Knowledge-Based Systems 251 (2022) 109189

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# A new metaheuristic optimization based on K-means clustering algorithm and its application to structural damage identification

Hoang-Le Minh [a,b], Thanh Sang-To [b], Magd Abdel Wahab [c,a,*], Thanh Cuong-Le [b,*]

[a] Department of Electrical Energy, Metals, Mechanical Constructions, and Systems, Faculty of Engineering and Architecture, Ghent University, 9000 Gent, Belgium
[b] Faculty of Civil Engineering, Ho Chi Minh City Open University, Ho Chi Minh City, Viet Nam
[c] Faculty of Mechanical - Electrical and Computer Engineering, School of Engineering and Technology, Van Lang University, Ho Chi Minh City, Viet Nam

ARTICLE INFO

*Article history:*
Received 21 December 2021
Received in revised form 27 May 2022
Accepted 30 May 2022
Available online 7 June 2022

*Keywords:*
Metaheuristic optimization
K-means clustering algorithm
Engineering problems
Structural damage identification
SAP2000-OAPI

ABSTRACT

This paper develops a new metaheuristic optimization algorithm named K-means Optimizer (KO) to solve a wide range of optimization problems from numerical functions to real-design challenges. First, the centroid vectors of clustering regions are established at each iteration using K-means algorithm, then KO proposes two movement strategies to create a balance between the ability of exploitation and exploration. The decision on the movement strategy for exploration or exploitation at each iteration depends on a parameter that will be designed to recognize if each search agent is too long in the region visited with no self-improvement. To demonstrate the effectiveness and reliability of KO, twenty-three classical benchmark functions, CEC2005 and CEC2014 benchmark functions, are employed as a first example and compared with other algorithms. Then, three well-known engineering problems are also considered and their results are compared to the results obtained by the other algorithms. Finally, KO is applied to structural damage identification (SDI) problem of a complex 3D concrete structure including seven stories building having a 25.2 m total height. For this purpose, SAP2000 is used to solve the finite element (FE) model of this structure. Then, for the first time, we successfully developed a sub-program that allows two-way data exchange between SAP2000 and MATLAB through the Open Application Programming Interface (OAPI) library to update the FE model. From the results, we found that KO has the best performance for the considered benchmark functions based on the Wilcoxon rank-sum test and Friedman ranking test. The results obtained in this work have proved the effectiveness and reliability of KO in solving optimization problems, especially for SDI. Source codes of KO is publicly available at http://goldensolutionrs.com/codes.html.



## 1. Introduction

Metaheuristic optimization algorithms have emerged drastically over the past two decades and have been widely applied to a large number of real-world engineering problems. When the classical mathematics approach and numerical analysis fail to handle the exact solutions, metaheuristics are selected as an alternative to solve the optimization problems more quickly. The primary characteristics of the metaheuristic technique can be summarized as follows.

- Metaheuristics are strategies that lead to the search process of "trial and error".
- The goal is to explore the potential search spaces for finding the best solutions and try to escape the local optimum.
- Techniques that constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate approaches and are not problem-specific.

No Free Lunch Theorem of Optimization [1] proved that no optimization algorithm showed the best overall different types of optimization problems. It means that metaheuristics perform better on particular optimization problems and not as good on others. From this perspective, two main trends are being studied today; (1) the improvement of current algorithms and (2) the development of a new optimization algorithm.

The application of K-means clustering algorithm technique for improving the movement strategies of the metaheuristic optimization algorithms has been widely used in the literature.

* Corresponding authors.
*E-mail addresses:* magd.a.w@vlu.edu.vn, magd.abdelwahab@ugent.be (M. Abdel Wahab), cuong.lt@ou.edu.vn (T. Cuong-Le).

https://doi.org/10.1016/j.knosys.2022.109189
0950-7051/

The combination between K-means clustering and PSO [2] improved the performance of the convergence rate and the accuracy level [3,4]. A novel hybrid K-means metaheuristic and Grey Wolf optimizer (GWO) [5] were presented in Ref. [6]. To prove the effectiveness of the proposed method, three cluster assignment heuristics are employed to assign customers to clusters. The resulting three variants of the algorithm are tested using several benchmark problems. The results obtained in this study were promising. The simulation results suggested that the proposed method outperforms other algorithms such as PSO.

In this paper, the advantages of K-means clustering algorithm is used to develop a new metaheuristic optimization algorithm named K-means optimizer (KO). The results obtained in this paper are promising, and the KO can be considered as a potential selection to solve optimization problems, especially structural health monitoring (SHM).

The original contributions of this paper are as follows:

- For the first time, K-means clustering algorithm obtains the best out of the advantages to establish a new metaheuristic optimization (KO), which makes use of K-means algorithm and two movement strategies to make a balance between exploitation and exploration abilities. First, K-means algorithm is employed to find the centroid vectors of clustering regions. Then, based on these centers, the processing of finding the best solution in KO is carried out following two movement strategies. The first movement strategy is established by three trends based on the centroid vectors and the best solution exploited at each iteration to effectively explore the feasible search spaces. And the second movement strategy is designed by two trends, whose characteristics are to reset the search spaces in poor regions. Thus, KO achieves an excellent balance to ensure the success of the algorithm.
- For the purpose using KO to solve the structural damaged detection (SDI) problem, a program, which allows the connection between SAP2000 and MATLAB, is successfully developed. Based on this program, a new model updating technique is proposed. The new technique allows the FE model to update the parameters according to the user's aim. This technique is used not only for damage detection, but also for topology design problems. It promises to be a powerful program and will be explored for other related studies in future.
- Based on the new model updating technique, a 3D concrete structures having complex shapes are used as examples to demonstrate the advantages of the new model updating technique. To the best of our knowledge, this is the first paper that considers complex structures. Most of the previous studies in the literature were only applied to simple structures or simple geometries.

In the remaining parts of the paper, Section 2 introduces inspired metaheuristic algorithms. Section 3 introduces the theory of K-means clustering algorithm. Section 4 explains the KO principles and introduces the schematic representation of KO, which describes the proposed algorithm's theories. In Section 5, the classical benchmark functions are used to evaluate KO's performance and efficiency. The performance of KO for solving the well-known engineering problems and SDI is evaluated in Section 6 and Section 7. Finally, the conclusion and future works are summarized in the last section.

## 2. Related works

There are different ways to classify and describe metaheuristic algorithms. If we use the number of solutions as the classification criterion, metaheuristics can be divided into two groups: (i) a single solution-based optimization algorithm, which focuses on adapting and improving a single candidate solution, and (ii) population-based metaheuristics, which perform search processes for the evolution of a set of solutions in the search space. Tabu Search [7], Iterated Local Search [8], Variable Neighborhood Search [9] are optimization algorithms that belong to the first group. The advantages of this group are lower computational costs, but they suffer from early convergence and the accuracy level obtained from this group is not appreciated. The second group includes a large number of optimization algorithms, such as Particle swarm optimization (PSO) [2], Ant colony optimization (ACO) [10], Artificial bee colony algorithm (ABC) [11], Cuckoo search (CS) [12], etc. The advantage of these algorithms is that each solution candidate is constrained to the best solution found at the previous step. Thus, the population can be self-improved over the course of iterations by sharing information among individuals in the populations. This group shows a high ability to avoid local optima and to find the best solutions compared to the first group because a set of solutions is involved at the same time during the process of iterations, especially with a large search spaces [13–15]. Besides these advantages, high computational cost and the need for more function evaluation are two major drawbacks of population-based algorithms.

If we rely on the source of inspiration to develop a search strategy in an optimization algorithm, metaheuristics can be classified into four group: (i) evolution-inspired, (ii) physics-inspired, (iii) swarm intelligence-inspired and (iv) human-inspired. In these groups, swarm intelligence-inspired is employed to propose the new optimization algorithms in large quantities. These algorithms mostly mimic the collective behavior of swarms of insects, herds of ungulates, flocks of birds or schools of fish observed in nature [16,17], The mechanism is almost similar to physics-based algorithm, but the search strategies are carried out by agents that navigate using the simulated collective intelligence specific to group-living species [18–20]. These algorithms have become popular in solving optimization problems because of their strong global searching abilities. Some of the most recent and popular algorithms in each of these subclasses are as follows:

- Evolution-inspired: Evolution Strategy (ES), Genetic algorithm (GA) [21], Differential Evolution (DE) [22], Biogeography-Based Optimization algorithm (BBO) [23].
- Physics-inspired: Simulated annealing (SA) [24], Gravitational Local Search (GLSA) [25], Gravitational Search Algorithm (GSA) [26], Charged System Search (CSS) [27], Central Force Optimization (CFO) [28], etc.
- Swarm intelligence-inspired: Wasp Swarm Algorithm (WSO) [29], Wolf pack search algorithm [30], Firefly Algorithm (FA) [31], Grey Wolf optimizer (GWO) [5], The Whale Optimization Algorithm (WOA) [32], African Vultures Optimization Algorithm (AVOA) [33], etc.
- Human-inspired: Teaching–Learning-Based Optimization (TLBO) [34], League Championship Algorithm (LCA) [35], Anarchic Society Optimization (ASO) [36], Artificial hummingbird algorithm (AHA) [37], etc.

The mathematical background of almost all optimization algorithms is stochastic optimization. Algorithms try to create a set of random variables and assign these variables to agents in a population. The value of the objective function, which is characterized by these random variables, is then calculated to identify the best solution. Through each iteration, the process, which generates a new set of random variables through the position updating of each agent, is executed continuously to update the new value of the objective function. Thus, the best compromise solution for the balance between exploration and exploitation needs to

be determined to guarantee the success of each metaheuristic optimization algorithm. To obtain the best compromise solution, normally, metaheuristic algorithms will be inspired by the laws of nature, try to convert these laws into mathematical formulations.

GA algorithm [21] was proposed in 1977 by Holland. This algorithm was inspired by Darwin's theory of evolution. In GA, each variable was considered a gene. Based on the value of the objective function calculated at the previous iteration, a random selection of them for creating the new genes was secured at the next iteration. In this way, each new gene will have a higher probability of finding the global optimum. These mechanisms were set up similar to what is happening in nature. Eventually, some of the individuals' genes in the population are changed randomly to mimic mutation. The success of GA algorithm proved the effectiveness of applying the natural laws to optimization problems. It created a strong motive power for the new field of optimization techniques. Particle Swarm Optimization (PSO) [2] is the most popular swarm intelligence algorithm to solve continuous optimization problems. The concept of PSO can be perceived as originality, and it has been a rich source of inspiration for researchers to propose new algorithms in the past two decades. Like to the GA algorithm, PSO mimics the collective behavior of swarms of insects, herds of ungulates, flocks of birds, or schools of fish observed in nature. The primary idea in PSO includes two steps. In the first step, PSO tries to create a balance between exploration and exploitation through velocity updating. Thus, at each iteration, the local best of the individual population and the global best found at the previous iteration will be combined by two random vectors, whose values are in the range [0, 1], to create a velocity vector. In the second step, the velocity vector is used as a separate strategy to move to a new position between the local and the global optimum. The new position is then updated by the combination of the current position and its velocity. PSO has been widely applied to optimization problems in many fields [38,39].

The new position updating strategies, that can be considered effective, have been introduced recently in Grey Wolf optimizer (GWO) [5], Cuckoo search (CS) [12], The whale optimization algorithm (WOA) [32]. GWO and WOA used an absolute vector instead of a variable vector (exact vector) to move from the current position to new position bounding the global best. By this way, the convergence rate in GWO and WOA performed a more complete improvement than other algorithms. CS used a random walk, whose steps obey Lévy-flight, to explore the new search spaces. Thus, CS shows more bias towards the ability of exploitation. If the number of iterations is sufficient, the accuracy level in CS is higher than that of other algorithms, and vice versa, the convergence rate is the major drawback of this optimization algorithm. It is observed that each algorithm has its own advantages and disadvantages that belong to more bias towards the ability of exploitation or exploitation. If it is more bias towards the ability of exploitation, the convergence rate is appreciated, and by contrast, the accuracy level is acceptable. As previously stated, good optimization algorithms require a mix of exploration and exploitation, as well as the ability to escape from the local optimum. These ideas underpin the KO proposed in this paper. At the same time, KO demonstrates that a metaheuristic algorithm does not need to be inspired by natural laws. Simple mathematical functions paired with the K-means clustering technique handle the major characteristics of the optimization algorithm in KO. In particular, instead of each search agent like in other algorithms, the process of finding the optimum solution in KO is assured by the self-improvement of an initial population. In this method, each search agent's information will be kept according to its compatible location in the initial population. Instead of improving themselves, each search agent can wander around freely to find new search spaces. This is promising for the appreciative ability to escape from local optima's in KO compared to the other algorithms.

## 3. K-means clustering algorithm

The K-means algorithm divides a set of data into K cluster-defined discrete non-overlapping subgroups, with each data point belonging to a separate group [40]. It seeks to maintain the intra-cluster data points as comparable as possible, while keeping the clusters as distinct as possible. It assigns data points to clusters by minimizing the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points in that cluster). The less variance within cluster, the more homogeneous (similar) the data points are. To find $K$ centroids, the problem is to minimize the objective function based on the data points $X = \{x_1, x_2, \ldots, x_n\}$ and the centroids position $C = \{c_1, c_2, \ldots, c_K\}$ as given in Eq. (1).

$$f_{obj}(X, C) = \sum_{i=1}^{n} \min \left\{ \|x_i - c_l\|^2,\ l = (1, 2, \ldots, K) \right\} \tag{1}$$

The steps of K-means clustering algorithm are shown in **Algorithm 1**.

In summary, with a set of input data, by applied K-means clustering algorithm, we can identify the centroid vector $C = \{c_1, c_2, \ldots, c_K\}$ with $K$ being the number of centroids defined by user. Fig. 1a illustrates a data set in $2D$ space distributed randomly with $-100 \leq x_i, y_i \leq 100$, and Fig. 1b presents the result of K-means clustering with the number of centroids $K = 3$.

## 4. K-means clustering optimizer (KO)

### 4.1. The process of centroid vector position updating in KO

The background for the development of KO is based on a stochastic algorithm and a population-based approach. In practically, in all optimization algorithms, the search for the global optimum is based on a strategy of self-improvement of each individual position in the population over the course of iterations. The possible search spaces from the previous iteration will be recorded to guide each individual position for position updating in the next iterations. Thus, the population tries to close these search spaces to exploit the other good search spaces. If the iterations are sufficient, we can reach on the required convergence rate and acceptable accuracy level. The inspiration from the laws of nature for simulating the process of self-improvement of the population is a primary trend to develop new optimization algorithms. However, this will cause a heavy dependence on those laws. KO is an exception to the rule, which is inspired by the K-means clustering algorithm to find the potential search spaces. At the same time, the basic mathematical equations will be applied flexibly to achieve two crucial factors. The first factor is the balance between the local optimum and the global optimum, and the second is the ability to escape from the local optima.

KO is a population-based approach including $N$ of population size $P = \left\{\overrightarrow{P_1}, \overrightarrow{P_2}, \ldots, \overrightarrow{P_N}\right\}$, where $\overrightarrow{P_i} = \left\{p_{i,1}, p_{i,2}, \ldots, p_{i,D}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, \ldots, N)$ is defined as an individual position in $D$ dimension. The population in KO can be represented by the following matrix as shown in Eq. (2).

$$P = \begin{Bmatrix} \overrightarrow{P_1} \\ \overrightarrow{P_2} \\ \vdots \\ \vdots \\ \overrightarrow{P_n} \end{Bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & \cdots & p_{1,D} \\ p_{2,1} & p_{2,2} & \cdots & \cdots & p_{2,D} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{N,1} & p_{N,2} & \cdots & \cdots & p_{N,D} \end{bmatrix}^{N\times D} \tag{2}$$

**Algorithm 1: The process of K-means algorithm**

***Step 1:***

*Select K cluster centroids* $C=\{c_1,c_2,...,c_K\}$ *randomly from n data points* $X=\{x_1,x_2,...,x_n\}$ *with* $(K\le n)$.

