty independent runs were implemented for each test function and the experimental results of the ISCA with different *w* values are shown in Tables 12-14.

Table 12

Experimental results of ISCA based on the increase of w varying step length 0.1 from 0.1 to 1 from f_1 to f_8.

w value	Index	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0.1	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.89E+01	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8.71E−02	0.00E+00	0.00E+00
0.2	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.89E+01	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E−02	0.00E+00	0.00E+00
0.3	Mean	0.00E+00	0.00E+00	1.23E−263	0.00E+00	8.92E−234	9.89E+01	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8.25E−02	0.00E+00	0.00E+00
0.4	Mean	0.00E+00	0.00E+00	4.48E−210	0.00E+00	1.96E−135	9.89E+01	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.40E−135	1.26E−02	0.00E+00	0.00E+00
0.5	Mean	0.00E+00	0.00E+00	5.66E−167	2.96E−207	3.36E−70	9.89E+01	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.51E−70	9.47E−02	0.00E+00	0.00E+00
0.6	Mean	3.02E−219	7.12E−209	2.38E−117	2.16E−135	6.14E−49	9.89E+01	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	5.30E−117	4.80E−135	1.36E−48	1.17E−01	0.00E+00	0.00E+00
0.7	Mean	7.74E−146	2.74E−142	7.50E−81	1.19E−95	9.34E−32	9.89E+01	0.00E+00	9.20E−262
	St.dev	1.70E−145	6.10E−142	1.55E−80	2.56E−95	1.94E−31	1.13E−01	0.00E+00	0.00E+00
0.8	Mean	1.90E−86	9.03E−86	1.45E−51	1.45E−45	3.33E−17	9.87E+01	0.00E+00	6.80E−147
	St.dev	4.06E−86	2.01E−85	3.11E−51	3.24E−45	5.08E−17	3.61E−01	0.00E+00	1.50E−146
0.9	Mean	1.40E−34	3.87E−35	3.57E−24	2.04E−08	1.57E−05	9.89E+01	0.00E+00	5.99E−60
	St.dev	2.55E−34	5.54E−35	5.05E−24	4.56E−08	1.03E−05	1.11E−02	0.00E+00	1.30E−59
1.0	Mean	1.23E+04	2.76E+03	5.81E+00	2.16E+05	9.63E+01	2.53E+07	9.75E+03	4.65E+01
	St.dev	8.63E+03	1.20E+03	3.77E+00	3.28E+04	1.14E+00	1.75E+07	8.60E+03	1.96E+01

Table 13

Experimental results of ISCA based on the increase of w varying step length 0.1 from 0.1 to 1 from f_9 to f_{16}.

w value	Index	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0.1	Mean	6.77E−05	0.00E+00	0.00E+00	8.88E−16	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	St.dev	7.33E−05	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
0.2	Mean	1.08E−04	0.00E+00	0.00E+00	8.88E−16	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	St.dev	1.18E−04	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
0.3	Mean	1.28E−04	1.29E−156	0.00E+00	8.88E−16	0.00E+00	0.00E+00	5.12E−263	0.00E+00
	St.dev	1.34E−04	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
0.4	Mean	1.40E−04	6.86E−120	0.00E+00	8.88E−16	0.00E+00	0.00E+00	4.22E−213	0.00E+00
	St.dev	1.13E−04	1.50E−119	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
0.5	Mean	1.50E−04	6.32E−72	0.00E+00	8.88E−16	0.00E+00	0.00E+00	1.47E−165	0.00E+00
	St.dev	8.52E−05	1.41E−71	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
0.6	Mean	1.70E−04	3.28E−45	0.00E+00	8.88E−16	0.00E+00	3.30E−214	1.20E−122	0.00E+00
	St.dev	1.01E−04	7.28E−45	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.00E−122	0.00E+00
0.7	Mean	2.17E−04	1.03E−17	0.00E+00	8.88E−16	0.00E+00	2.04E−142	1.38E−81	0.00E+00
	St.dev	2.68E−04	2.30E−17	0.00E+00	0.00E+00	0.00E+00	2.90E−142	3.05E−81	0.00E+00
0.8	Mean	2.75E−04	1.25E+00	0.00E+00	2.31E−15	0.00E+00	3.38E−82	3.26E−51	0.00E+00
	St.dev	3.23E−04	1.77E+00	0.00E+00	1.95E−15	0.00E+00	7.12E−82	6.58E−51	0.00E+00
0.9	Mean	3.50E−04	2.82E+00	0.00E+00	3.73E−15	0.00E+00	4.71E−32	8.48E−22	0.00E+00
	St.dev	3.17E−04	7.50E−01	0.00E+00	1.59E−15	0.00E+00	1.05E−31	1.67E−21	0.00E+00
1.0	Mean	6.62E+01	3.26E+00	2.96E+02	7.83E+00	3.27E+01	1.59E+03	1.56E+01	2.37E+00
	St.dev	6.89E+01	3.22E−01	1.01E+02	3.50E+00	2.89E+01	5.50E+02	1.18E+01	9.67E−01

Table 14

Experimental results of ISCA based on the increase of w varying step length 0.1 from 0.1 to 1 from f_{17} to f_{24}.

w value	Index	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}
0.1	Mean	5.88E−227	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.04E−117	5.83E−03	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.30E−117	5.32E−03	0.00E+00
0.2	Mean	7.04E−186	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.27E−87	5.83E−03	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.83E−87	5.32E−03	0.00E+00
0.3	Mean	2.84E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.66E−07	1.94E−03	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.95E−07	4.35E−03	0.00E+00
0.4	Mean	1.48E−135	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.00E−02	5.83E−03	0.00E+00
	St.dev	4.40E−135	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.47E−02	5.32E−03	0.00E+00