***Step 2:***

*Assign each point* $x_i, i=(1,2,...,n)$ *to cluster* $c_j$, $j\in\{1,2,...,K\}$ *If* $\|x_i-c_j\|\le\|x_i-c_p\|$, $p=(1,2,...,K)$ *and* $j\neq p$

***Step 3:***

*Update the new cluster centroids* $C^*=\{c_1^*,c_2^*,...,c_K^*\}$ *as follows*

$C_i^*=\frac{1}{n_i}\sum_{x_j\in c_i}x_j, i=(1,2,...,K)$, *where* $n_i$ *is the number of elements belonging to cluster* $c_i$

***Step 4:***

*If termination criteria satisfied, stop otherwise continues from step 2*

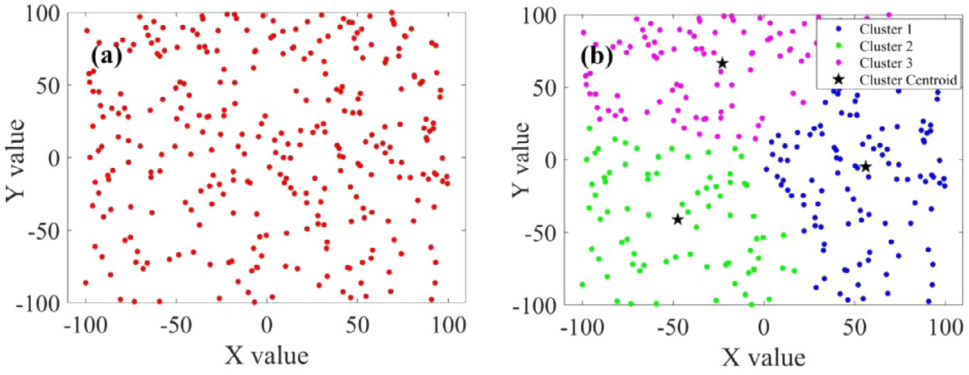


**Fig. 1.** K-means clustering: (a) a set of datasets and (b) closest cluster centroid with $K = 3$.

The *Fitness* of each individual position is then calculated by transmitting the vector of its position to the objective function defined by user *F(x)*. These values will be stored in the vector as shown in Eq. (3).

$$Fitness=\begin{Bmatrix}Fitness_1=F(\overrightarrow{P_1})\\Fitness_2=F(\overrightarrow{P_2})\\\vdots\\\vdots\\Fitness_N=F(\overrightarrow{P_N})\end{Bmatrix}^{N\times 1} \tag{3}$$

Based on the vector of *Fitness,* we can depict the quality of the search spaces. To increase the opportunity of finding a better position and reaching a fast convergence rate, only the search spaces that achieve a probability of finding a high-quality search space will be considered. The remaining search spaces, which show poor quality, will be ignored. To achieve this, KO uses a suitable technique, which is mechanistic sort in a potential matrix symbolized $\overline{P}$. Thus, the potential matrix is established by arranging the positions $\overrightarrow{P_i}=\{p_{i,1},p_{i,2},\ldots,p_{i,D}\}\in\mathbb{R}^{1\times D}, (i=1,2,\ldots,N)$ in population and their *Fitness* in ascending order according to their objective function value. This process is illustrated by Eq. (4)

$$\overline{P}=\begin{bmatrix}\overline{p_{1,1}} & \overline{p_{1,2}} & \cdots & \cdots & \overline{p_{1,D}} & \overline{Fitness_1}\\ \overline{p_{2,1}} & \overline{p_{2,1}} & \cdots & \cdots & \overline{p_{2,D}} & \overline{Fitness_2}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \overline{p_{N,1}} & \overline{p_{N,2}} & \cdots & \cdots & \overline{p_{N,D}} & \overline{Fitness_N}\end{bmatrix}^{N\times D} \tag{4}$$

Where

$$\overline{Fitness_1}\le\overline{Fitness_2}\le\overline{Fitness_3}\le\cdots\le\overline{Fitness_N} \tag{5}$$

After finding the potential matrix $\overline{P}$, to enhance the quality of the search spaces as well as to consider the high-quality position, KO use the Linear Population Size Reduction (LPSR) method. The population sizes in KO will be reduced with a linear function as the number of iterations increases as given in Eq. (6).

$$f(t)=\frac{(N-4)t}{1-T_{\max}}+N-\frac{(N-4)}{1-T_{\max}} \tag{6}$$

Where $t$th is the current iteration and $T_{\max}$ is the maximum number of iterations. Thus, the population size at the first iteration ($t = 1$) is $N$, and at the last iteration ($t = T_{\max}$) is four. The population size in the potential matrix $\overline{P(t)}$ will be reduced from

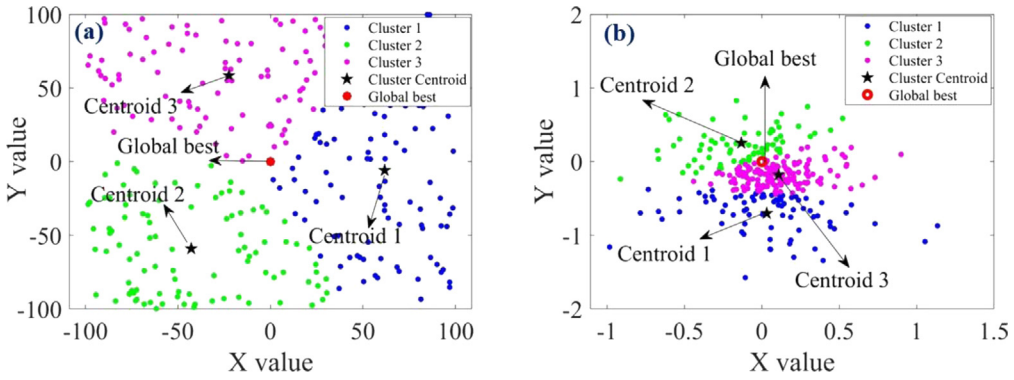


**Fig. 2.** The process of centroid vectors positions updating in two-dimensional search space over ten iterations.; (a) with 5 iterations and (b) with 10 iterations.

$N$ to $f(t)$ as shown in Eq. (7).

$$\overline{P(t)} = \begin{bmatrix} \overline{p_{1,1}^{(t)}} & \overline{p_{1,2}^{(t)}} & \cdots & \cdots & \overline{p_{1,D}^{(t)}} & \overline{Fitness_1^{(t)}} \\ \overline{p_{2,1}^{(t)}} & \overline{p_{2,1}^{(t)}} & \cdots & \cdots & \overline{p_{2,D}^{(t)}} & \overline{Fitness_2^{(t)}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \overline{p_{f(t),1}^{(t)}} & \overline{p_{f(t),2}^{(t)}} & \cdots & \cdots & \overline{p_{f(t),D}^{(t)}} & \overline{Fitness_{f(t)}^{(t)}} \end{bmatrix}^{f(t)\times D} \tag{7}$$

In KO, the number of clusters is homologous to the number of centroids are selected $K = 3$ for the purpose of reaching a fast-high-quality search space. If we consider $\overline{P(t)}$ as a set of input data at any iteration, we can establish the centroid vector $C(t)$ including three rows and $D$ columns calculated by Eq. (8).

$$\overline{P(t)} \xrightarrow[A\lg orithm1]{K-means} C(t) = \begin{Bmatrix} C_1(t) \\ C_2(t) \\ C_3(t) \end{Bmatrix} = \begin{bmatrix} c_{1,1}^{(t)} & c_{1,2}^{(t)} & \cdots & \cdots & c_{1,D}^{(t)} \\ c_{2,1}^{(t)} & c_{2,2}^{(t)} & \cdots & \cdots & c_{2,D}^{(t)} \\ c_{3,1}^{(t)} & c_{3,2}^{(t)} & \cdots & \cdots & c_{3,D}^{(t)} \end{bmatrix}^{3\times D} \tag{8}$$

It can be realized each column $\left\{\overline{p_{1,m}^{(t)}}, \overline{p_{2,m}^{(t)}}, \ldots, \overline{p_{f(t),m}^{(t)}}\right\}^T, m = (1, 2, \ldots, D)$ of $\overline{P(t)}$ in Eq. (7), in which the number of cluster is selected as $K = 3$, we can find its compatible centroid vector column $\left\{c_{1,m}^{(t)}, c_{2,m}^{(t)}, c_{3,m}^{(t)}\right\}^T, m = (1, 2, \ldots, D)$ by applying K-means clustering algorithm. And by assembling this column centroid vector while increasing the number of columns from 1 to $D$, we can record a centroid vector $\overrightarrow{C(t)}$ as shown in Eq. (8). Thus, $\overrightarrow{C(t)}$ can be considered as a higher potential search space than $\overline{P(t)}$.

As the iterations increase, each individual position $\overrightarrow{P_i} = \left\{p_{i,1}, p_{i,2}, \ldots, p_{i,D}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, \ldots, N)$ will close to the best solution. As a result, the centroid vector $\overrightarrow{C(t)}$ is also improved as shown in Fig. 2

Fig. 2 shows that the centroid vector positions are always generated around the best solution exploited at the previous iteration. And these positions are getting closer and closer to the best solution as the number of iterations increases. Especially, these positions do not overlap with the best solution, so it avoids the ability to stick at local optima. These centroid vector positions, together with the best solution will establish a new trend of position updating, which considers the primary background in the KO algorithm. The process of centroid vector position updating is implemented in **Algorithm 2**.

### 4.2. The strategy of movement in KO

KO algorithm shows a different research trend compared to the other algorithms. KO focuses on the strategy of perfecting the initial population $P = \left\{\overrightarrow{P_1}, \overrightarrow{P_2}, \ldots, \overrightarrow{P_N}\right\}$ instead of the search agents as the other algorithms. To achieve this purpose, KO will create a search agents' population $SA = \left\{\overrightarrow{SA_1}, \overrightarrow{SA_2}, \ldots, \overrightarrow{SA_N}\right\}$ with respect to the initial population $P = \left\{\overrightarrow{P_1}, \overrightarrow{P_2}, \ldots, \overrightarrow{P_N}\right\}$. In other words, each individual population $\overrightarrow{P_i} = \left\{p_{i,1}, p_{i,2}, \ldots, p_{i,D}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, \ldots, N)$ will be improved by each its individual search agent $\overrightarrow{SA_i} = \left\{sa_{i,1}, sa_{i,2}, \ldots, sa_{i,D}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, \ldots, N)$ through the greedy selection strategy as shown in **Algorithm 3**.

Based on Algorithm 3, the better position of each search agent $\overrightarrow{SA_i^{(t)}}$ will be stored by its compatible position ($\overrightarrow{P_i}$) in the initial population. And the movement strategy of each search agent in KO is more biased towards fluctuation to achieve the balance between the ability of exploration and exploitation regardless to its convergence like the other algorithms. For this purpose, KO proposes two movement strategies. The first strategy focuses on the movement to exploit the search regions with high-quality of sensitivity. These regions are defined to close the search spaces, which are rounded by the centroid vector positions and the best solution. The second strategy is bias towards the expanding of search spaces to escape local optimum. In KO, based on information about its position at $(t)$th iteration, an intelligent selection is presented to guide the movement of each search agent at $(t+1)$th iteration. KO will calculate a relative distance between its current position and the best solution, the worst solution exploited at $(t)$th iteration as shown in Eq. (9).

$$D_{SA_i}(t) = \frac{\left\| \overrightarrow{SA_i^{(t)}} - \overrightarrow{P_{best}^{(t)}} \right\|_2}{\left\| \overrightarrow{P_{worst}^{(t)}} - \overrightarrow{P_{best}^{(t)}} \right\|_2} \tag{9}$$

Where $D_{SA_i}(t)$ is the relative distance of each search agent $SA_i(t)$, and $\overrightarrow{P_{best}^{(t)}}$ and $\overrightarrow{P_{worst}^{(t)}}$ are the best and the worst position found at $(t)$th iteration, respectively. The symbol $\|\|_2$ is the Euclidean distance between two vectors.

For convenience in calculation, $D_{SA_i}(t)$ will be converted to a new value $CL_{SA_i}(t)$ called "*characteristic length*" as shown in Eq. (10).

$$CL_{SA_i}(t) = e^{-D_i(t)} \tag{10}$$

**Algorithm 2: The process of centroid vector positions updating in KO over the course of iterations**

***For*** $t = 1: T_{\max}$

*Calculate the objective function of each individual position* $\overrightarrow{P_i}\ (i = 1, 2, ..., N)$

$f_{obj}\left(\overrightarrow{P_i}\right),\ \overrightarrow{P_i} = \left\{p_{i,1}, p_{i,2}, ..., p_{i,D}\right\} \in \mathbb{R}^{1\times D}$

*Calculate the reduction population size at each iteration following Eq.* (6)

*Arrange the Fitness of reduction population size at each iteration in ascending order according to* Eq. (7)

*Apply K-means with number of clusters K = 3 for finding the centroid vector positions following Eq.* (8)

***End***

**Algorithm 3: The strategy of self-improvement of initial population in KO over the course of iterations**

***For*** $t = 1: T_{\max}$

***For*** *each search agent* $\overrightarrow{SA_i^{(t)}} = \left\{sa_{i1}, sa_{i2}, ..., sa_{iD}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$

*Update the new positions* $\overrightarrow{SA_i^{t+1}}$ *following the movement strategy of KO;*

*Calculate the objective function* $f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right)$;

*% Greedy selection to update* $\overrightarrow{P_i} = \left\{p_{i1}, p_{i2}, ..., p_{iD}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$

***IF*** $f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right) < f_{obj}\left(\overrightarrow{P_i}\right)$;

$\overrightarrow{P_i} = \left(\overrightarrow{SA_i^{(t+1)}}\right);\ f_{obj}\left(\overrightarrow{P_i}\right) = f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right)$;

***End***

***End***

***End***

Eq. (10) shows that the value of $CL_{SA_i}$ is always from zero to one. Based on this value, we can depict the position of each search agent $\overrightarrow{SA_i^{(t)}}$ in comparison with the best solution $\overrightarrow{P_{best}^{(t)}}$ and the worst solution $\overrightarrow{P_{worst}^{(t)}}$. If $\overrightarrow{SA_i^{(t)}}$ is far away from $\overrightarrow{P_{best}^{(t)}}$, $CL_{SA_i}$ is approximately equal to zero and if $\overrightarrow{SA_i^{(t)}}$ is close to $\overrightarrow{P_{best}^{(t)}}$, $CL_{SA_i}$ is approximately equal to one. In some cases, $\overrightarrow{SA_i^{(t)}}$ will move quickly to $\overrightarrow{P_{best}^{(t)}}$ after the first few iterations, and this leads to difficulties in the orientation of movement at the next iteration of $\overrightarrow{SA_i^{(t)}}$. To overcome this limitation, KO proposes an equation to adjust the value of $CL_{SA_i}$ as given in Eq. (11).

$$PDP_{SA_i} = CL_{SA_i} \sin\left(\frac{t\pi}{T_{\max}}\right) \tag{11}$$

Based on Eq. (11), $PDP_{SA_i}$ will a new term named "*position density probability (PDP)*". This term will decide on the movement strategy in the next iteration $(t + 1)$th as shown in **Algorithm 4**.

**Algorithm 4: The selection for movement strategy of each search agent**

***For*** *each search agent* $\overrightarrow{SA_i^{(t)}} = \left\{sa_{i1}, sa_{i2}, ..., sa_{iD}\right\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$

***IF*** $PDP_{SA_i} = CL_{SA_i} \sin\left(\frac{t\pi}{T_{\max}}\right) > threshold$ ;

*Move following the ability of exploitation*

***Else***

*Move following the ability of exploration*

***End***

***End***

Where *threshold* is a turning parameter, which can be selected based on the particular optimization problem. In this work, we propose *threshold* $= 0.5$ for a balance between the ability of exploitation and exploration. Based on Algorithm 3, only the search agents, whose position is close to $\overrightarrow{P_{best}^{(t)}}$ with $PDP_{SA_i} > 0.5$, will move according to the ability of exploitation. In contrast, the remaining search agents, whose position are located in the poor search spaces, will move according to the ability of exploration to find other, hopefully better search spaces. In the remaining sections, the ability of exploration and exploitation in KO will be described in details.

### 4.3. The strategy for exploitation

To diversify the search spaces in KO, besides three centroid vector positions exploited at each iteration, KO adds a new search region called the mean centroid vector position $\overrightarrow{C_{mean}^{(t)}}$ as given in Eq. (12).

$$\overrightarrow{C_{mean}^{(t)}} = \frac{\overrightarrow{C_1^{(t)}} + \overrightarrow{C_2^{(t)}} + \overrightarrow{C_3^{(t)}}}{3} \tag{12}$$

The strategy of exploitation is secured through the process of simulation of how the search agent population moves forward to the search spaces having the high probability of exploiting the new global best values. In KO, a high-quality search space is defined by a set of vectors $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}, \overrightarrow{P_{best}^{(t)}}\right\}$. Based on this space, KO proposes three trends of movement to achieve efficient exploitation of search regions, which are recorded to close search spaces $\Omega^*$.

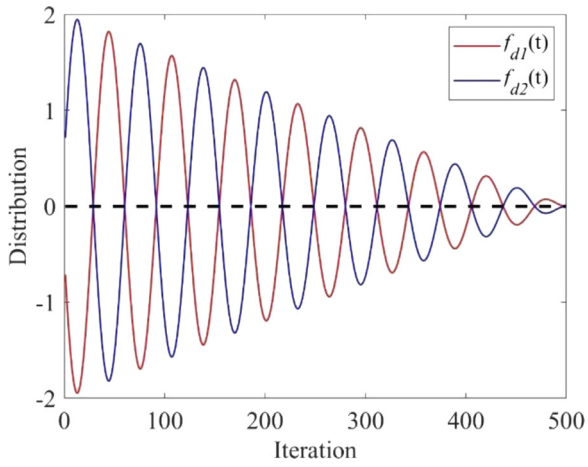


**Fig. 3.** The distribution of $f_{d_1}(t)$ and $f_{d_2}(t)$ as a function of iterations.

#### 4.3.1. The first trend for exploitation

The first trend focuses on exploiting the search spaces established between the current position of each search agent $\overrightarrow{SA_i^{(t)}}$ and $\overrightarrow{P_{best}^{(t)}}$, $\overrightarrow{C_{mean}^{(t)}}$ as shown in Eq. (13).

$$\overrightarrow{SA_i^{(t+1)}} = w_{11}\overrightarrow{P_{best}^{(t)}} + d_1(t)\left|\overrightarrow{SA_i^{(t)}} - w_{12}\overrightarrow{P_{best}^{(t)}}\right| + d_2(t)\left|\overrightarrow{SA_i^{(t)}} - w_{13}\overrightarrow{C_{mean}^{(t)}}\right| \tag{13}$$

Where $w_{11}$, $w_{12}$, $w_{13}$ are randomly distributed values according to stochastic algorithm. They can be either a scalar number or a scalar vector. In KO, $w_{11}$ is selected as a scalar number having value in the range [0, 2]. This parameter specifies the effect of the current position on the next position. $w_{12}$ and $w_{13}$ are selected as scalar vectors having value in range [0, 1]. These parameters are to expand the search spaces around $\overrightarrow{P_{best}^{(t)}}$ and $\overrightarrow{C_{mean}^{(t)}}$.

$d_1(t)$ and $d_2(t)$ are two scalar parameters called "*drift control*", which are proposed with the purpose of providing the new high-quality search spaces. To shrink these search spaces for increasing the accuracy level and the convergence rate, $d_1(t)$ and $d_2(t)$ will be designed to reduce as the iterations increase. For this purpose, KO proposes two symmetric functions $f_{d_1}(t)$ and $f_{d_2}(t)$ having values in the range $[-2, 2]$ as shown in Eq. (14), and their distribution as a function of iterations is illustrated in Fig. 3.

$$\begin{aligned} f_{d1}(t) &= 2\sin\left[0.1T_{\max}\left(1-\frac{t}{T_{\max}}\right)\right]\left(1-\frac{t}{T_{\max}}\right) \\ f_{d2}(t) &= -2\sin\left[0.1T_{\max}\left(1-\frac{t}{T_{\max}}\right)\right]\left(1-\frac{t}{T_{\max}}\right) \end{aligned} \tag{14}$$

Based on $f_{d_1}(t)$ and $f_{d_2}(t)$, the "*drift control*" $d_1(t)$ and $d_2(t)$ parameters are calculated by Eqs. (15) and (16)

$$d_1(t) = (r_1 f_{d1}(t) + r_2 f_{d2}(t)) \tag{15}$$

$$d_2(t) = (r_3 f_{d1}(t) + r_4 f_{d2}(t)) \tag{16}$$

Where $r_1$, $r_2$, $r_3$ and $r_4$ are random scalar numbers having values in the range [0, 1], Based on the random distribution of these parameters and two symmetric functions $f_{d_1}(t)$, $f_{d_2}(t)$, the values of $d_1(t)$ and $d_2(t)$ can be registered either negative or positive at any iteration as shown in Fig. 4.

KO uses the absolute value $|distance|$ as given in Eq. (13) instead of *distance* as almost all algorithms, in which *distance* is defined as a reduction distance vector to ensure a trend for moving toward the potential search area. A new concept is proposed by the parameters $d_1(t)$ and $d_2(t)$ having values in the range $[-2, 2]$. Depending on the distribution of $d_1(t)$ and $d_2(t)$, four cases of movement are established as shown in Fig. 5.

Besides the exploitation of high-quality search spaces bounded between $\overrightarrow{P_{best}^{(t)}}$, $\overrightarrow{C_{mean}^{(t)}}$ and the position of each search agent $\overrightarrow{SA_i^{(t)}}$, this trend can also gain the expanding search spaces, which are controlled by two proposed parameters $d_1(t)$ and $d_2(t)$. Thus, during the first few iterations, the search spaces will be generated diversely to increase the chances of escaping the local optimal problem and searching for better search space. And during the last few iterations, because of the shrinking of the search spaces, the accuracy level in KO will be improved.

#### 4.3.2. The second trend for exploitation

The second trend focuses on exploiting the search spaces established between the current search agent $\overrightarrow{SA_i^{(t)}}$ and the high-quality search spaces symbolized $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}, \overrightarrow{P_{best}^{(t)}}\right\}$. For this purpose, the position updating of each search agent $\overrightarrow{SA_i^{(t)}}$ is given in Eq. (17).

$$\overrightarrow{SA_i^{(t+1)}} = w_{21}\overrightarrow{SA_i^{(t)}} + w_{22}\left(\overrightarrow{P_{best}^{(t)}} - w_{23}\overrightarrow{SA_i^{(t)}}\right) + \sum_{k=1}^{3}\delta_k\left(\overrightarrow{C_k^{(t)}} - \beta_k\overrightarrow{SA_i^{(t)}}\right) \tag{17}$$

Where $w_{21}$ is a scalar number with a value in the same range of $w_{11}$ in the first trend. The parameters $w_{22}$, $w_{23}$, $\delta_k$, and $\beta_k$ are four vectors having the value in the range [0, 1]. By adding the position of three centroid vectors $\overrightarrow{C_k^{(t)}}$ ($k = 1, 2, 3$), the search spaces using Eq. (17) will become more flexible to move the high-quality search spaces $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}, \overrightarrow{P_{best}^{(t)}}\right\}$. Together with the process of self-improvement of the initial population based on Algorithm 1, the centroid vectors will be gradually shrunk to close $\overrightarrow{P_{best}^{(t)}}$. Thus, the search space of each search agent $\overrightarrow{SA_i^{(t)}}$ can reach wide regions during the first iterations, and as the number of iterations increases, these regions will also gradually shrink together with the position of centroid vectors. Based on this advantage, the new location of each search agent $\overrightarrow{SA_i}$ is always ensured that they are close to the search space around $\overrightarrow{P_{best}^{(t)}}$ and $\overrightarrow{C_k^{(t)}}$ ($k = 1, 2, 3$). The process of position updating of each search agent $\overrightarrow{SA_i}$ in the second trend is illustrated in Figs. 6 and 7.

#### 4.3.3. The third trend for exploitation

The third trend focuses on exploiting the search spaces established between the mean centroid vector position $\overrightarrow{C_{mean}^{(t)}}$ and the best solution $\overrightarrow{P_{best}^{(t)}}$. In this trend, the search spaces are deeply exploited in comparison with the first two trends. Thus, the position of search agent $\overrightarrow{SA_i^{(t)}}$ is ignored. Only the position of $\overrightarrow{C_{mean}^{(t)}}$ and $\overrightarrow{P_{best}^{(t)}}$ are considered to establish the new search spaces as shown in Eq. (18)

$$\overrightarrow{SA_i^{(t+1)}} = w_{31}\overrightarrow{C_{mean}^{(t)}} + w_{32}\left(\overrightarrow{P_{best}^{(t)}} - w_{33}\overrightarrow{C_{mean}^{(t)}}\right) \tag{18}$$

Where $w_{31}$, $w_{32}$, $w_{33}$ are the same as $w_{11}$, $w_{12}$, $w_{13}$ in the first trend. It can be seen that the search spaces in this strategy will be narrower than that of the first two trends. The movement in this trend is more bias to increase the convergence rate and

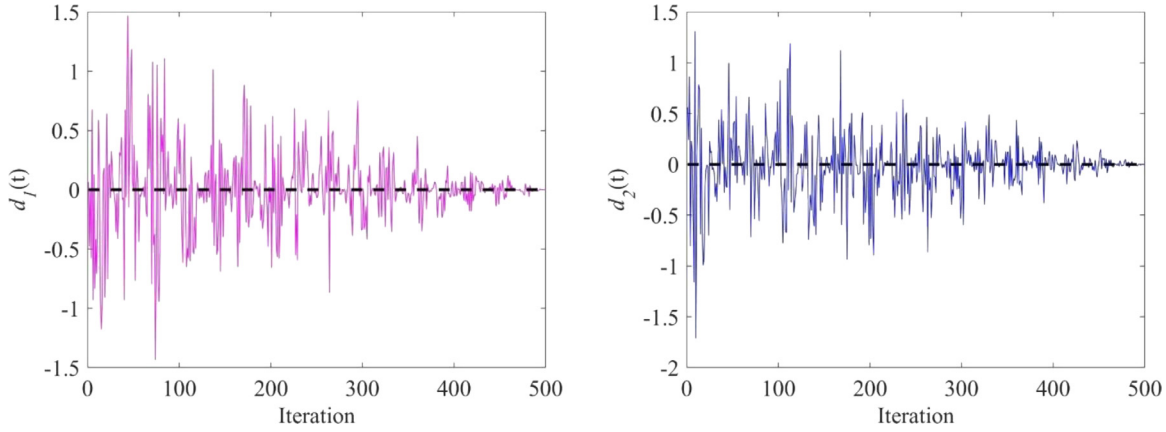


**Fig. 4.** The distribution of $d_1(t)$ and $d_2(t)$ as function of iterations.

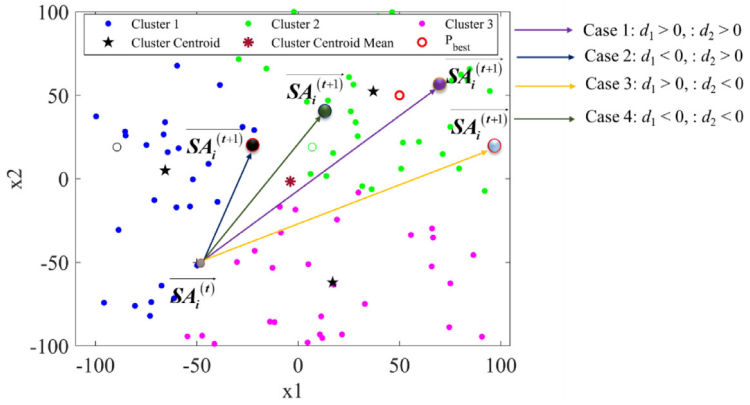


**Fig. 5.** Four movement cases of the first trend of exploitation.

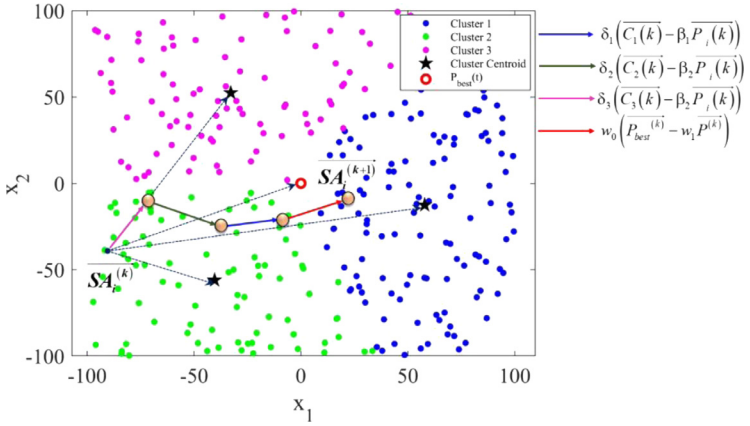


**Fig. 6.** The process of position updating of search agent $\overrightarrow{SA_i^{(t)}}$ for $t$th iteration to $(t+1)$th iteration during the first few iterations.

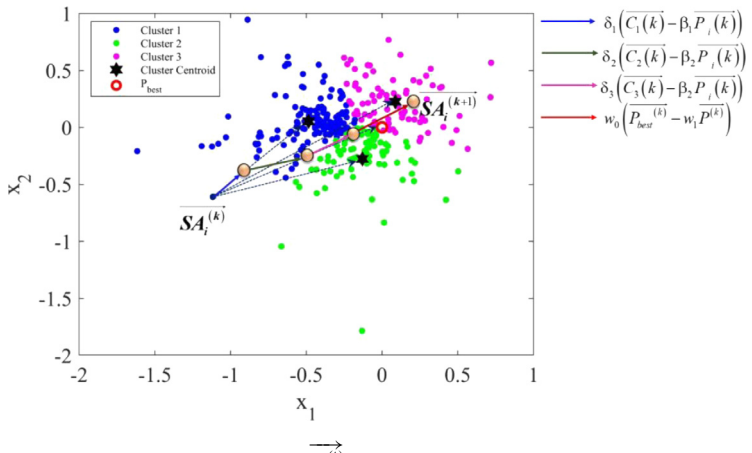


**Fig. 7.** The process of position updating of search agent $\overrightarrow{SA_i^{(t)}}$ for $t$th iteration to $(t+1)$th iteration during the last few iterations.

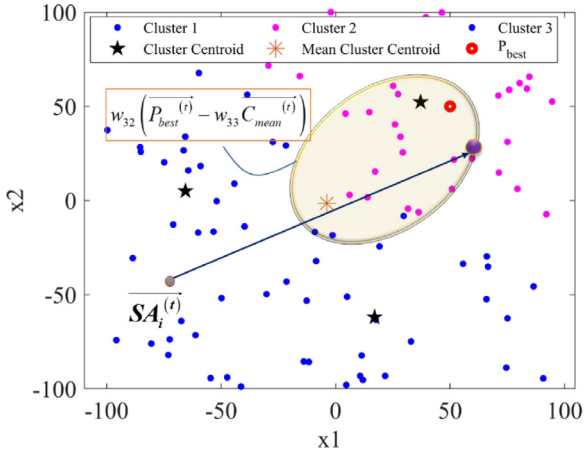


**Fig. 8.** The process of position updating of each search agent following the third trend.

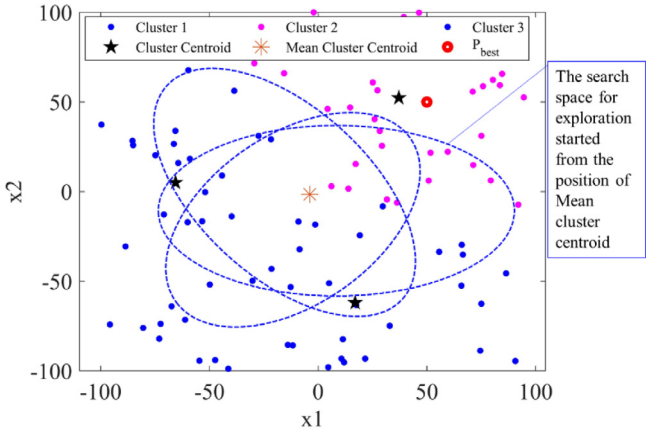


**Fig. 9.** The new search spaces rounded the position of centroid mean vector.

the accuracy level than the ability of escaping local optima. The process of position updating for the third trend is shown in Fig. 8.

### 4.4. The movement strategy for exploration

The algorithm's effectiveness depends on its ability to find a balance between exploitation and exploration. In KO, the exploration strategy movement is based on two trends, each of which is based on a random selection process with a probability of equilibrium.

### 4.5. The first trend of exploration

The first trend for exploration uses the advantages of exploiting the position of centroid vectors $\overrightarrow{C_k^{(t)}}$ $(k = 1, 2, 3)$ through an intermediary $\overrightarrow{C_{mean}^{(t)}}$ given in Eq. (12). Thus, the large search spaces rounded the position of $\overrightarrow{C_{mean}^{(t)}}$ will be generated to find the new search spaces as shown in Fig. 9. By this way, only the local neighborhood is visited if the movement distance is short enough, and if the movement distance is long enough, the region visited can be too far away from the region of high-quality, which will increase diversification significantly. As mentioned in the previous section, KO uses a sort of linear reduction initial population to ensure that the centroid vectors $\overrightarrow{C_k^{(t)}}$ $(k = 1, 2, 3)$ generated are close to $\overrightarrow{P_{best}^{(t)}}$ as the iteration increases. Therefore, the search spaces explored around $\overrightarrow{C_{mean}^{(t)}}$ are diverse during the first few iterations, for improving the ability of exploration.