0.5	Mean	2.75E−117	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.00E−02	7.77E−03	0.00E+00
	St.dev	6.10E−117	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.47E−02	4.35E−03	0.00E+00
0.6	Mean	2.32E−85	0.00E+00	1.53E−217	3.16E−194	0.00E+00	3.99E−02	5.83E−03	9.51E−61
	St.dev	4.18E−85	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.47E−02	5.32E−03	1.94E−60
0.7	Mean	3.10E−54	0.00E+00	4.96E−140	3.44E−141	0.00E+00	7.99E−02	9.72E−03	3.30E−44
	St.dev	6.93E−54	0.00E+00	1.10E−139	7.60E−141	0.00E+00	4.47E−02	1.79E−07	3.52E−44
0.8	Mean	6.29E−32	0.00E+00	3.18E−90	1.53E−80	0.00E+00	7.99E−02	9.72E−03	2.34E−28
	St.dev	6.45E−32	0.00E+00	3.94E−90	3.43E−80	0.00E+00	4.47E−02	4.47E−08	2.55E−28
0.9	Mean	6.25E−09	1.06E+01	1.57E−36	1.65E−30	0.00E+00	7.99E−02	9.72E−03	1.03E−13
	St.dev	1.06E−08	1.90E+01	3.48E−36	2.40E−30	0.00E+00	4.46E−02	8.37E−08	7.04E−14
1.0	Mean	4.60E+02	4.45E+01	5.33E+00	9.03E+05	0.00E+00	1.14E+01	4.99E−01	2.25E+01
	St.dev	5.36E+01	4.39E−01	3.70E+00	3.82E+05	0.00E+00	1.93E+00	7.26E−04	1.05E+01

From Tables 12-14, an interesting result was that the ISCA most reliably obtained the minimum of f_{21} with all the w values. Compared with the ISCA with adaptive w values, the ISCA with w = 0.1 and w = 0.2 can find similar results on all the test functions except for f_{17}, f_{22}, and f_{23}. With respect to the ISCA with adaptive w values, the ISCA with w = 0.3 and w = 0.4 yielded similar results on 17 test functions and worse results on 7 test functions. Compared to the ISCA with w = 0.6, the ISCA with adaptive w values provided better results on 8 test functions and similar results on 16 test functions. Compared with the ISCA with w = 0.6 and w = 0.7, the ISCA with adaptive w values yielded better results on 14 test functions and similar results on 10 test functions. With respect to the ISCA with w = 0.8, the ISCA with adaptive w values can get better results on 18 test functions and similar results on 6 functions. Compared to the ISCA with w = 0.9, the ISCA with adaptive w values yielded better results on 19 functions and similar results on 5 functions. Compared with the ISCA with w = 1.0, the ISCA with adaptive w values obtained better results on all the test functions except for f_{21}.

5.7 ISCA for engineering design problems

This subsection further verifies the performance and efficiency of the ISCA by solving three constrained real engineering design problems: pressure vessel design, tension/compression spring design, and welded beam design. These problems which were widely discussed in the literature and have been solved to better clarify the effectiveness of the algorithms. We applied the penalty function method to deal with the constraints. In the ISCA, the population size was 50 and the maximum number of iterations was 1000.

5.7.1 Pressure vessel design problem

The aim of the pressure vessel design problem is to minimize the whole cost of the cylindrical pressure vessel. 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. 5.

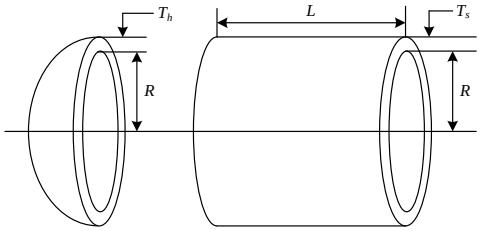


Fig. 5. Schematic view of pressure vessel design problem

The mathematical formulation of pressure vessel design problem is defined as follows:

Consider $\mathbf{X} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L]$,

Minimize $f(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$,

Subject to $g_1(\mathbf{X}) = -x_1 + 0.00193x_3 \le 0$,

$g_2(\mathbf{X}) = -x_3 + 0.00954x_3 \le 0$,

$$g_3(\mathbf{X}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 129600 \le 0,$$

$$g_4(\mathbf{X}) = x_4 - 240 \le 0,$$

where $0 \le x_1, x_2 \le 99$, $10 \le x_3, x_4 \le 200$.

The ISCA was applied to solve the pressure vessel design problem and compared it with twenty optimization algorithms which were reported in previews works as shown in Table 15. From Table 15, with respect to MVO, CPSO, GSA, GA, CGDA, GWO, EEGWO, and SCA, the ISCA provided a better result for the pressure vessel design problem.

Table 15
The experimental results of ISCA and other algorithms for pressure vessel design problem.