The new position of each search agent $\overrightarrow{SA_i^{(t)}}$ must be started at the position of centroid mean $\overrightarrow{C_{mean}^{(t)}}$. At the same time, the distance of the movement must be randomly long or short at any iteration. For this purpose, KO uses the concept of Lévy flight, which is a type of random walk whose step length obeyed Lévy distribution. Mantegna proposed a fast algorithm with high accuracy for generating a step length whose value depends on the index distribution $\beta$ ranged between 1 and 2. This algorithm is employed in KO to generate the random step length as given in Eq. (19).

$$S = \frac{U}{|V|^{1/\beta}} \tag{19}$$

Where $U$ and $V$ are two normal distribution variables with standard deviations of $\sigma_u$ and $\sigma_v$ as given in Eq. (20).

$$\begin{aligned} U &= normal\left(0, \sigma_u^2\right) \\ V &= normal\left(0, \sigma_v^2\right) \end{aligned} \tag{20}$$

Where $\sigma_u$ and $\sigma_v$ are standard deviation as given in Eq. (21).

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma\left[(1+\beta)/2\right]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \sigma_v = 1, 1 \le \beta \le 2 \tag{21}$$

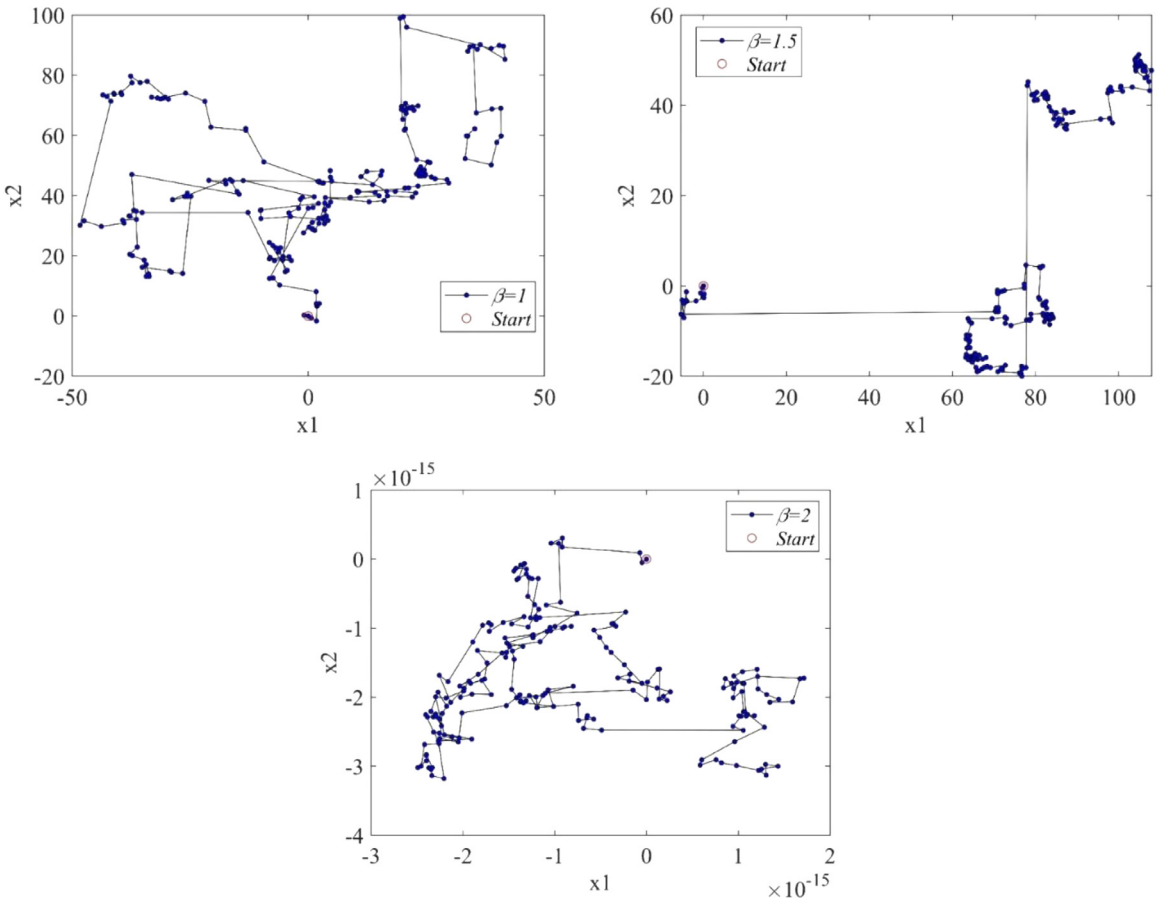


**Fig. 10.** The distribution of step length $S$ in two-dimensional space with different $\beta$ parameters.

In Eq. (20), the Gamma function $\Gamma$ for an integer $z$ is expressed as in Eq. (22)

$$\Gamma(z) = \int_0^\infty t^{z-1} e^t dt \tag{22}$$

The step length $S$ depends on the distribution of the parameter $\beta$ as shown in Eqs. (19) and (21). If $\beta$ is a small scalar number, the step length $S$ will reach a long movement, and vice versa, If $\beta$ is a large number scalar, the step length $S$ will reach a short movement as shown in Fig. 10. It can be seen that with the gradual increase of the parameter $\beta$, the distribution of step length $S$ tends to reduce gradually. The range distribution of $[x_1, x_2]$ is $[-50, 100]$ with $\beta = 1$, which will be narrowed as the parameter $\beta$ increases from one to two, especially, if $\beta$ reaches the maximum value, i.e. $\beta = 2$. The step length $S$ will reach an infinitesimal value, for which the distribution of $[x_1, x_2]$ can reach $10^{-15}$. Based on this feature, it is a robust method for the exploration of a large search space.

Based on the analysis of the influence of the parameter $\beta$ on the step length $S$, the goal of expanding the search spaces with random steps, long enough or short enough, and more random movements are achieved. In this work, the parameter $\beta$ is calculated as given in Eq. (23) and the proposed distribution of step length with $\beta$ is shown in Fig. 11.

$$\beta = 1 + rand\,(0, 1) \tag{23}$$

Based on a controlled adjustment of step length $S$, the first trend for exploration is given in Eq. (24) and the new search spaces expanded from the centroid mean vector is shown in Fig. 12.

$$\overrightarrow{SA_i^{(t+1)}} = \overrightarrow{C_{mean}^{(t)}} + step\ length\ (S) \quad IF\ rand\ (0, 1) > 0.5 \tag{24}$$

### 4.6. The second trend of exploration

In some cases, the global optima will be located in a region, which is too far away from the position of $\overrightarrow{C_{mean}^{(t)}}$. And if the effort of the first trend following Eq. (24) is unable to reach these spaces, KO will face a risk of local optimal problem. To overcome this limitation, the search spaces is adjusted as shown in Eq. (25).

$$\overrightarrow{SA_i^{(t+1)}} = L_b + rand\,(0, 1)\,(U_b - L_b) \quad IF\ rand\,(0, 1) < 0.5 \tag{25}$$

Where $L_b$ and $U_b$ are the lower-bound and upper-bound vectors, respectively. The new search spaces increase the opportunity of exploring a better high-quality ones. And in this way, the new position of each search agent is generated to replace the old ones. Then, based on the high-quality search spaces $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}, \overrightarrow{P_{best}^{(t)}}\right\}$ exploited at each iteration, these new positions will move to region $\Omega^*$ to find other hopefully better regions. This process will be performed continuously through the over course of iterations. It leads to the appreciative evaluation of KO compared to the other optimization algorithms.

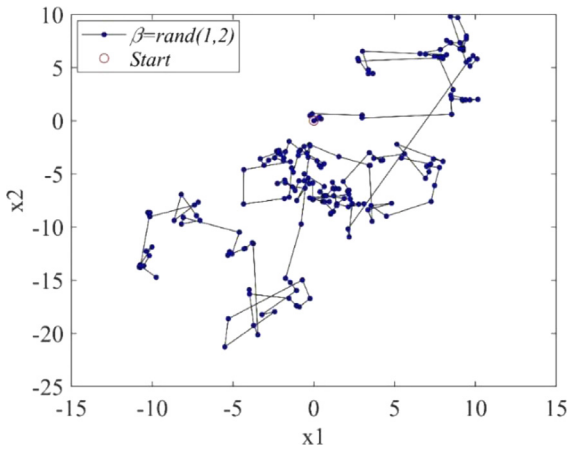


**Fig. 11.** The distribution of step length S in two-dimensional space with $\beta = 1 + rand\,(0, 1)$.

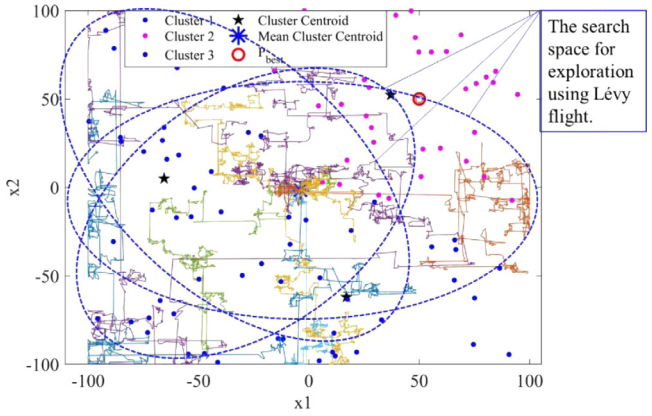


**Fig. 12.** The new search space expanded from centroid mean vector using Lévy flight.

### 4.7. Schematic representation of K-means clustering optimizer (KO)

According to the descriptions of KO in the previous sections, some primary characteristics can be summarized to show how KO can be effective for solving optimization problems:

- KO uses K-means clustering algorithm to generate as many diverse results as possible in the feasible search spaces. This is a relatively new concept in comparison with the other optimization algorithms. Thus, KO establishes the new high-quality search space defined as $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}\right\}$. Especially, these spaces will be gradually narrowed as the number of iterations increases. This can be achieved by a technique named the Linear Population Size Reduction (LPSR) method. Based on these advantages, the movement strategy in KO will have more access to the potential search spaces compared to the other algorithms.
- It is not like the other algorithms, the search process will focus on the self-improvement of each agent search, this is sometimes will cause local optimization problems. In KO, the search process is secured through the self-improvement of an initial population $\left(\overrightarrow{P_i}\right)$, $i = 1, 2, \ldots, N$ according to Algorithm 3. By this way, the adjustment direction of the movement of each search agent becomes more flexible because the information gained from each search agent $\overrightarrow{SA_i^{(t)}}$ at any iteration is stored by its compatible individual position $\left(\overrightarrow{P_i}\right)$ in the initial population.
- In KO, the decision of direction movement at the $(t + 1)$th iteration of each search agent $\overrightarrow{SA_i^{(t)}}$ is based on a new term named "*position density probability (PDP)*", which is calculated at the $(t)$th iteration. In this way, only the search agents having a high chance of finding a better search spaces will move bias to the ability of exploitation. In contrast, search agents located in regions, which are too far away from high-quality spaces will be biased for the ability of exploration.
- Based on the diversity of the high-quality search spaces $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}\right\}$, KO proposed three trends for exploitation to take full advantage of these spaces. Each trend has its own unique movement strategy to achieve its own high-quality search space. Among them, the first trend and the second trend focus on exploiting the large search spaces established between the positions of each search agent and high-quality search spaces $\Omega^*$. Meanwhile, the third trend focuses on deepening the search spaces created by the position of $\overrightarrow{C_{mean}^{(t)}}$ and $\overrightarrow{P_{best}^{(t)}}$. Based on these trends, it can be seen that all spaces, where the opportunity to find better solutions is recognized, are considered. This is an advantage, that makes robust enhancements to the convergence rate and accuracy level in KO.
- The ability of exploration in KO is secured by two trends. The first trend is to take full advantage of the spaces found by the mean centroid vector $\overrightarrow{C_{mean}^{(t)}}$. A modification of step length $S$ is proposed to create various random movements, which can reach either enough lengths or enough shorts. The proposed step length $S$ in association with $\overrightarrow{C_{mean}^{(t)}}$ is the foundation of the first trend. However, this trend will face the risk of a local optimum problem if the position of $\overrightarrow{C_{mean}^{(t)}}$ is too far away from the global optimal. Thus, the second trend is proposed to overcome this problem. This trend is to reset the search spaces of each search agent $\overrightarrow{SA_i^{(t)}}$. The selection trend for exploration is implemented by a random selection mechanism with equilibrium probability.
- Besides the advantages mentioned above, in some cases, if the iterations of KO are not sufficient enough, the ability of exploitation in KO can face challenge. However, these are also considered as general limitations of current algorithms. The results obtained in this paper indicate that with the total iterations $T_{\max} = 1000$, KO can achieve the necessary convergence rate and the best values with acceptable error.

The number of function evaluations (*NFE*) is a significant way to assess the effectiveness of any optimization algorithm. Similar to other algorithms, KO will have to calculate *NFE* according to the equation $NFE = N \times T_{max}$, where $N$ is the number of initial particles, and $T_{\max}$ is the maximum number of iterations. For this purpose, at any iteration, only unique movement trends for exploitation and exploration is selected. Thus, based on the combination of Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm 4 the pseudo code of KO is given in Algorithm 5.

## 5. Numerical experiments

### 5.1. Classical benchmark functions

To evaluate the effectiveness and performance of KO in different functions, in this section, 23 classical benchmark functions

**Algorithm 5: The implementation process of KO to find the best solution**

***%% Start----------***

*Select the initial population size* $(N)$*, the maximum iterations* $(T_{max})$*;*

*Generate the initial population* $\overrightarrow{P_i}\ (i=1,2,...,N)$*;*

*Generate the initial search agent population* $\overrightarrow{SA_i}\ (i=1,2,...,N)$*;*

*Find the best solution* $\overrightarrow{P_{best}}$ *and the best objective function* $f\left(\overrightarrow{P_{best}}\right)$*;*

*Find the worst solution* $\overrightarrow{P_{worst}}$ *and the worst objective function* $f\left(\overrightarrow{P_{worst}}\right)$*;*

*Find the position of the initial centroid vectors* $\left\{\overrightarrow{C_1},\overrightarrow{C_2},\overrightarrow{C_3}\right\}$ *using Eq. (8) and the position of initial mean centroid vector* $\overrightarrow{C_{mean}}$ *using Eq. (12);*

*%%* ***Main loop ----------***

***For*** *t =1: Tmax*

***For*** *each search agent* $\overrightarrow{SA_i^{(t+1)}}$

*Trend = randi (3); % select random the first trend, the second trend, the third trend*

*Calculate the position density probability* $PDP_{SA_i}$ *of each search agent* $\overrightarrow{SA_i^{(t+1)}}$ using Eq. (*11*)

***If*** $PDP_{SA_i} > threshold$ *% Move following the ability of exploitation*

***Switch Trend***

***Case 1*** *% the first trend for exploitation*

*Update the position of each search agent* $\overrightarrow{SA_i^{(t+1)}}$ *using Eq.(13);*

***Case 2*** *% the second trend for exploitation*

*Update the position of each search agent* $\overrightarrow{SA_i^{(t+1)}}$ *using Eq.(14);*

***Case 3*** *% the third trend for exploitation*

*Update the position of each search agent* $\overrightarrow{SA_i^{(t+1)}}$ *using Eq.(18);*

***End switch***

*Calculate the objective function of each search agent according to the ability of exploitation work* $f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right)$*;*

***If*** $f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right) < f_{obj}\left(\overrightarrow{P_i}\right)$

$\overrightarrow{P_i}=\left(\overrightarrow{SA_i^{(t+1)}}\right);\ f_{obj}\left(\overrightarrow{P_i}\right)=f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right)$

*Update* $\overrightarrow{P_{best}^{(t+1)}}$ *and the best objective function* $f\left(\overrightarrow{P_{best}^{(t+1)}}\right)$*;*

*Update* $\overrightarrow{P_{worst}^{(t+1)}}$ *and the best objective function* $f\left(\overrightarrow{P_{worst}^{(t+1)}}\right)$*;*

***End***

***Else (*** $PDP_{SA_i} > threshold$ ) *% Move following the ability of exploration*

***If*** *rand(0,1) > 0.5 % the first trend for exploration*

*Update the position of each search agent* $\overrightarrow{SA_i^{(t+1)}}$ *using Eq.(24);*

***Else*** *% the second trend for exploration*

*Update the position of each search agent* $\overrightarrow{SA_i^{(t+1)}}$ *using Eq.(25);*

***End***

*Calculate the objective function of each search agent according to the exploration* $f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right)$

***If*** $f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right) < f_{obj}\left(\overrightarrow{P_i}\right)$ *% apply Algorithm 2*

$\overrightarrow{P_i}=\left(\overrightarrow{SA_i^{(t+1)}}\right);\ f_{obj}\left(\overrightarrow{P_i}\right)=f_{obj}\left(\overrightarrow{SA_i^{(t+1)}}\right)$

*Update* $\overrightarrow{P_{best}^{(t+1)}}$ *and the best objective function* $f\left(\overrightarrow{P_{best}^{(t+1)}}\right)$*;*

*Update* $\overrightarrow{P_{worst}^{(t+1)}}$ *and the best objective function* $f\left(\overrightarrow{P_{worst}^{(t+1)}}\right)$*;*

***End***

***End***

***End For***

***%% Update the centroid vector positions for the next iteration----------***

*Calculate the reduction population size at each iteration following Eq. (6);*

*Arrange the Fitness of reduction population size at each iteration in ascending order following* Eq. (*7*)

*Apply K-means with number of clustering K = 3 for finding the centroid vector positions* $\left\{\overrightarrow{C1^{(t)+1}},\overrightarrow{C2^{(t+1)}},\overrightarrow{C3^{(t+1)}},\right\}$ *following Eq. (8);*

*Out put the best result at each iteration;*

***End For***

(F1–F23), which were released in previous studies [32,41–43], are used to check the ability of exploration and exploitation of KO. These functions include unimodal (F1–F7) having one global optimum with no local optima. They are used to check the ability of the exploitation of KO. Meanwhile, these functions (F8–F13) are multimodal functions having changeable dimensions. These functions are used to test the exploration performance of KO. And the remaining functions (F14–F23) are fixed dimensions, which show the ability of exploration in the low dimension of KO. Besides the 23 classical benchmark functions, hybrid composition functions including six functions in CEC2005 (CF15–CF20), are also considered. These functions are high-complex and have many local optima as in real engineering problems. As a result, these functions are mentioned in order to thoroughly evaluate KO's performance in escaping local optima. In this test, the dimension of whole functions is selected to be $D = 30$, except for fixed-dimension functions (F14–F23). The formulation and properties of 29 benchmark functions are given in Table 1.

### 5.2. Sensitivity analysis of the population size in KO

In almost all optimization algorithms, the search process will focus on the self-improvement of each search agent. This sometimes causes local optimization problems. In KO, the search process is secured through the self-improvement of an initial population. And the search agents in KO will tend to fluctuate for exploring the new search spaces. Thus, how to select the number of search agents is an important factor to cover a sufficiently large search space. If the search space created by search agents are not wide enough, KO will face problems of local optimal problems. For measuring the impact of the population size to the KO's performance. For this purpose, the population size $N = \{20, 30, 40, 50, 60, 70, 80, 100\}$ with the number of evaluation functions $(NEF) = 10{,}000$ is used to find the best solution of the first 23 classical benchmark functions with dimension $(D = 10)$. Based on the statistic results shown in Table 2, it can be seen that, in most cases, KO can find the values with acceptable error. Based on the Friedman mean ranking test, the best performance is found for $N = 30$. This can be considered as a good reference result to select the population size for solving the optimization problems. Based on this sensitivity analysis, the population size in KO is selected $N = 30$ for the remaining parts of this section.

### 5.3. The performance of KO

Before measuring the effectiveness of KO in comparison with the other algorithms. A detailed analysis of the process of finding the best solution for the basic functions is presented. For this purpose, F1, F4, F8, F10, F11, F14 and F23 functions are selected to assess the performance of KO for the ability of exploitation and exploration. For this test, the size of the initial population and the number of search agents are selected as $N = 30$. The maximum number of iterations is $T_{max} = 100$. Six qualitative metrics, including the search spaces for exploitation, the search spaces for exploration, the process of self-improvement of the initial population, the trajectory, and the convergence rate, are plotted in Fig. 13. The first figure column at the first row depicts the shape of function in two-dimensional view to give insight about the domain's topology.

The ability of exploitation in KO is described fully and in detail in all 3 trends as shown in the second figure column at the first row in Fig. 13. It can be seen that the search spaces for exploitation is not only distributed around the high-quality search spaces with dense density, but also scattered throughout the whole solution spaces with lower frequency. Because of the reduction of the initial population size for establishing the centroid vectors, based on this feature, the search spaces for exploitation will be gradually narrow and achieve a high density bounded by the best solution position exploited during the last iterations for the accuracy improvement. The first trend and the second trend focus on the exploitation of the search spaces located between the high-quality search spaces $\Omega^* = \left\{\overrightarrow{C_1^{(t)}}, \overrightarrow{C_2^{(t)}}, \overrightarrow{C_3^{(t)}}, \overrightarrow{C_{mean}^{(t)}}\right\}$ and the current position search agent $\overrightarrow{SA_i^{(t)}}$, so these regions show a wider distribution in comparison with the third trend. This wide distribution increases the opportunity for exploiting new potential search spaces. This is a robust advantage in KO for solving problems having many local optima as shown in (F8, F10, F11, F23) functions. Meanwhile, the third trend establishes a limited search spaces, which is only located between the mean centroid vector $\overrightarrow{C_{mean}^{(t)}}$ and the best solution $\overrightarrow{P_{best}^{(t)}}$. For the functions having one global optimum (F1, F4). This trend will enhance the convergence rate and reach an acceptable accuracy level during the first few iterations. The mutual support of these three trends results is an effective KO movement strategy for achieving both an increased convergence rate and a local optimum.

The ability of exploration is illustrated in the third column in the first row of each figure. Because information about good search spaces gained by each search agent $\overrightarrow{SA_i^{(t)}}$ is already memorized by its respective initial population $\left(\overrightarrow{P_i}\right)$. So each search agent do not need to improve itself, it can abandon the search spaces visited, whether these search spaces are recognized as poor regions. This is an outstanding advantage of the KO for solving problems with many local optima compared to the other algorithms.

The advantage of step length $S$ is effectively exploited to establish a new search spaces around the position of mean centroid vector $\overrightarrow{C_{mean}^{(t)}}$. Especially, these spaces are not too large to avoid conflict with the search space established by the second trend in the movement strategy for exploitation, as shown clearly in Fig. 13. And the remaining task is to reset the current search spaces to new search spaces to continue for exploration. Thus, the search spaces in KO can be expanded throughout the solution space. These new spaces will continue to be refined in the next iterations, hopefully leading to better opportunities.

The process of self-improvement of the initial population is shown in the first column at the second row. This strategy will be more effective than the self-improvement of each search agent in almost all the other algorithms.

The trajectory and convergence curve are shown in the last two columns at the second row. They are used to evaluate the convergence performance of KO. The convergence curves of F1, F4 functions, which have one global optimum, are very smooth and drop rapidly. They reach a stable value only after the first few iterations. This proves the effectiveness of the movement strategy for exploitation in the KO. In the case of the remaining functions, whose curves are rougher, the skill of exploration is more biased than the skill of exploitation. Finally, the convergence curves can all accurately approximate the global optimum in the final iterations.

### 5.4. Comparison between KO and other optimization algorithms for 29 benchmark functions

In this section, seven well-known optimization algorithms are employed to compare with KO, namely Particle Swarm Optimization (PSO) [2], Grey Wolf Optimizer (GWO) [5], Whale Optimization algorithm WOA [32], Slime Mould Algorithm (SMA) [42], Arithmetic Optimization Algorithm (AOA) [41], Reptile Search

**Table 1**
Mathematical formulation and properties of 29 benchmark functions.

| **23 classical benchmark functions with Dimension $D$ = 30 for functions (F1–F13)** | Solution search space (S) | $f_{\min}$ |
|---|---|---|
| $F_1(x) = \sum_{i=1}^{n} x_i^2$ | $[-100, 100]^D$ | 0 |
| $F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$ | $[-100, 100]^D$ | 0 |
| $F_3(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2$ | $[-100, 100]^D$ | 0 |
| $F_4(x) = \max \{ \lvert x_i \rvert , 1 \le i \le n \}$ | $[-100, 100]^D$ | 0 |
| $F_5(x) = \sum_{i=1}^{n-1} \left[ 100 \left( x_{i+1} - x_i^2 \right) + (x_i - 1)^2 \right]$ | $[-30, 30]^D$ | 0 |
| $F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$ | $[-100, 100]^D$ | 0 |
| $F_7(x) = \sum_{i=1}^{n} i x_i^4 + rand(0, 1)$ | $[-1.28, 1.28]^D$ | 0 |
| $F_8(x) = \sum_{i=1}^{n} -x_i \sin \left( \sqrt{\lvert x_i \rvert} \right)$ | $[-500, 500]^D$ | $-418.9829 \times D$ |
| $F_9(x) = \sum_{i=1}^{n} \left[ x_i^2 - 10 \cos (2\pi x_i) + 10 \right]$ | $[-5.15, 5.12]^D$ | 0 |
| $F_{10}(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos (2\pi x_i) \right) + 20 + e$ | $[-32, 32]^D$ | 0 |
| $F_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1$ | $[-500, 500]^D$ | 0 |
| $F_{12}(x) = \frac{\pi}{x} \left\{ 10 \sin (\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[ 1 + 10 \sin^2 (\pi y_{i+1}) \right] + (y_n - 1)^2 + \sum_{i=1}^{n} u (x_i, 10, 100, 4) \right\}$<br>$y_i = 1 + \frac{x_i + 1}{4} u (x_i, a, k, m) = \begin{cases} k (x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k (-x_i - a)^m & x_i < -a \end{cases}$ | $[-50, 50]^D$ | 0 |
| $F_{13}(x) = 0.1 \left\{ \sin^2 (3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 \left[ 1 + \sin^2 (3\pi x_i + 1) \right] + (x_n - 1)^2 \left[ 1 + \sin^2 (2\pi x_n) \right] \right\} + \sum_{i=1}^{n} u (x_i, 5, 100, 4)$ | $[-50, 50]^D$ | 0 |
| $F_{14}(x) = \left( \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$ | $[-65.53, 65.53]^D$ | 1 |
| $F_{15}(x) = \sum_{i=1}^{11} \left[ a_i - \frac{x_1 \left( b_i^2 + b_i x_2 \right)}{b_i^2 + b_i x_3 + x_4} \right]^2$ | $[-5, 5]^D$ | 0.0003 |
| $F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$ | $[-5, 5]^D$ | $-1.0316$ |
| $F_{17}(x) = \left( x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \right) \cos (x_1) + 10$ | $[-5, 10] \times [0, 15]$ | 0.398 |
| $F_{18}(x) = \left( 1 + (x_1 + x_2 + 1)^2 \right) \left( 19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2 \right) \times \left( 30 + (2x_1 - 3x_2)^2 \left( 18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2 \right) \right)$ | $[-5, 5]^D$ | 3 |
| $F_{19}(x) = -\sum_{i=1}^{4} c_i \exp \left( -\sum_{j=1}^{3} a_{ij} \left( x_j - p_{ij} \right)^2 \right)$ | $[0, 1]^D$ | $-3.86$ |
| $F_{20}(x) = -\sum_{i=1}^{4} c_i \exp \left( -\sum_{j=1}^{6} a_{ij} \left( x_j - p_{ij} \right)^2 \right)$ | $[0, 1]^D$ | $-3.32$ |
| $F_{21}(x) = -\sum_{i=1}^{5} \left[ (X - a_i) (X - a_i)^T + c_i \right]^{-1}$ | $[0, 10]^D$ | $-10.1532$ |
| $F_{22}(x) = -\sum_{i=1}^{7} \left[ (X - a_i) (X - a_i)^T + c_i \right]^{-1}$ | $[0, 10]^D$ | $-10.4028$ |
| $F_{23}(x) = -\sum_{i=1}^{10} \left[ (X - a_i) (X - a_i)^T + c_i \right]^{-1}$ | $[0, 10]^D$ | $-10.5363$ |
| **CEC2005 - Hybrid Composition Function with Dimension $D$ = 30** | Solution space (S) | $f_{\min}$ |
| $F_{24}$ - *Hybrid Composition Function (CF15)* | $[-5, 5]$ | 120 |
| $F_{25}$ - *Rotated Version of Hybrid Composition Function (CF16)* | $[-5, 5]$ | 120 |
| $F_{26}$ - *Rotated Hybrid Composition Function with Noise in Fitness (CF17)* | $[-5, 5]$ | 120 |
| $F_{27}$ - *Rotated Hybrid Composition Function (CF18)* | $[-5, 5]$ | 10 |
| $F_{28}$ - *Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum (CF19)* | $[-5, 5]$ | 10 |
| $F_{29}$ - *Rotated Hybrid Composition Function with Global Optimum on the Bounds (CF20)* | $[-5, 5]$ | 10 |

Algorithm (RSA) [44] and Aquila Optimizer (AO) [45]. Among them, PSO, GWO and WOA are considered to be the most popular algorithms, and they are usually employed as classic examples to compare with the new proposed optimization algorithms. In this work, we select two more recently released optimization, namely (SMA) proposed in year of 2020, and (AOA), (AO) and (RSA) proposed in years of 2021 and 2022, respectively. The parameters of these algorithms are described in Table 3. For this test, all the algorithms mentioned above are implemented with the same initial conditions to achieve fairness in comparative experiments. Among them, the dimension of the solution space is $D = 30$, except for fixed-dimension functions (F14–F23) as shown in Table 1, the population size is $N = 30$ and the total number of iterations $T_{max} = 1000$. So, all five algorithms have the same number of function evaluations, i.e. $NFE = 30.000$. To limit the impacts of random factors in the algorithms on the results, all the compared algorithms were run individually 30 times in each function to gain the best values of two items, including the mean

**Table 2**
Comparison results on the 23 first classical benchmark functions with different population size with the same *NEF* = 10,000.