Algorithm	Optimal values for variables				Optimal cost
	T_s	T_h	R	L	
WOA (Mirjalili & Lewis, 2016)	0.8125	0.4375	42.0982699	176.638998	6059.7410
MVO (Mirjalili et al., 2016)	0.8125	0.4375	42.090738	176.73869	6060.8066
CPSO (He & Wang, 2007)	0.8125	0.4375	42.091266	176.7465	6061.0777
GSA (Rashedi et al., 2009)	1.125	0.625	55.9886598	84.4542025	8538.8359
GA (Coello & Mezura-Montes, 2002)	0.8125	0.4375	42.097398	176.65405	6059.9463
ES (Mezura-Montes & Coello, 2008)	0.8125	0.4375	42.098087	176.640518	6059.7456
ACO (Kaveh & Talatahari, 2010)	0.8125	0.4375	42.098353	176.637751	6059.7258
CGDA (Baykasoğlu, 2012)	0.8125	0.4375	42.0975	176.6484	6059.8391
CSA (Gandomi et al., 2013)	0.8125	0.4375	42.0984	176.6366	6059.7140
BA (Gandomi et al., 2013)	0.8125	0.4375	42.0984	176.6366	6059.7143
MFO (Mirjalili, 2015)	0.8125	0.4375	42.098445	176.636596	6059.7143
HPSODE (Liu et al., 2010)	0.8125	0.4375	42.098446	176.636596	6059.714335
CDE (Hang et al., 2007)	0.8125	0.4375	42.0984	176.6376	6059.7340
UABC (Brajevic & Tuba, 2013)	0.8125	0.4375	42.098446	176.636596	6059.714335
AFA (Baykasoğlu & Ozsoydan, 2015)	0.8125	0.4375	42.0984	176.6366	6059.7143
CSA (Askarzadeh, 2016)	0.8125	0.4375	42.0984	176.6366	6059.7144
TEO (Kaveh & Dadras, 2017)	0.8125	0.4325	42.0984	173.6366	6059.71
GWO (Mirjalili et al., 2014)	0.8125	0.4345	42.089181	176.758731	6051.5639
EEGWO (Long et al., 2018)	13.09291	6.792196	42.09758	176.6495	6059.8704
SCA (Mirjalili, 2016)	0.817577	0.417932	41.74939	183.5727	6137.3724
ISCA (this work)	12.96419	7.150134	42.09829	176.6392	6059.7489

5.7.2 Tension/compression spring design problem

Fig. 6 illustrates the tension/compression spring design problem. 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), number of active coils (P), and wire diameter (d).

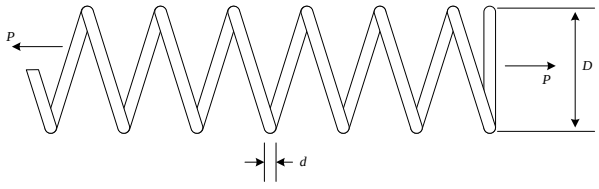


Fig. 6. Schematic view of tension/compression spring design problem

The mathematical definition of tension/compression spring design problem as follows:

Consider $\mathbf{X} = [x_1\ x_2\ x_3] = [d\ D\ P]$,

Minimize $f(\mathbf{X}) = (x_3 + 2)x_2 x_1^2$,

Subject to $g_1(\mathbf{X}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0$,

$$g_2(\mathbf{X}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} \le 0,$$

$$g_3(\mathbf{X}) = 1 - \frac{140.45x_1}{x_2^2 x_3} \le 0,$$

$$g_4(\mathbf{X}) = \frac{x_1 + x_2}{1.5} - 1 \le 0,$$

where $0.05 \le x_1 \le 2$, $0.25 \le x_2 \le 1.30$, $2.00 \le x_3 \le 15$.

The optimization results obtained by the proposed ISCA for this problem were evaluated by comparing it with 19 other optimization algorithms that were reported previously, as shown in Table 16. From Table 16, compared with the WOA, MVO, CPSO, GSA, GA, ES, HS, CDE, and CSA, the ISCA yielded better results for the tension/compression spring design problem.

Table 16
The experimental results of ISCA and other algorithms for tension/compression spring design problem.

Algorithm	Optimal values for variables			Optimal cost
	d	*D*	*P*	
WOA (Mirjalili & Lewis, 2016)	0.051207	0.345215	12.54854	0.0126763
MVO (Mirjalili et al., 2016)	0.05251	0.37602	10.33513	0.012790
CPSO (He & Wang, 2007)	0.051728	0.357644	11.244543	0.0126747
GSA (Rashedi et al., 2009)	0.05000	0.317312	14.22867	0.0128739
GA (Coello & Montes, 2002)	0.051989	0.363965	10.890522	0.012681
ES (Mezura-Montes & Coello, 2008)	0.051643	0.35536	11.397926	0.012698
BA (Gandomi et al., 2013)	0.05169	0.35673	11.2885	0.0126652
HS (Mahdavi et al., 2007)	0.051154	0.349871	12.076432	0.0126706
MFO (Mirjalili, 2015)	0.051994457	0.36410932	10.868421862	0.0126669
CGDA (Baykasoğlu, 2012)	0.0516925	0.3568108	11.2835059	0.012665
AFA (Baykasoğlu & Ozsoydan, 2015)	0.051667	0.356198	11.319561	0.0126653
HPSODE (Liu et al., 2010)	0.0516888101	0.3567117001	11.289319935	0.0126652329
CDE (Hang et al., 2007)	0.051609	0.354714	11.410831	0.0126702
CSA (Askarzadeh, 2016)	0.051689	0.356717	11.289012	0.0126652
UABC (Brajevic & Tuba, 2013)	0.051691	0.356769	11.285988	0.012665
TEO (Kaveh & Dadras, 2017)	0.051775	0.358792	11.168390	0.012665
GWO (Mirjalili et al., 2014)	0.05169	0.356737	11.28885	0.012666
EEGWO (Long et al., 2018)	0.051673	0.35634	11.3113	0.012665
SCA (Mirjalili, 2016)	0.05078	0.334779	12.72269	0.0127097
ISCA (this work)	0.0520217	0.364768	10.8323	0.012667

5.7.3 Welded beam design problem

The aim of the welded beam design problem is to minimize the fabrication costs by determining the optimal value of four variables: the thickness of weld (*h*), thickness of the bar (*b*), length of attached part of bar (*l*), and height of the bar (*t*). This problem is illustrated in Fig. 7.