| Function | Different POPULATION with *NEF* = 10,000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $N = 20$ | $N = 30$ | $N = 40$ | $N = 50$ | $N = 60$ | $N = 70$ | $N = 80$ | $N = 100$ |
| F1 | Mean | 2.690E−264 | 6.585E−197 | 1.661E−159 | 2.824E−130 | 1.197E−109 | 9.611E−97 | 1.035E−86 | 4.919E−72 |
| | Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | Std | 0.000E+00 | 0.000E+00 | 0.000E+00 | 6.205E−130 | 3.227E−109 | 2.746E−96 | 5.518E−86 | 1.972E−71 |
| F2 | Mean | 4.414E−125 | 3.002E−93 | 7.985E−77 | 9.905E−67 | 6.399E−56 | 5.044E−49 | 8.503E−45 | 4.473E−37 |
| | Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | Std | 2.183E−124 | 1.566E−92 | 3.286E−76 | 1.248E−66 | 1.561E−55 | 8.740E−49 | 2.150E−44 | 9.441E−37 |
| F3 | Mean | 1.458E−178 | 3.733E−129 | 5.310E−101 | 3.479E−84 | 6.355E−74 | 1.276E−63 | 3.778E−57 | 1.071E−48 |
| | Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | Std | 0.000E+00 | 1.966E−128 | 1.876E−100 | 1.086E−83 | 1.824E−73 | 6.279E−63 | 2.061E−56 | 3.443E−48 |
| F4 | Mean | 4.378E−120 | 3.584E−189 | 8.924E−74 | 2.539E−62 | 1.854E−53 | 2.817E−47 | 6.944E−43 | 1.301E−35 |
| | Rank | 2 | 1 | 3 | 4 | 5 | 6 | 7 | 8 |
| | Std | 1.425E−119 | 1.417E−188 | 4.202E−73 | 4.470E−62 | 2.644E−53 | 4.708E−47 | 2.519E−42 | 3.513E−35 |
| F5 | Mean | 7.113E+00 | 6.990E+00 | 7.013E+00 | 7.163E+00 | 7.058E+00 | 6.932E+00 | 7.050E+00 | 7.064E+00 |
| | Rank | 7 | 2 | 3 | 8 | 5 | 1 | 4 | 6 |
| | Std | 4.425E−01 | 4.579E−01 | 5.316E−01 | 3.966E−01 | 4.703E−01 | 2.004E−01 | 3.535E−01 | 2.701E−01 |
| F6 | Mean | 3.879E−03 | 3.032E−03 | 3.810E−03 | 3.556E−03 | 3.201E−03 | 3.229E−03 | 5.537E−03 | 4.677E−03 |
| | Rank | 6 | 1 | 5 | 4 | 2 | 3 | 8 | 7 |
| | Std | 2.711E−03 | 1.919E−03 | 1.989E−03 | 3.201E−03 | 1.835E−03 | 1.831E−03 | 6.376E−03 | 2.935E−03 |
| F7 | Mean | 7.504E−04 | 6.661E−04 | 6.103E−04 | 7.658E−04 | 3.472E−04 | 7.220E−04 | 7.162E−04 | 7.264E−04 |
| | Rank | 7 | 3 | 2 | 8 | 1 | 5 | 4 | 6 |
| | Std | 8.096E−04 | 6.212E−04 | 5.754E−04 | 6.044E−04 | 2.304E−04 | 5.258E−04 | 6.129E−04 | 8.789E−04 |
| F8 | Mean | −2.665E+03 | −2.675E+03 | −2.560E+03 | −2.528E+03 | −2.796E+03 | −2.442E+03 | −2.581E+03 | −2.505E+03 |
| | Rank | 3 | 2 | 5 | 6 | 1 | 8 | 4 | 7 |
| | Std | 5.703E+02 | 5.126E+02 | 6.369E+02 | 5.628E+02 | 5.572E+02 | 4.615E+02 | 4.712E+02 | 4.569E+02 |
| F9 | Mean | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| | Rank | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F10 | Mean | 8.882E−16 | 8.882E−16 | 8.882E−16 | 8.882E−16 | 8.882E−16 | 8.882E−16 | 8.882E−16 | 8.882E−16 |
| | Rank | 3 | 3 | 3 | 1 | 1 | 3 | 3 | 3 |
| | Std | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F11 | Mean | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| | Rank | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F12 | Mean | 2.227E−03 | 3.584E−03 | 2.441E−03 | 3.615E−03 | 3.447E−03 | 3.614E−03 | 3.071E−03 | 3.831E−03 |
| | Rank | 1 | 5 | 2 | 7 | 4 | 6 | 3 | 8 |
| | Std | 2.472E−03 | 3.095E−03 | 2.755E−03 | 2.663E−03 | 2.042E−03 | 2.625E−03 | 2.892E−03 | 3.271E−03 |
| F13 | Mean | 6.286E−03 | 6.144E−03 | 5.239E−03 | 6.317E−03 | 4.522E−03 | 7.006E−03 | 7.237E−03 | 8.274E−03 |
| | Rank | 4 | 3 | 2 | 5 | 1 | 6 | 7 | 8 |
| | Std | 3.308E−03 | 4.007E−03 | 3.533E−03 | 4.443E−03 | 3.025E−03 | 4.031E−03 | 5.672E−03 | 5.439E−03 |
| F14 | Mean | 2.650E+00 | 2.128E+00 | 2.167E+00 | 2.672E+00 | 2.479E+00 | 1.159E+00 | 1.854E+00 | 1.713E+00 |
| | Rank | 7 | 4 | 5 | 8 | 6 | 1 | 3 | 2 |
| | Std | 2.932E+00 | 2.176E+00 | 2.339E+00 | 2.820E+00 | 2.637E+00 | 4.183E−01 | 2.185E+00 | 1.393E+00 |
| F15 | Mean | 5.220E−04 | 5.212E−04 | 5.053E−04 | 6.774E−04 | 4.568E−04 | 5.189E−04 | 5.140E−04 | 4.482E−04 |
| | Rank | 7 | 6 | 3 | 8 | 2 | 5 | 4 | 1 |
| | Std | 1.273E−04 | 1.355E−04 | 1.036E−04 | 3.924E−04 | 1.054E−04 | 1.196E−04 | 1.193E−04 | 1.223E−04 |
| F16 | Mean | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 |
| | Rank | 3 | 3 | 3 | 1 | 2 | 3 | 3 | 3 |
| | Std | 5.241E−06 | 1.529E−05 | 9.835E−06 | 7.980E−06 | 1.062E−05 | 1.095E−05 | 2.280E−05 | 1.034E−05 |
| F17 | Mean | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 |
| | Rank | 1 | 1 | 1 | 8 | 7 | 1 | 1 | 1 |
| | Std | 2.619E−06 | 3.161E−06 | 2.371E−06 | 4.541E−06 | 2.735E−06 | 3.105E−06 | 3.903E−06 | 4.264E−06 |
| F18 | Mean | 3.000E+00 | 3.000E+00 | 3.000E+00 | 3.000E+00 | 3.000E+00 | 3.000E+00 | 3.000E+00 | 3.000E+00 |
| | Rank | 3 | 2 | 5 | 1 | 8 | 3 | 5 | 5 |
| | Std | 1.723E−04 | 1.595E−04 | 4.003E−04 | 9.835E−05 | 5.146E−04 | 1.537E−04 | 5.366E−04 | 3.020E−04 |
| F19 | Mean | −3.862E+00 | −3.862E+00 | −3.862E+00 | −3.862E+00 | −3.862E+00 | −3.862E+00 | −3.862E+00 | −3.862E+00 |
| | Rank | 1 | 5 | 5 | 2 | 3 | 7 | 7 | 4 |
| | Std | 8.644E−04 | 1.292E−03 | 1.253E−03 | 6.765E−04 | 1.235E−03 | 9.499E−04 | 1.780E−03 | 8.607E−04 |
| F20 | Mean | −3.237E+00 | −3.228E+00 | −3.204E+00 | −3.229E+00 | −3.239E+00 | −3.243E+00 | −3.238E+00 | −3.222E+00 |
| | Rank | 4 | 6 | 8 | 5 | 2 | 1 | 3 | 7 |
| | Std | 8.303E−02 | 7.392E−02 | 1.015E−01 | 6.552E−02 | 8.190E−02 | 7.063E−02 | 8.026E−02 | 8.804E−02 |
| F21 | Mean | −8.616E+00 | −9.965E+00 | −9.398E+00 | −1.014E+01 | −1.013E+01 | −1.003E+01 | −9.998E+00 | −1.008E+01 |
| | Rank | 8 | 6 | 7 | 1 | 2 | 4 | 5 | 3 |
| | Std | 2.372E+00 | 9.280E−01 | 1.587E+00 | 1.774E−02 | 2.780E−02 | 5.654E−01 | 7.009E−01 | 2.051E−01 |

(*continued on next page*)

**Table 2** (*continued*).

| Function | Different POPULATION with *NEF* = 10,000 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $N = 20$ | $N = 30$ | $N = 40$ | $N = 50$ | $N = 60$ | $N = 70$ | $N = 80$ | $N = 100$ |
| F22 | Mean −9.608E+00 | −9.882E+00 | −9.772E+00 | −1.036E+01 | −1.039E+01 | −1.038E+01 | −1.009E+01 | −1.039E+01 |
| | Rank 8 | 6 | 7 | 4 | 1 | 3 | 5 | 2 |
| | Std 1.826E+00 | 1.552E+00 | 1.632E+00 | 4.906E−02 | 2.010E−02 | 3.546E−02 | 1.148E+00 | 1.472E−02 |
| F23 | Mean −9.877E+00 | −1.047E+01 | −1.034E+01 | −1.052E+01 | −1.052E+01 | −1.052E+01 | −1.034E+01 | −1.050E+01 |
| | Rank 8 | 5 | 6 | 2 | 3 | 1 | 7 | 4 |
| | Std 1.725E+00 | 1.370E+00 | 9.854E−01 | 1.620E−02 | 2.578E−02 | 1.205E−02 | 9.843E−01 | 9.457E−02 |
| Sum | 88 | 72 | 86 | 97 | 73 | 87 | 106 | 111 |
| Mean rank | 3.826 | 3.130 | 3.739 | 4.217 | 3.174 | 3.783 | 4.609 | 4.826 |
| Final rank | 5 | 1 | 3 | 6 | 2 | 4 | 7 | 8 |

*Note: The symbol* $(E \pm x)$ *is equal* $(10^{\pm x})$.

**Table 3**
Parameter settings for the other optimization algorithm.

| Algorithm | Parameter | Value |
|---|---|---|
| KO | *threshold* parameter | 0.5 |
| PSO | Cognitive and social | $c_1 = 2, c_2 = 2$ |
| | Inertia weight | $w = 0.5$ |
| | Velocity limit | $0.1(U_b - L_b)$; $U_b$, $L_b$ are lower and upper of boundary condition |
| GWO | Convergence parameter | $\vec{d}$ linearly decreased from 2 to 0 |
| | $\vec{r_1}$, $\vec{r_2}$ | Random vector having value in range [0, 1] |
| WOA | The shape of the logarithmic spiral | $b = 1$ |
| | Convergence parameter | $\vec{d}$ linearly decreased from 2 to 0 |
| | The random number parameter | $l = 1$ |
| SMA | $z$ parameter | $z = 0.03$ |
| | The other parameters | As the default |
| AOA | The sensitive parameter | $\alpha = 5$ |
| | $\mu$ parameters | $\mu = 0.5$ |
| | Minimum value of accelerated function | $Min = 0.2$ |
| | Maximum value of accelerated function | $Max = 1$ |
| RSA | The sensitive parameter | $\alpha = 0.1, \beta = 0.1$ |
| AO | The exploitation adjustment parameters | $\alpha = 0.1, \delta = 0.1$ |
| | The parameter for calculating the Lévy flight | $\beta = 1.5$ |

The results of comparison are detailed in.

value and the standard deviation. At the same time, the Friedman ranking test has been carried out for ranking comparison of the above-mentioned algorithms in a statistical way.

In Table 4. KO wins absolute supremacy over almost all functions, for the function having one global optimum (F1–F4), AOA and RSA are placed first, followed by KO. However, it can be seen that there is no big difference between AOA, RSA and KO. For the functions (F5–F7), AO, SMA and RSA are shown to have better performance than KO, however KO still has better performance than PSO, GWO, AOA, WOA algorithms. For the multimodal functions (F8–F13), which have many local optimum. In general, KO, SMA and AO show a superior performance compared to the rest of the algorithms.

For fixed-dimension functions (F14–F23), the results show that KO can exploit the global optimum value in almost functions. Meanwhile the remaining algorithms sometimes fail to find the best solution for some functions. Note that these functions are multimodal functions, and are used to evaluate the ability for escaping the local optimum in a fixed dimension. Although the ability to escape the local optimal of SMA is well evaluated in the first 23 benchmark functions, However, SMA fails in more complex functions, especially (F24, F25, F26). Meanwhile, KO emerges as the most efficient algorithm. The best fitness reported by KO is straight ahead of all the rest of the algorithms, even SMA for functions (F24, F25, F26). This is achieved by the flexible movement strategy for exploration in KO, thus each search agent in KO can reset the search spaces to escape of the local optimum. KO does not record a worst performance in any function, while SMA shows a poor performance for (F24, F25, F26) functions. The last functions (F27, F28, F29) KO is still ranked first again together with RSA and AO. This proves that the balance between the ability of exploration and exploitation in KO is still superior over that of the other optimization algorithms. Based on the Friedman ranking test, in general, KO is rated the highest, followed by SMA, AO and RSA. Four optimization algorithms including PSO, GWO, WOA and AOA have the worst performance for this test.

### 5.5. Comparison between KO and the other optimization algorithms for CEC2014 benchmark functions

Although KO performs well in the previous example, CEC2014 benchmark functions are used in this section to test the performance of all algorithms for solving complex problems in a more rigorous manner, as given in Table 5. They are used to check the algorithms' performance for large scale dimensions. For this comparison, L-SHADE optimization algorithm [46], which is one of winners in CEC2014, is used as standard measurement to evaluate the effectiveness of KO. The dimensions $D = 50$, and $D = 100$ are investigated in this example. The parameters in L-SHADE are considered as default, and the parameters of the rest of algorithms remain the same as in Table 3. The number of evaluation functions $NEF = 30.000$ is set for all algorithms. A total number of 30 functions is examined. These functions are arranged from simple to complex. Especially, the ability to escape from local optima of algorithms will face great challenges for

**Table 4**
Comparison of 29 classical benchmark functions with different optimization algorithms with D=30, except for fixed-dimension functions (F14–F23).