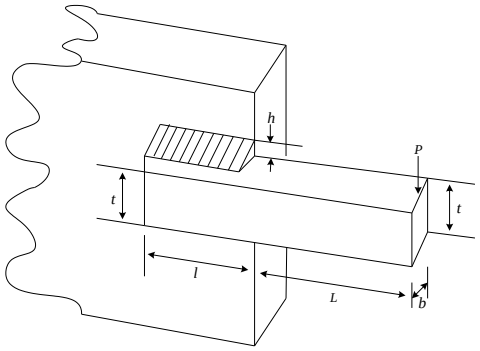


Fig. 7. Schematic view of welded beam design problem

The mathematical formulation of welded beam design problem is defined as follows:

Consider $\mathbf{X} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b]$,

$$\textit{Minimize} \quad f(\mathbf{X}) = 1.10471x_1^2 x_2 + 0.04811x_3x_4(14.0 + x_2),$$

$$\textit{Subject to} \quad g_1(\mathbf{X}) = \tau(\mathbf{X}) - \tau_{\max} \le 0,$$

$$g_2(\mathbf{X}) = \sigma(\mathbf{X}) - \sigma_{\max} \le 0,$$

$$g_3(\mathbf{X}) = \delta(\mathbf{X}) - \delta_{\max} \le 0,$$

$$g_4(\mathbf{X}) = x_1 - x_4 \le 0,$$

$$g_5(\mathbf{X}) = P - P_c(\mathbf{X}) \le 0,$$

$$g_6(\mathbf{X}) = 0.125 - x_1 \le 0,$$

$$g_7(\mathbf{X}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,$$

where $0.1 \le x_1, x_4 \le 2$, $0.1 \le x_2, x_3 \le 10$,

where $\tau(\mathbf{X}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$, $\tau' = \frac{p}{\sqrt{2x_1x_2}}$, $\tau'' = \frac{MR}{J}$, $M = P\left(L + \frac{x_2}{2}\right)$, $R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_2}{2}\right)^2}$,

$$P_c(\mathbf{X}) = \frac{4.013\sqrt{EGx_3^2x_4^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \quad J = 2\left\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \quad \sigma(\mathbf{X}) = \frac{6PL}{x_4x_3^2}, \quad \delta(\mathbf{X}) = \frac{4PL^3}{Ex_4x_3^3},$$

$P = 6000\ lb$, $L = 14\ in.$, $\delta_{\max} = 0.25\ in.$, $E = 30 \times 10^6\ psi$, $G = 12 \times 10^6\ psi$, $\tau_{\max} = 13600\ psi$, $\sigma_{\max} = 30000\ psi$.

The ISCA was applied to solve the welded beam design problem and compared it with 12 optimization algorithms that were reported in previews works as shown in Table 17. From Table 17, the result obtained by HS was the best. Compared with GA, SA, SBM, SCA, DFSA, EMEA, ISA, and MTSA, the ISCA found a better result for the welded beam design problem. With respect to IPSO, NOSA, and DEDS, the ISCA get similar result.

Table 17
The experimental results of ISCA and other algorithms for welded beam design problem.

Algorithm	Optimal values for variables				Optimal cost
	h	*l*	*t*	*b*	
GA (Deb, 2000)	0.2489	6.1730	8.1789	0.2533	2.4331
HS (Lee & Geem, 2005)	0.2442	6.2231	8.2915	0.2443	2.3807
SA (Atiqullah & Rao, 2000)	0.2471	6.1451	8.2721	0.2495	2.4148
SBM (Akhtar et al., 2002)	0.2407	6.4851	8.2399	0.2497	2.4426
SCA (Ray & Liew, 2003)	0.2444	6.2380	8.2886	0.2446	2.3854
IPSO (He et al., 2004)	0.2444	6.2175	8.2915	0.2444	2.3810
NOSA (Liu, 2005)	0.2444	6.2175	8.2915	0.2444	2.3810
DFSA (Hedar & Fukushima, 2006)	0.2444	6.2158	8.2939	0.2444	2.3811
DEDS (Zhang et al., 2008)	0.2444	6.2175	8.2915	0.2444	2.3810
EMEA (Zhang et al., 2009)	0.2443	6.2201	8.2940	0.2444	2.3816
ISA (Gandomi, 2014)	0.2443	6.2199	8.2915	0.2443	2.3812
MTSA (Babalik, et al., 2018)	0.24415742	6.22306595	8.29555011	0.24440474	2.38241101
ISCA (this work)	0.24435	6.2178	8.2919	0.24437	2.3810

5.8 ISCA for training multilayer perceptrons

Multilayer perceptron (MLP) is the most popular and common type of neural network. Fig. 8 shows a general example of MLP with only one hidden layer.

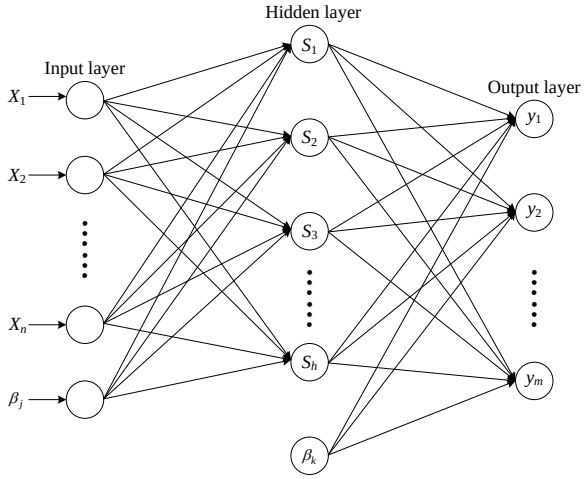


Fig. 8. Multilayer perceptron neural network

After providing the inputs, weights, and biases, the output of MLPs are calculated as follows (Mirjalili, 2015):

The weighted sums of inputs are first calculated by

$$s_j = \sum_{i=1}^{n} (w_{ij} \cdot X_i) + \beta_j, \ j = 1, 2, \ldots, h \tag{7}$$

where n is the number of the input nodes, X_i indicates the ith input, w_{ij} indicates the connection weight from the ith node in the input layer to the jth node in the hidden layer, and β_j is the bias of the jth hidden node.

The output of each hidden node is calculated as follows:

$$S_j = sigmoid(s_j) = \frac{1}{1 + \exp(-s_j)}, \ j = 1, 2, \ldots, h \tag{8}$$

The final outputs are defined as follows:

$$y_k = \frac{1}{1 + \exp\left(-\left(\sum_{j=1}^{h} (W_{jk} \cdot S_j) + \beta_k\right)\right)}, \ k = 1, 2, \ldots, m \tag{9}$$

where y_k is the kth output, W_{jk} indicates the connection weight from the jth hidden node to the kth output node, and β_k indicates the bias of the kth output node.

From Eq. (7) to Eq. (9), the weights and biases are responsible for defining the final output of MLPs from given inputs. A trainer should find a set of values for weights and biases that provides the highest classification accuracy. In this subsection, we describe the proposed approach based on the ISCA for training the MLP network which will be named as ISCA-MLP.

In ISCA-MLP, the dimension of each variable equals the total of weights and biases. The variables of an MLP are provided as follows:

$$\mathbf{V} = \{\mathbf{W}, \boldsymbol{\beta}\} = \{w_{11}, w_{12}, ..., w_{nh}, \beta_1, \beta_2, ..., \beta_h, w_1, w_2, ..., w_h, \beta_k\} \tag{10}$$

For example, the Ionosphere classification dataset has 34 attributes, 351 training/test samples, and 2 classes. Thus, a MLP of 33-67-1 is chosen to be trained and solve this dataset. The trainers for this dataset have dimensions of 2346. Another case is called Wine dataset. There is 13 attributes, 178 training/test samples. A MLP of 13-27-1 is chosen to be trained and solve this dataset. Every individual of the trainers have 406 variables to be optimized. In other words, the dimension of the variable is up to 2346 and this problem is known as a high dimensional problem.

After defining the variables, we need to define the objective function for the ISCA. A common metric for the evaluation of an MLP is the mean square error (MSE) as follows:

$$\text{MSE} = \frac{1}{N} \sum_{p=1}^{N} (y - \hat{y})^2 \tag{11}$$

where y indicates the actual value, $\hat{y}$ indicates the predicted value, and N indicates number of instances in the training dataset.

For all experiments, we used MATLAB R2014a implement the proposed ISCA trainer and SCA, GA, PSO, ACO, DE, ES, and WOA. All experiments are executed for ten different runs, and each run includes 250 iterations. Table 18 provides the statistical results, namely best (Best), average (Mean), and standard deviation (St.dev) of classification accuracy. It is noteworthy that the results of other algorithms (except SCA) in Table 18 were directly extracted from (Aljarah, Faris, & Mirjalili, 2018).

Table 18

Experimental results of ISCA and other algorithms for Ionosphere and Wine dataset.

Algorithm	Ionosphere dataset			Wine dataset		
	Best	Mean	St.dev	Best	Mean	St.dev
GA	0.8250	0.8025	0.0157	0.9394	0.8894	0.0580
PSO	0.7917	0.7600	0.0242	0.8939	0.8227	0.0474
ACO	0.8000	0.7067	0.0484	0.8181	0.6803	0.1098
DE	0.8417	0.7767	0.0340	0.8788	0.7576	0.0763
ES	0.7917	0.7358	0.0281	0.8333	0.7515	0.0436
WOA	0.8667	0.7942	0.0429	0.9545	0.8894	0.0335
SCA	0.7179	0.6436	0.1468	0.8519	0.7852	0.0663
ISCA	**0.8974**	**0.8265**	0.0387	**0.9682**	**0.9065**	0.0266

From Table 18, ISCA trainer outperforms all other trainers' optimizers for Ionosphere and Wine datasets with an average accuracy of 0.8265 and 0.9065, respectively. In addition, the best accuracy obtained by ISCA showed improvements compared to other algorithms employed.

5.9 ISCA for iron ores sintering burdening optimization

As far as we know, the heuristic algorithms have been applied to solve many engineering applications, such as image segmentation (Li, Tao, Liu, & Liu, 2018), respiratory disease detection (Woźniak & Polap, 2018), engineering optimization (Ferreira, Rocha, Neto, & Sacco, 2018), optimal planning of distributed energy resources (Kumawat, Gupta, Jain, & Bansal, 2018), simulation and control of dynamic systems (Woźniak & Polap, 2017), etc.

To further investigate its performance, this subsection applies ISCA to solve the sintering burdening processes optimization problem of iron ores. Sintering is one of the important steps in the iron ores smelting process. Burdening is a key process before the sintering of iron ores, which determines the chemical composition and quality of the agglomerate. The purpose of the burdening process is to reduce the cost of the business while keeping the harmful chemical components within a reasonable range. The proportions of TFe, P, S, Al_2O_3, S_iO_2, and MgO (denoted by e_1, e_2, e_3, e_4, e_5, and e_6, respectively) in seven iron ore materials of a China's steel company and prices are shown in Table 19.

Table 19

The chemical compositions and prices of different iron ore materials.

Iron ore powder	TFe (%)	P (%)	S (%)	Al_2O_3 (%)	S_iO_2 (%)	MgO (%)	Price (￥/ton)
Material 1	61.1	0.18	0.29	1.41	2.38	0.99	385.0
Material 2	52.9	0.06	0.04	1.29	11.1	6.27	319.0
Material 3	65.2	0.07	0.09	1.78	1.42	1.01	470.0
Material 4	62.5	0.02	0.41	1.28	4.99	5.18	431.0
Material 5	65.4	0.03	0.31	1.28	1.09	2.27	493.0
Material 6	59.2	0.03	0.32	7.76	5.68	1.32	299.0
Material 7	59.3	0.04	0.02	2.27	7.23	0.92	342.0