| Function | Different algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | KO | PSO | GWO | WOA | SMA | AOA | RSA | AO |
| F1 | Mean | 0.000E+00 | 1.862E−10 | 6.262E−66 | 7.627E−150 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 9.136E−206 |
| | Rank | 1 | 8 | 7 | 6 | 1 | 1 | 1 | 5 |
| | Std | 0.000E+00 | 4.241E−10 | 1.342E−65 | 4.155E−149 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F2 | Mean | 1.121E−299 | 1.481E+01 | 3.537E−39 | 7.577E−100 | 7.339E−208 | 0.000E+00 | 0.000E+00 | 5.554E−107 |
| | Rank | 3 | 8 | 7 | 6 | 4 | 1 | 1 | 5 |
| | Std | 0.000E+00 | 3.907E+01 | 4.827E−39 | 4.146E−99 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 3.042E−106 |
| F3 | Mean | 0.000E+00 | 5.152E+02 | 5.776E−12 | 1.892E+04 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 2.433E−196 |
| | Rank | 1 | 7 | 6 | 8 | 1 | 1 | 1 | 5 |
| | Std | 0.000E+00 | 1.274E+03 | 1.172E−11 | 1.053E+04 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F4 | Mean | 7.792E−279 | 3.874E+00 | 1.061E−15 | 3.226E+01 | 8.285E−194 | 0.000E+00 | 0.000E+00 | 2.109E−99 |
| | Rank | 3 | 7 | 6 | 8 | 4 | 1 | 1 | 5 |
| | Std | 0.000E+00 | 1.453E+00 | 1.355E−15 | 2.648E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 1.155E−98 |
| F5 | Mean | 2.704E+01 | 4.464E+01 | 2.736E+01 | 2.729E+01 | 3.770E+00 | 2.900E+01 | 6.767E+00 | 1.401E−03 |
| | Rank | 4 | 8 | 6 | 5 | 2 | 7 | 3 | 1 |
| | Std | 5.592E−01 | 3.310E+01 | 5.394E−01 | 6.531E−01 | 9.194E+00 | 0.000E+00 | 1.248E+01 | 2.258E−03 |
| F6 | Mean | 8.626E−02 | 1.148E−10 | 5.998E−01 | 8.149E−02 | 8.351E−04 | 7.500E+00 | 7.128E+00 | 5.534E−05 |
| | Rank | 5 | 1 | 6 | 4 | 3 | 8 | 7 | 2 |
| | Std | 5.593E−02 | 1.382E−10 | 3.322E−01 | 9.228E−02 | 3.630E−04 | 0.000E+00 | 5.458E−01 | 8.069E−05 |
| F7 | Mean | 2.217E−04 | 2.333E−02 | 6.915E−05 | 1.930E−03 | 7.306E−05 | 3.786E−05 | 5.538E−05 | 5.421E−05 |
| | Rank | 6 | 8 | 4 | 7 | 5 | 1 | 3 | 2 |
| | Std | 1.850E−04 | 7.324E−03 | 6.312E−05 | 2.412E−03 | 6.234E−05 | 3.714E−05 | 5.433E−05 | 5.316E−05 |
| F8 | Mean | −6.798E+03 | −7.611E+03 | −5.512E+03 | −1.152E+04 | −1.257E+04 | −2.803E+03 | −5.487E+03 | −9.070E+03 |
| | Rank | 5 | 4 | 6 | 2 | 1 | 8 | 7 | 3 |
| | Std | 1.457E+03 | 7.704E+02 | 5.478E+02 | 1.570E+03 | 6.572E−02 | 4.329E+02 | 2.904E+02 | 3.672E+03 |
| F9 | Mean | 0.000E+00 | 4.365E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| | Rank | 1 | 8 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 1.225E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F10 | Mean | 8.882E−16 | 5.488E−02 | 1.007E−15 | 4.678E−15 | 8.882E−16 | 8.882E−16 | 8.882E−16 | 8.882E−16 |
| | Rank | 3 | 8 | 6 | 7 | 3 | 3 | 1 | 1 |
| | Std | 0.000E+00 | 3.006E−01 | 6.486E−16 | 2.628E−15 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F11 | Mean | 0.000E+00 | 1.694E−02 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| | Rank | 1 | 8 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 1.908E−02 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F12 | Mean | 7.326E−03 | 4.492E−02 | 5.779E−02 | 9.341E−03 | 8.504E−04 | 1.664E+00 | 1.330E+00 | 9.017E−07 |
| | Rank | 3 | 5 | 6 | 4 | 2 | 8 | 7 | 1 |
| | Std | 3.703E−03 | 8.898E−02 | 1.938E−02 | 1.786E−02 | 1.012E−03 | 2.576E−02 | 4.253E−01 | 1.973E−06 |
| F13 | Mean | 4.136E−02 | 2.198E−03 | 4.463E−01 | 2.318E−01 | 8.781E−04 | 2.998E+00 | 1.567E−01 | 5.343E−06 |
| | Rank | 4 | 3 | 7 | 6 | 2 | 8 | 5 | 1 |
| | Std | 1.708E−02 | 4.470E−03 | 2.177E−01 | 1.613E−01 | 4.517E−04 | 1.180E−02 | 6.202E−01 | 8.353E−06 |
| F14 | Mean | 1.012E+00 | 1.921E+00 | 4.855E+00 | 1.947E+00 | 9.980E−01 | 1.122E+01 | 4.287E+00 | 2.595E+00 |
| | Rank | 2 | 3 | 7 | 4 | 1 | 8 | 6 | 5 |
| | Std | 7.668E−02 | 1.657E+00 | 3.603E+00 | 2.495E+00 | 6.054E−14 | 3.072E+00 | 3.564E+00 | 3.508E+00 |
| F15 | Mean | 3.960E−04 | 1.201E−03 | 1.729E−03 | 7.592E−04 | 5.192E−04 | 9.618E−02 | 2.324E−03 | 4.255E−04 |
| | Rank | 1 | 5 | 6 | 4 | 3 | 8 | 7 | 2 |
| | Std | 9.605E−05 | 3.648E−03 | 5.107E−03 | 5.152E−04 | 2.919E−04 | 5.388E−02 | 1.851E−03 | 8.721E−05 |
| F16 | Mean | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 | −1.032E+00 | −5.244E−01 | −1.031E+00 | −1.031E+00 |
| | Rank | 1 | 1 | 1 | 1 | 1 | 8 | 7 | 6 |
| | Std | 3.549E−06 | 6.775E−16 | 1.375E−08 | 7.574E−11 | 1.049E−10 | 3.866E−01 | 8.282E−04 | 2.107E−04 |
| F17 | Mean | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 | 3.979E−01 | 4.200E−01 | 3.979E−01 | 3.980E−01 |
| | Rank | 1 | 1 | 1 | 1 | 1 | 8 | 1 | 7 |
| | Std | 4.812E−07 | 0.000E+00 | 4.587E−07 | 5.695E−07 | 4.921E−08 | 2.801E−02 | 6.812E−07 | 3.258E−03 |
| F18 | Mean | 3.000E+00 | 3.000E+00 | 5.194E+00 | 3.000E+00 | 3.000E+00 | 3.116E+02 | 4.821E+00 | 3.014E+00 |
| | Rank | 1 | 1 | 7 | 1 | 1 | 8 | 6 | 5 |
| | Std | 2.474E−05 | 1.353E−15 | 1.202E+01 | 2.110E−05 | 1.401E−12 | 2.318E+02 | 6.930E+00 | 1.390E−02 |
| F19 | Mean | −3.863E+00 | −3.863E+00 | −3.858E+00 | −3.861E+00 | −3.863E+00 | −3.854E+00 | −3.781E+00 | −3.860E+00 |
| | Rank | 1 | 1 | 6 | 4 | 1 | 7 | 8 | 5 |
| | Std | 2.820E−04 | 2.696E−15 | 2.658E−03 | 2.364E−03 | 3.079E−08 | 2.956E−03 | 5.936E−02 | 2.737E−03 |
| F20 | Mean | −3.248E+00 | −3.280E+00 | −3.230E+00 | −3.225E+00 | −3.235E+00 | −3.106E+00 | −2.665E+00 | −3.158E+00 |
| | Rank | 2 | 1 | 4 | 5 | 3 | 7 | 8 | 6 |
| | Std | 6.545E−02 | 6.124E−02 | 8.386E−02 | 1.382E−01 | 5.349E−02 | 6.244E−02 | 1.606E−01 | 6.757E−02 |
| F21 | Mean | −1.015E+01 | −6.484E+00 | −6.269E+00 | −8.703E+00 | −1.015E+01 | −3.909E+00 | −5.055E+00 | −1.015E+01 |
| | Rank | 1 | 5 | 6 | 4 | 1 | 8 | 7 | 3 |
| | Std | 1.211E−03 | 3.577E+00 | 2.183E+00 | 2.460E+00 | 1.209E−04 | 1.232E+00 | 3.446E−07 | 9.149E−03 |

(*continued on next page*)

**Table 4** (*continued*).

| Function | Different algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | KO | PSO | GWO | WOA | SMA | AOA | RSA | AO |
| F22 | Mean | −1.040E+01 | −5.969E+00 | −9.337E+00 | −8.076E+00 | −1.040E+01 | −4.303E+00 | −5.088E+00 | −1.040E+01 |
| | Rank | 1 | 6 | 4 | 5 | 1 | 8 | 7 | 3 |
| | Std | 1.727E−03 | 3.515E+00 | 2.055E+00 | 2.954E+00 | 1.034E−04 | 1.538E+00 | 7.630E−07 | 1.226E−02 |
| F23 | Mean | −1.054E+01 | −8.002E+00 | −1.054E+01 | −8.359E+00 | −1.054E+01 | −4.305E+00 | −5.128E+00 | −1.053E+01 |
| | Rank | 1 | 6 | 1 | 5 | 1 | 8 | 7 | 4 |
| | Std | 7.776E−04 | 3.683E+00 | 5.810E−05 | 3.153E+00 | 1.272E−04 | 1.823E+00 | 1.918E−06 | 5.950E−03 |
| F24 (CEC2005) | Mean | 5.823E+02 | 5.570E+02 | 6.752E+02 | 9.378E+02 | 1.710E+03 | 1.270E+03 | 1.203E+03 | 7.689E+02 |
| | Rank | 2 | 1 | 3 | 5 | 8 | 7 | 6 | 4 |
| | Std | 5.144E+01 | 1.702E+02 | 5.118E+01 | 1.822E+02 | 2.313E−13 | 9.140E+01 | 1.029E+02 | 6.893E+01 |
| F25 (CEC2005) | Mean | 4.183E+02 | 4.902E+02 | 4.860E+02 | 6.511E+02 | 1.830E+03 | 1.200E+03 | 9.847E+02 | 5.044E+02 |
| | Rank | 1 | 3 | 2 | 5 | 8 | 7 | 6 | 4 |
| | Std | 1.558E+02 | 2.008E+02 | 1.519E+02 | 1.109E+02 | 2.313E−13 | 1.622E+02 | 8.471E+01 | 4.140E+01 |
| F26 (CEC2005) | Mean | 4.518E+02 | 4.923E+02 | 6.403E+02 | 7.475E+02 | 1.274E+03 | 1.210E+03 | 1.071E+03 | 4.720E+02 |
| | Rank | 1 | 3 | 4 | 5 | 8 | 7 | 6 | 2 |
| | Std | 1.322E+02 | 1.234E+02 | 1.558E+02 | 1.000E+02 | 1.171E+02 | 1.404E+02 | 1.071E+03 | 4.720E+02 |
| F27 (CEC2005) | Mean | 9.100E+02 | 9.810E+02 | 9.901E+02 | 1.105E+03 | 9.100E+02 | 9.100E+02 | 9.100E+02 | 9.100E+02 |
| | Rank | 1 | 6 | 7 | 8 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 3.749E+01 | 3.061E+01 | 9.644E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F28 (CEC2005) | Mean | 9.100E+02 | 9.839E+02 | 9.841E+02 | 1.109E+03 | 9.100E+02 | 9.100E+02 | 9.100E+02 | 9.100E+02 |
| | Rank | 1 | 6 | 7 | 8 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 3.222E+01 | 2.012E+01 | 9.051E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| F29 (CEC2005) | Mean | 9.100E+02 | 9.908E+02 | 9.812E+02 | 1.116E+03 | 9.100E+02 | 9.100E+02 | 9.100E+02 | 9.100E+02 |
| | Rank | 1 | 7 | 6 | 8 | 1 | 1 | 1 | 1 |
| | Std | 0.000E+00 | 3.774E+01 | 1.648E+01 | 9.934E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 |
| Sum | | 59 | 139 | 141 | 138 | 71 | 151 | 124 | 92 |
| Mean rank | | 2.034 | 4.793 | 4.862 | 4.759 | 2.448 | 5.207 | 4.276 | 3.172 |
| Final rank | | 1 | 6 | 7 | 5 | 2 | 8 | 4 | 3 |

*Note: The symbol* $(E \pm x)$ *is equal* $(10^{\pm x})$.

hybrid functions (CF16–CF22) functions and composition functions (CF23–CF30). All algorithms are run for 30 independent runs to find the best mean value and the best standard deviation. Friedman mean ranking test is employed to assess the general performance of each algorithm. At the same time, to assess the significant difference between KO and the rest of the algorithms, Wilcoxon rank-sum test, for which the significance level of $\alpha$ is selected as 0.05, is used. Thus, the performance of KO and peer algorithms is compared for the results of the values of $p_{value}$. If $p_{value} > \alpha$, there is no significant difference between the pair of algorithms. If $p_{value} < \alpha$, a significant difference is observed between the pair of algorithms. The statistical tests using Wilcoxon rank-sum test for $D = 50$ and $D = 100$ are shown in Table 7 and Table 9, respectively. Note that, in these tables, the notation (+) indicates a better performance of KO compared to other mentioned algorithms, (-) indicates a worse performance of KO compared to other mentioned algorithm, and (∼) indicates no significant difference of performance between the compared algorithms and KO.

Following the statistical results obtained, the L-SHADE, has the best performance for $D = 50$, followed by KO. However, when the search dimension increase from $D = 50$ to $D = 100$, KO registers a robust improvement and it ranks first based on the Friedman mean ranking test, followed by L-SHADE. For the rest of the algorithms, KO shows a superiority. The values achieved by KO shows a big difference compared to AO and PSO, which are ranked as third and fourth, respectively. SMA is very competitive with KO for the first 29 classical functions in the previous example. But in this example, SMA shows the worse performance compared to KO. Meanwhile, the stability of the KO is maintained. This proves that, KO introduces an effective and stable movement to achieve a good balance between the ability of exploration and exploitation. In both cases of $D = 50$ and $D = 100$, based on Wilcoxon rank-sum statistic test, KO shows superiority in comparison with the other algorithms except L-SHADE. Especially in solving problems with multiple local optima's problem for (CF23–CF29), the results obtained by KO are even better than those of L-SHADE (Tables 6 and 8).

## 6. Engineering design problems

To demonstrate the effectiveness of the KO algorithm in solving real-world problems. In this section, three well-known engineering problems including tension/compression spring, pressure vessel design, and welded beam design are used to test the KO's performance for solving engineering problems. To consider constrained optimization problems, the penalty method is used to handle the constraint violations. The penalty parameter is selected at $10^{10}$ to penalize the objective function in equality/inequality constraints. All engineering design problems will be solved by using $N = 50$ and $T_{\max} = 5000$, and the process is implemented with 50 independence runs to report the best solution and the best fitness for each problem. The results obtained by KO are compared with several similar techniques published in the literature.

### 6.1. The tension/compression spring

The objective of this problem is to find the minimum weight of spring. There are three design variables including wire diameter ($d$), mean coil diameter ($D$) and number of active coils ($N$) as shown in Fig. 14. The constraints are established by minimum shear stress, deflection, and surge frequency. The mathematical formulation and four constraints of this problem are expressed as follows:

**Table 5**
Summary of the CEC'14 test functions.

| Function | Description | *Dimension* | $f_{min}$ | Note |
|---|---|---|---|---|
| *CF1* | *Rotated High Conditioned Elliptic Function* | *D* | *100* | *Unimodal Functions* |
| *CF2* | *Rotated Bent Cigar Function* | *D* | *200* | |
| *CF3* | *Rotated Discus Function* | *D* | *300* | |
| *CF4* | *Shifted and Rotated Rosenbrock's Function* | *D* | *400* | *Simple Multimodal Functions* |
| *CF5* | *Shifted and Rotated Ackley's Function* | *D* | *500* | |
| *CF6* | *Shifted and Rotated Weierstrass Function* | *D* | *600* | |
| *CF7* | *Shifted and Rotated Griewank's Function* | *D* | *700* | |
| *CF8* | *Shifted Rastrigin's Function* | *D* | *800* | |
| *CF9* | *Shifted and Rotated Rastrigin's Function* | *D* | *900* | |
| *CF10* | *Shifted Schwefel's Function* | *D* | *1000* | |
| *CF11* | *Shifted and Rotated Schwefel's Function* | *D* | *1100* | |
| *CF12* | *Shifted and Rotated Katsuura Function* | *D* | *1200* | |
| *CF13* | *Shifted and Rotated HappyCat Function* | *D* | *1300* | |
| *CF14* | *Shifted and Rotated HGBat Function* | *D* | *1400* | |
| *CF15* | *Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function* | *D* | *1500* | |
| *CF16* | *Shifted and Rotated Expanded Scaffer's F6 Function* | *D* | *1600* | *Hybrid Function* |
| *CF17* | *Hybrid Function 1 (N = 3)* | *D* | *1700* | |
| *CF18* | *Hybrid Function 2 (N = 3)* | *D* | *1800* | |
| *CF19* | *Hybrid Function 3 (N = 4)* | *D* | *1900* | |
| *CF20* | *Hybrid Function 4 (N = 4)* | *D* | *2000* | |
| *CF21* | *Hybrid Function 5 (N = 5)* | *D* | *2100* | |
| *CF22* | *Hybrid Function 6 (N = 5)* | *D* | *2200* | |
| *CF23* | *Composition Function 1 (N = 5)* | *D* | *2300* | *Composition Functions* |
| *CF24* | *Composition Function 2 (N = 3)* | *D* | *2400* | |
| *CF25* | *Composition Function 3 (N = 3)* | *D* | *2500* | |
| *CF26* | *Composition Function 4 (N = 5)* | *D* | *2600* | |
| *CF27* | *Composition Function 5 (N = 5)* | *D* | *2700* | |
| *CF28* | *Composition Function 6 (N = 5)* | *D* | *2800* | |
| *CF29* | *Composition Function 7 (N = 3)* | *D* | *2900* | |
| *CF30* | *Composition Function 8 (N = 3)* | *D* | *3000* | |
| *Solution search range* [−100, 100] | | | | |

Design variables: $x_1 = d,\ x_2 = D,\ x_3 = P$

Minimize the objective function: $f(x) = (x_3 + 2)x_2x_1^2$

Subject to:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0; \quad g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0; \quad g_3(x) = 1 - \frac{140.25x_1}{x_2^2x_3} \le 0$$

$g_4(x) = \frac{x_1+x_2}{1.5} - 1 \le 0$; Where $0.05 \le x_1 \le 2, 0.25 \le x_2 \le 1.3, 2 \le x_3 \le 15$

The results of design variables, the value of constrained functions and the best fitness obtained by KO are listed in Table 10. And the results of the different methods in the literature are listed in Table 11. The results recorded by CSA [47] are the best $f(x) = 0.01266523$, followed by AVOA [33]. The best result obtained by KO is $f(x) = 0.012665994155756$. This value is very competitive with that of the HHO [48], MPA [43] and EO [49]. Although KO fails to find the best solution in this problem, there is no big difference between CSA [47], AVOA [33] and KO. Meanwhile, KO still shows a superiority in comparison with the other methods.

### 6.2. Pressure vessel design

This problem is one of the most well-known benchmark design tests. The primary objective is to minimize the overall cost with four optimization variables, including material, forming, and welding of a cylindrical vessel as shown in Fig. 15. There are four linear and nonlinear constraints affecting the design of pressure vessel. $T_s$, $T_h$ are the thickness of the shell and the thickness of the head, respectively. The inner radius is $R$, and the length of the cylindrical section without considering the head is $L$.