The objective of the burdening process is to ensure that the contents of iron and impurity meet the requirements and minimize the total costs. Thus, the objective function and constraints are formulated as follows:

$$\begin{cases} \min \quad f(\mathrm{X}) = \sum_{k=1}^{7} c_k x_k, k = 1,2,\cdots,7 \\ s.t. \quad 61\% \le \frac{1}{1000}\sum_{k=1}^{7} e_1 x_k \le 62\% \\ \qquad 0 \le \frac{1}{1000}\sum_{k=1}^{7} e_2 x_k \le 0.07\% \\ \qquad 0 \le \frac{1}{1000}\sum_{k=1}^{7} e_3 x_k \le 0.13\% \\ \qquad 0 \le \frac{1}{1000}\sum_{k=1}^{7} e_4 x_k \le 2.0\% \\ \qquad 4.9\% \le \frac{1}{1000}\sum_{k=1}^{7} e_5 x_k \le 5.4\% \\ \qquad 0 \le \frac{1}{1000}\sum_{k=1}^{7} e_6 x_k \le 2.3\% \end{cases} \tag{12}$$

where k is the types of iron ore involved in the burdening process, c_k represents the unit price (￥/kg) of each iron ore, and x_k is the amount of each iron ore (kg).

We utilize the ISCA to solve the iron ores sintering burdening optimization problem (12), and compare it with the particle swarm optimization (PSO), the whale optimization algorithm (WOA) and the basic sine cosine algorithm (SCA) algorithms. For four algorithms, the feasibility-based rule (Deb, 2000) is used to handle constraints. The parameters of four algorithms are set as follows: the population size of four algorithms is set to 30, the total number of iterations is set to 1000. We independently tested each algorithm 30 times. The best results of four algorithms are shown in Tables 20 and 21.

Table 20

Comparisons of the best iron ratio and cost price optimized by different algorithms.

Algorithm	Material 1 (%)	Material 2 (%)	Material 3 (%)	Material 4 (%)	Material 5 (%)	Material 6 (%)	Material 7 (%)	Cost price (￥/ton)
PSO	15.8347	1.0537	25.2269	0	0	0	57.8848	380.8569
WOA	12.4879	1.1109	21.2265	0	4.8189	0	60.3558	381.5608
SCA	16.2840	0	24.0841	2.4617	0	0	57.1701	382.0207
ISCA	16.6715	0	22.9473	1.4385	0	0	58.9428	379.8215

Table 21

The content of each element in the burdening material optimized by different algorithms (%).

Algorithm	TFe	P	S	Al_2O_3	S_iO_2	MgO
PSO	61.0060	0.0699	0.0806	1.9999	5.0371	1.0102
WOA	61.0000	0.0636	0.0828	2.0000	5.1382	1.0723
SCA	61.0929	0.0695	0.0904	1.9876	4.9858	1.0579
ISCA	61.0000	0.0699	0.0867	1.9999	5.0560	1.0136

According to Tables 20 and 21, compared with PSO, WOA, and SCA algorithms, ISCA obtained better results for solving the iron ores sintering blending optimization problem. Assuming that the burdening materials of a company are 1 million tons per year, compared to the PSO, WOA, SCA algorithms, ISCA can reduce the total costs 1.0354 million, 1.7393 million, and 2.1992 million yuan, respectively.

6. Conclusions

As a relatively novel population-based algorithm, the SCA has hitherto been successfully applied in a variety of fields. In this paper, we illustrated the influence of inertia weight and conversion parameter to improve the accuracy of the SCA for solving global optimization with high dimensionality and engineering design problems and proposed an improved version of the SCA (ISCA). The experimental results of the ISCA were compared with basic SCA and

other meta-heuristic algorithms. From these comparisons, we concluded that the ISCA performed better than the basic SCA and other population-based algorithms in terms of convergence speed, time complexity, and convergence precision. However, for function f_6 with D = 1000, the result obtained by the ISCA was worse than those by the TLBO and WOA algorithms. Additionally, the results of the ISCA in solving real-world engineering design problems were compared with several state-of-the-art algorithms and demonstrated encouraging results.

In future work, we will attempt to apply the proposed ISCA in various applications such as parameter optimization, constrained optimization problems, multi-objective optimization, image processing, data mining, feature selection and renewable energy problems.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant No. 61463009, Program for the Science and Technology Top Talents of Higher Learning Institutions of Guizhou under Grant No. KY[2017]070, Science and Technology Foundation of Guizhou Province under Grant No. [2016]1022, Joint Foundation of Guizhou University of Finance & Economics and Ministry of Commerce under Grant No. 2016SWBZD13, Fundamental Research Funds for Beijing University of Civil Engineering and Architecture under Grant No. X18193, Funding for key construction disciplines in Hunan Province, and Innovation Group Major Research Program Funded by Guizhou Provincial Education Department under Grant No. KY[2016]051. The authors appreciate the anonymous reviewers and the editors for their valuable comments and suggestions for improving this paper.

References

Akhtar, S., Tai, K., & Ray, T. (2002). A socio-behavioural simulation model for engineering design optimization. *Engineering Optimization*, *34*, 341−354.

Ali, M. Z., Awad, N. H., & Suganthan, P. N. (2015). Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization. *Applied Soft Computing*, *33*, 304−327.

Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, *22*, 1−15.

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures*, *169*, 1-12.

Atiqullah, M. M., & Rao, S. (2000). Simulated annealing and parallel processing: an implementation for constrained global design optimization. *Engineering Optimization*, *32*, 659−685.

Babalik, A., Cinar, A. C., & Kiran, M. S. (2018). A modification of tree-seed algorithm using Deb's rules for constrained optimization. *Applied Soft Computing*, *63*, 289−305.

Baykasoğlu, A. (2012). Design optimization with chaos embedded great deluge algorithm. *Applied Soft Computing*, *12*, 1055-1067.

Baykasoğlu, A., & Ozsoydan, F. B. (2015). Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Applied Soft Computing*, *36*, 152-164.

Brajevic, I., & Tuba, M. (2013). An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems. *Journal of Intelligent Manufacturing*, 24, 729-740.

Cano, A., & García-Martínez, C. (2016). 100 million dimensions large-scale global optimization using distributed GPU computing. in: *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 3566−3573.

Cano, A., García-Martínez, C., & Ventura, S. (2017). Extremely high-dimensional optimization with MapReduce: scaling functions and algorithm. *Information Sciences*, 415-416, 110−127.

Cheng, R., & Jin, Y. (2014). A competitive swarm optimizer for large scale optimization. *IEEE Transactions on Cybernetics*, *45*, 191−204.

Chu, W., Gao, X., & Sorooshian, S. (2011). A new evolutionary search strategy for global optimization of high-dimensional problems. *Information Sciences*, *181*, 4909−4927.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, *16*, 193-203.

Das, S., Bhattacharya, A., & Chakraborty, A. K. (2017). Solution of short-term hydrothermal scheduling using sine cosine algorithm. *Soft Computing*, *22*, 6409−6427.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics & Engineering*, *186*, 311−338.

Elaziz, M. A., Oliva, D., & Xiong, S. (2017). An improved opposition-based sine cosine algorithm for global optimization. *Expert Systems with Applications*, *90*, 484−500.

Ferreira, M. P., Rocha, M. L., Neto, A. J. S., & Sacco, W. F. (2018). A constrained ITGO heuristic applied to engineering optimization.

Expert Systems with Applications, *110*, 106−124.

Gandomi, A. H. (2014). Interior search algorithm (ISA): a novel approach for global optimization. *ISA Transactions*, *53*, 1168−1183.

Gandomi, A. H., Yang, X., & Alavi, A. H. (2013). Cuckoo search algorithm: a meta-heuristic approach to solve structural optimization problems. *Engineering Applications of Artificial Intelligence*, *29*, 17-35.

Gandomi, A. H., Yang, X., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, *22*, 1239-1255.

Gao, W., Liu, S., & Huang, L. (2013). A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Transactions on Cybernetics*, *43*, 1011−1024.

Hang, F., Wang, L., & He, Q. (2007). An effective coevolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation*, *186*, 340-356.

Hansen N, Müller S D, Koumoutsakos P. (2014). Reducing the time complexity of the derandomed evolution strategy with covariance matrix adaptation. *Evolutionary Computation*, *11*, 1−18.

Hedar, A. R., & Fukushima, M. (2006). Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, *35*, 521−549.

He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, *20*, 89−99.

He, S., Prempain, E., & Wu, Q. (2004). An improved particle swarm optimization for mechanical design optimization problems. *Engineering Optimization*, *36*, 585−605.

Huang, F, Li, X, & Zhang, S. (2018). Harmonious genetic clustering. *IEEE Transactions on Cybernetics*, *48*, 199−214.

Imanian, N., Shiri, M. E., & Moradi, P. (2014). Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems. *Engineering Applications of Artificial Intelligence*, *36*, 148−163.

Issa, M., Hassanien, A. E., Oliva, D., Helmi, A., Ziedan, I., & Alzohairy, A. (2018). ASCA-PSO: Adaptive sine cosine optimization algorithm integrated with particle swarm for pairwise local sequence alignment. *Expert Systems with Applications*, *99*, 56−70.

Jain, M., Singh, V., & Rani, A. (2018). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation*, https://doi.org/10.1016/j.swevo.2018.02.013.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computation*, *27*, 155-182.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *IEEE International Conference on Neural Network*, *4*, 1942−1948.

Kong, X., Gao, L., Ouyang, H., & Li, S. (2015). Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm. *Computers & Operations Research*, 63, 7−22.

Krohling, R. A., & Coelho, L. D. S. (2006). Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 36, 1407-1416.

Kumawat, M., Gupta, N., Jain, N., & Bansal, R. C. (2018). Optimal planning of distributed energy resources in harmonics polluted distribution system. *Swarm and Evolutionary Computation*, *39*, 99−113.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34, 2743−2757.

Lastra, M., Molina, D., & Benítez, J. M. (2015). A high performance memetic algorithm for extremely high-dimensional problems. *Information Sciences*, *293*, 35−58.

Lee, K.S., & Geem, Z.W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics & Engineering*, 194: 3902−3933.

Li, K., Tao, W., Liu, X., & Liu, L. (2018). Iterative image segmentation with feature driven heuristic four-color labeling. *Pattern Recognition*, *76*, 69−79.

Li, S., Fang, H., & Liu, X. (2018). Parameter optimization of support vector regression based on sine cosine algorithm. *Expert Systems with Applications*, *91*, 63−77.

Li, X., & Yao, X. (2012). Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, *16*, 210−224.

Li, Z., Wang, W., Yan, Y., & Li, Z. (2015). PS-ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. *Expert Systems with Applications*, *42*, 8881−8895.

Liu, F., & Zhou, Z. (2014). An improved QPSO algorithm and its application in the high-dimensional complex problems. *Chemometrics and Intelligent Laboratory Systems*, *132*, 82−90.

Liu, H., Cai, Z. & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, *10*, 629-640.

Liu, J. L. (2005). Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems. *Engineering Optimization*, *37*, 499−519.

Long, W., Jiao, J., Liang, X., & Tang, M. (2018). An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. *Engineering Applications of Artificial Intelligence*, *68*, 63−80.

Luo, J., Wang, Q., & Xiao, X. (2013). A modified artificial bee colony algorithm based on converge-onlookers approach for global optimization. *Applied Mathematics and Computation*, *219*, 10253−10262.

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, *188*, 1567-1579.

Meshkat, M., & Parhizgar, M. (2017a). A novel weighted update position mechanism to improve the performance of sine cosine algorithm. *5th Iranian Joint Congress on Fuzzy and Intelligent Systems* (CFIS 2017), 166−171.

Meshkat, M., & Parhizgar, M. (2017b). A novel sine and cosine algorithm for global optimization. *7th International Conference on*

Computer and Knowledge Engineering (ICCKE 2017), 60−65.

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-based Systems, 89*, 228−249.

Mirjalili, S. (2015). How effective is the grey wolf optimizer in training multi-layer perceptrons. *Applied Intelligence, 43*, 150−161.

Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-based Systems, 96*, 120−133.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163−191.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51−67.

Mirjalili, S., Mirjalili S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*, 495-513.

Mirjalili, S., Mirjalili S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Mezura-Montes, E., & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems, 37*, 443-473.

Ning, J., Zhang, Q., Zhang, C., & Zhang, B. (2018). A best-path-updating information-guided ant colony optimization algorithm. *Information Sciences, 433-434*, 142−162.

Mohapatra, P., Das, K. N., & Roy, S. (2017). A modified competitive swarm optimizer for large scale optimization problems. *Applied Soft Computing, 59*, 340−362.

Molina, D., Lozano, M., & Herrera, F. (2010). MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization. in: *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 3153−3160.

Nenavath, H., & Jatoth, R. K. (2017). Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking. *Applied Soft Computing, 62*, 1019−1043.

Polap, D., & Woźniak, M. (2017). Polar bear optimization algorithm: meta-heuristic with fast population movement and dynamic birth and death mechanism. *Symmetry, 9*, 1−20.

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2012). Teaching-learning-based optimization: An optimization method for continuous nonlinear large scale problems. *Information Sciences, 183*, 1−15.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences, 179*, 2232-2248.

Ray, T., & Liew, K.M. (2003). Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation, 7*, 386−396.

Rizk-Allah, R. M. (2018). Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. *Journal of Computational Design and Engineering, 5*, 249−273.

Rodriguez, F., Lozano, M., Blum, C., & Garcia-Martinez, C. (2013). An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. *Computers & Operations Research, 40* (7), 1829−1841.

Sharma, A., Sharma, A, Panigrahi, B. K., Kiran, D., & Kumar, R. (2016). Ageist spider monkey optimization algorithm. *Swarm and Evolutionary Computation, 28*, 58−77.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *IEEE International Conference on Evolutionary Computation, 1*, 69−73.

Sindhu, R., Ngadiran, R., Yacob, Y. M., Zahri, N. A. H., & Hariharan, M. (2017). Sine-cosine algorithm for feature selection with elitism strategy and new updating mechanism. *Neural Computing & Applications, 28*, 2947−2958.

Singh, D., & Agrawal, S. (2016). Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. *Applied Soft Computing, 38*, 1040−1048.

Storn, R., & Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous space. *Journal of Global Optimization, 11*, 341−359.

Tang, K., Li, X., Suganthan, P. N., Yang, Z., & Weise, T. (2010). Benchmark functions for the CEC2010 special session and competition on large scale global optimization. *Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, & Nanyang Technological University*, 2010.

Tawhid, M. A., & Savsani, V. (2017). Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems. *Neural Computing & Applications*, http://dx.doi.org/10.1007/s00521-017-3049-x.

Trivedi, A., Srinivasan, D., Biswas, S., & Reindl, T. (2015). Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation, 23*, 50−64.

Tsai, J. T., Liu, T. K., & Chou, J. H. (2008). Hybrid taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 8*, 365−377.

Tuo, S., Zhang, J., Yong, L., Yuan, X., Liu, B., Xu, X., & Deng, F. (2015). A harmony search algorithm for high-dimensional multimodal optimization problems. *Digital Signal Processing, 46*, 151−163.

Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*, 225−239.

Wang, C., & Gao, J. H. (2014). A differential evolution algorithm with cooperative coevolutionary selection operation for high-dimensional optimization. *Optimization Letters, 8*, 477−492.

Wang, H., Rahnamayan, S., & Wu, Z. (2013). Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. *Journal of Parallel & Distributed Computing, 73*, 62−73.

Wang, J., Yang, W., Du, P., & Niu, T. (2018). A novel hybrid forecasting system of wind speed based on a newly developed multi-objective sine cosine algorithm. *Energy Conversion and Management, 163*, 134-150.

Wang, Y., Li, B., & Lai, X. (2009). Variance priority based cooperative co-evolution differential evolution for large scale global optimization. in: *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1232−1239.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation,*

1, 67–82.

Woźniak, M., & Polap, D. (2017). Hybrid neuro-heuristic methodology for simulation and control of dynamic systems over time interval. *Neural Networks*, *93*, 45–56.

Woźniak, M., & Polap, D. (2018). Bio-inspired methods modeled for respiratory disease detection from medical images. *Swarm and Evolutionary Computation*, *41*, 69–96.

Xue, Y., Zhong, S., Zhuang, Y., & Xu, B. (2014). An ensemble algorithm with self-adaptive learning techniques for high-dimensional numerical optimization. *Applied Mathematics and Computation, 231*, 329–346.

Yang, Z., Tang, K., & Yao, X. (2008). Multilevel cooperative co-evolution for large scale optimization. in: *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1663–1670.

Yurtkuran, A., & Emel, E. (2015). An adaptive artificial bee colony algorithm for global optimization. *Applied Mathematics and Computation, 271*, 1004–1023.

Zhang, F., & Chiang, H. D. (2017). A novel consensus-based particle swarm optimization-assisted trust-tech methodology for large-scale global optimization. *IEEE Transactions on Cybernetics, 47*, 2717–2729.

Zhang, J., Liang, C., Huang, Y., Wu, J., & Yang, S. (2009). An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. *Applied Mathematics and Computation*, *211*, 392–416.

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, *178*, 3043–3074.

Zhao, S., Suganthan, P. N., & Das S. (2011). Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing, 15*, 2175–2185.