The mathematical formulation of this problem is expressed as follows:

Give design variables $x_1 = T_s,\ x_2 = T_h,\ x_3 = R,\ x_4 = L$

Minimize the objective function: $f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

Subject to

$$g_1(x) = -x_1 + 0.0193x_3 \le 0; \quad g_2(x) = -x_2 + 0.00954x_3 \le 0; \quad g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0$$

$$g_4(x) = x_4 - 240 \le 0$$

Where $0.0625 \le x_1 \le 99 \times 0.0625, 0.0625 \le x_2 \le 99 \times 0.0625, 10 \le x_3 \le 200, 10 \le x_4 \le 200$

Table 12 provides the statistical summary of results obtained by KO, meanwhile, Table 11 summarizes the optimal results for different algorithms. It can seen that the current best result obtained by other optimization algorithms is around $f(x) = 6059$. The best results obtained using KO is $f(x) = 6059.71475731827$, which is a little bit lower than that of CSA [47], EO [49] and MPA [43]. In comparison with the other optimization algorithms, KO still shows a better performance (see Table 13).

### 6.3. Welded beam design

The final example is to evaluate the performance of KO in the welded beam design. KO is employed to minimize the total fabrication cost of a welded beam subjected to seven constraints, which are related to shear stress ($\tau$), bending stress ($\sigma$), buckling load (Pc) and deflection ($\delta$). There are four design variables as shown in Fig. 16.

Give design variables: $x_1 = h,\ x_2 = l,\ x_3 = t,\ x_4 = b$

Minimize the objective function:

$f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4\,(14 + x_2)$

Subject to: $g_1(x) = \tau(x) - \tau^{max} \le 0$, $g_2(x) = \sigma(x) - \sigma^{max} \le 0$, $g_3(x) = x_1 - x_4 \le 0$, $g_4(x) = 1.1047x_1^2x_2 + 0.04811x_3x_4\,(14 + x_2) -$

**Table 6**
Comparison of CEC2014 benchmark functions for different optimization algorithms for $D = 50$.

| Function | Different algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | KO | PSO | GWO | WOA | SMA | AOA | RSA | AO | LSHADE |
| CF1 | Mean | 3.364E+07 | 8.947E+07 | 3.423E+08 | 2.801E+08 | 1.665E+10 | 4.719E+09 | 3.660E+09 | 3.062E+08 | 8.720E+05 |
| | Rank | 2 | 3 | 6 | 4 | 9 | 8 | 7 | 5 | 1 |
| | Std | 9.730E+06 | 8.346E+07 | 1.245E+08 | 8.570E+07 | 0.000E+00 | 2.086E+09 | 1.292E+09 | 1.160E+08 | 2.516E+05 |
| CF2 | Mean | 1.348E+07 | 7.713E+09 | 2.703E+10 | 1.480E+10 | 1.996E+11 | 1.654E+11 | 1.517E+11 | 9.886E+09 | 5.125E+03 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 5.152E+06 | 4.547E+09 | 6.512E+09 | 3.165E+09 | 3.104E−05 | 1.659E+10 | 1.059E+10 | 2.369E+09 | 2.193E+03 |
| CF3 | Mean | 8.591E+04 | 8.762E+04 | 1.428E+05 | 1.460E+05 | 6.963E+08 | 6.902E+05 | 1.480E+05 | 2.002E+05 | 3.880E+02 |
| | Rank | 2 | 3 | 4 | 5 | 9 | 8 | 6 | 7 | 1 |
| | S. Deviation | 1.143E+04 | 2.243E+04 | 1.281E+04 | 1.884E+04 | 3.637E−07 | 9.421E+05 | 1.299E+04 | 3.652E+04 | 5.858E+01 |
| CF4 | Mean | 6.580E+02 | 9.384E+02 | 3.983E+03 | 2.057E+03 | 7.299E+04 | 5.008E+04 | 3.695E+04 | 1.886E+03 | 4.973E+02 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 5.337E+01 | 2.763E+02 | 1.110E+03 | 4.896E+02 | 1.480E−11 | 8.911E+03 | 7.864E+03 | 4.350E+02 | 1.660E+00 |
| CF5 | Mean | 5.212E+02 | 5.211E+02 | 5.209E+02 | 5.210E+02 | 5.217E+02 | 5.213E+02 | 5.212E+02 | 5.212E+02 | 5.209E+02 |
| | Rank | 6 | 4 | 2 | 3 | 9 | 8 | 7 | 5 | 1 |
| | Std | 5.507E−02 | 6.447E−02 | 1.325E−01 | 8.033E−02 | 3.469E−13 | 7.806E−02 | 4.358E−02 | 6.110E−02 | 1.796E−01 |
| CF6 | Mean | 6.473E+02 | 6.289E+02 | 6.426E+02 | 6.703E+02 | 6.907E+02 | 6.761E+02 | 6.733E+02 | 6.588E+02 | 6.112E+02 |
| | Rank | 4 | 2 | 3 | 6 | 9 | 8 | 7 | 5 | 1 |
| | Std | 5.636E+00 | 3.644E+00 | 8.418E+00 | 4.230E+00 | 0.000E+00 | 3.294E+00 | 1.966E+00 | 3.829E+00 | 2.665E+00 |
| CF7 | Mean | 7.012E+02 | 7.829E+02 | 9.144E+02 | 8.274E+02 | 2.579E+03 | 2.247E+03 | 2.008E+03 | 7.812E+02 | 7.000E+02 |
| | Rank | 2 | 4 | 6 | 5 | 9 | 8 | 7 | 3 | 1 |
| | Std | 4.186E−02 | 4.864E+01 | 5.102E+01 | 4.092E+01 | 2.313E−12 | 1.940E+02 | 1.089E+02 | 1.915E+01 | 4.844E−03 |
| CF8 | Mean | 1.042E+03 | 9.510E+02 | 1.231E+03 | 1.236E+03 | 1.709E+03 | 1.603E+03 | 1.476E+03 | 1.119E+03 | 9.128E+02 |
| | Rank | 3 | 2 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 3.824E+01 | 2.651E+01 | 3.011E+01 | 6.176E+01 | 0.000E+00 | 4.265E+01 | 2.946E+01 | 2.708E+01 | 1.161E+01 |
| CF9 | Mean | 1.232E+03 | 1.130E+03 | 1.332E+03 | 1.467E+03 | 1.911E+03 | 1.742E+03 | 1.598E+03 | 1.301E+03 | 1.096E+03 |
| | Rank | 3 | 2 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 4.331E+01 | 6.548E+01 | 4.518E+01 | 6.791E+01 | 4.625E−13 | 7.766E+01 | 2.726E+01 | 4.110E+01 | 2.489E+01 |
| CF10 | Mean | 8.202E+03 | 5.223E+03 | 1.079E+04 | 1.100E+04 | 1.944E+04 | 1.613E+04 | 1.407E+04 | 9.042E+03 | 5.327E+03 |
| | Rank | 3 | 1 | 5 | 6 | 9 | 8 | 7 | 4 | 2 |
| | Std | 2.656E+03 | 6.964E+02 | 8.628E+02 | 8.507E+02 | 0.000E+00 | 6.464E+02 | 5.635E+02 | 1.201E+03 | 4.189E+02 |
| CF11 | Mean | 1.059E+04 | 8.421E+03 | 1.207E+04 | 1.270E+04 | 1.943E+04 | 1.524E+04 | 1.536E+04 | 1.101E+04 | 1.064E+04 |
| | Rank | 2 | 1 | 5 | 6 | 9 | 7 | 8 | 4 | 3 |
| | Std | 2.944E+03 | 1.315E+03 | 1.262E+03 | 1.170E+03 | 0.000E+00 | 7.524E+02 | 4.661E+02 | 1.107E+03 | 1.030E+03 |
| CF12 | Mean | 1.204E+03 | 1.203E+03 | 1.201E+03 | 1.203E+03 | 1.214E+03 | 1.204E+03 | 1.204E+03 | 1.202E+03 | 1.202E+03 |
| | Rank | 6 | 4 | 1 | 4 | 9 | 8 | 7 | 3 | 2 |
| | Std | 9.422E−01 | 5.334E−01 | 3.947E−01 | 5.828E−01 | 4.625E−13 | 8.554E−01 | 3.897E−01 | 5.636E−01 | 8.791E−01 |
| CF13 | Mean | 1.301E+03 | 1.301E+03 | 1.303E+03 | 1.301E+03 | 1.310E+03 | 1.309E+03 | 1.308E+03 | 1.301E+03 | 1.300E+03 |
| | Rank | 2 | 4 | 6 | 5 | 9 | 8 | 7 | 3 | 1 |
| | Std | 8.688E−02 | 5.924E−01 | 1.063E+00 | 9.442E−01 | 2.313E−13 | 5.278E−01 | 4.633E−01 | 4.716E−01 | 4.641E−02 |
| CF14 | Mean | 1.400E+03 | 1.420E+03 | 1.454E+03 | 1.427E+03 | 1.880E+03 | 1.800E+03 | 1.697E+03 | 1.415E+03 | 1.400E+03 |
| | Rank | 1 | 4 | 6 | 5 | 9 | 8 | 7 | 3 | 1 |
| | Std | 6.518E−02 | 2.109E+01 | 1.888E+01 | 8.787E+00 | 6.938E−13 | 4.419E+01 | 4.519E+01 | 6.921E+00 | 4.721E−02 |
| CF15 | Mean | 1.562E+03 | 3.649E+03 | 4.682E+04 | 3.265E+04 | 2.740E+07 | 1.167E+07 | 3.563E+06 | 4.114E+03 | 1.521E+03 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 9.470E+00 | 5.690E+03 | 5.434E+04 | 1.740E+04 | 0.000E+00 | 8.484E+06 | 1.737E+06 | 2.485E+03 | 2.240E+00 |
| CF16 | Mean | 1.622E+03 | 1.622E+03 | 1.622E+03 | 1.623E+03 | 1.625E+03 | 1.623E+03 | 1.623E+03 | 1.622E+03 | 1.621E+03 |
| | Rank | 3 | 2 | 4 | 6 | 9 | 7 | 8 | 4 | 1 |
| | Std | 4.090E−01 | 3.703E−01 | 4.024E−01 | 5.218E−01 | 9.250E−13 | 3.766E−01 | 2.369E−01 | 4.576E−01 | 6.361E−01 |
| CF17 | Mean | 3.791E+06 | 9.104E+06 | 3.113E+07 | 1.406E+08 | 3.878E+09 | 6.623E+08 | 5.174E+08 | 4.457E+07 | 5.845E+03 |
| | Rank | 2 | 3 | 4 | 6 | 9 | 8 | 7 | 5 | 1 |
| | Std | 1.708E+06 | 7.489E+06 | 2.367E+07 | 7.272E+07 | 1.940E−06 | 2.537E+08 | 3.061E+08 | 2.653E+07 | 1.472E+03 |
| CF18 | Mean | 3.285E+05 | 1.266E+08 | 1.233E+09 | 5.142E+07 | 3.821E+10 | 2.251E+10 | 1.322E+10 | 9.659E+07 | 2.099E+03 |
| | Rank | 2 | 5 | 6 | 3 | 9 | 8 | 7 | 4 | 1 |
| | Std | 6.679E+05 | 2.426E+08 | 4.424E+08 | 9.769E+07 | 7.760E−06 | 5.945E+09 | 4.031E+09 | 1.141E+08 | 3.963E+01 |
| CF19 | Mean | 1.972E+03 | 1.993E+03 | 2.137E+03 | 2.177E+03 | 1.083E+04 | 5.068E+03 | 4.347E+03 | 2.029E+03 | 1.921E+03 |
| | Rank | 2 | 3 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 2.826E+01 | 6.957E+01 | 4.415E+01 | 7.767E+01 | 0.000E+00 | 1.476E+03 | 1.028E+03 | 3.065E+01 | 7.976E+00 |
| CF20 | Mean | 3.793E+04 | 3.282E+04 | 1.296E+05 | 1.147E+06 | 3.218E+09 | 1.224E+06 | 2.421E+05 | 1.704E+05 | 2.182E+03 |
| | Rank | 3 | 2 | 4 | 7 | 9 | 8 | 6 | 5 | 1 |
| | Std | 1.110E+04 | 1.935E+04 | 3.810E+04 | 1.273E+06 | 0.000E+00 | 1.839E+06 | 1.681E+05 | 8.301E+04 | 3.084E+01 |
| CF21 | Mean | 3.015E+06 | 4.433E+06 | 1.056E+07 | 2.185E+07 | 1.867E+09 | 1.764E+08 | 9.714E+07 | 8.555E+06 | 4.768E+03 |
| | Rank | 2 | 3 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 8.871E+05 | 4.524E+06 | 8.042E+06 | 1.132E+07 | 7.275E−07 | 1.106E+08 | 4.560E+07 | 3.535E+06 | 7.825E+02 |

(*continued on next page*)

**Table 6** (*continued*).

| Function | Different algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | KO | PSO | GWO | WOA | SMA | AOA | RSA | AO | LSHADE |
| CF22 | Mean | 3.843E+03 | 3.480E+03 | 4.275E+03 | 4.678E+03 | 6.111E+06 | 3.491E+05 | 1.847E+05 | 4.161E+03 | 3.132E+03 |
| | Rank | 3 | 2 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 2.834E+02 | 3.241E+02 | 3.076E+02 | 6.442E+02 | 3.789E−09 | 4.049E+05 | 1.708E+05 | 3.789E+02 | 2.164E+02 |
| CF23 | Mean | 2.500E+03 | 2.681E+03 | 2.955E+03 | 2.828E+03 | 2.500E+03 | 2.500E+03 | 2.500E+03 | 2.500E+03 | 2.644E+03 |
| | Rank | 1 | 7 | 9 | 8 | 1 | 1 | 1 | 1 | 6 |
| | Std | 0.000E+00 | 3.676E+01 | 5.461E+01 | 1.185E+02 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 2.870E−04 |
| CF24 | Mean | 2.600E+03 | 2.687E+03 | 2.600E+03 | 2.606E+03 | 2.600E+03 | 2.600E+03 | 2.600E+03 | 2.600E+03 | 2.676E+03 |
| | Rank | 1 | 9 | 1 | 7 | 1 | 1 | 1 | 1 | 8 |
| | Std | 2.749E−05 | 9.808E+00 | 9.503E−04 | 2.004E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 6.895E−06 | 1.099E+00 |
| CF25 | Mean | 2.700E+03 | 2.720E+03 | 2.700E+03 | 2.705E+03 | 2.700E+03 | 2.700E+03 | 2.700E+03 | 2.700E+03 | 2.707E+03 |
| | Rank | 1 | 9 | 1 | 7 | 1 | 1 | 1 | 1 | 8 |
| | Std | 0.000E+00 | 6.366E+00 | 4.050E−13 | 1.734E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 6.315E−01 |
| CF26 | Mean | 2.771E+03 | 2.797E+03 | 2.787E+03 | 2.718E+03 | 2.800E+03 | 2.798E+03 | 2.800E+03 | 2.787E+03 | 2.700E+03 |
| | Rank | 3 | 6 | 4 | 2 | 8 | 7 | 8 | 4 | 1 |
| | Std | 4.417E+01 | 2.491E+01 | 3.334E+01 | 3.738E+01 | 0.000E+00 | 8.800E+00 | 0.000E+00 | 3.335E+01 | 7.223E−02 |
| CF27 | Mean | 2.900E+03 | 4.159E+03 | 4.244E+03 | 4.874E+03 | 2.900E+03 | 2.900E+03 | 2.900E+03 | 2.900E+03 | 3.148E+03 |
| | Rank | 1 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 6 |
| | Std | 1.388E−12 | 2.668E+02 | 1.821E+02 | 1.330E+02 | 1.388E−12 | 1.388E−12 | 1.388E−12 | 1.388E−12 | 5.865E+01 |
| CF28 | Mean | 3.000E+03 | 6.726E+03 | 9.192E+03 | 8.846E+03 | 3.000E+03 | 3.000E+03 | 3.000E+03 | 3.000E+03 | 3.997E+03 |
| | Rank | 1 | 7 | 9 | 8 | 1 | 1 | 1 | 1 | 6 |
| | Std | 1.388E−12 | 7.972E+02 | 5.818E+02 | 1.835E+03 | 1.388E−12 | 1.388E−12 | 1.388E−12 | 1.388E−12 | 4.484E+01 |
| CF29 | Mean | 3.100E+03 | 8.655E+07 | 1.288E+08 | 7.452E+07 | 3.100E+03 | 3.100E+03 | 3.100E+03 | 3.100E+03 | 7.645E+03 |
| | Rank | 1 | 8 | 9 | 7 | 1 | 1 | 1 | 1 | 6 |
| | Std | 0.000E+00 | 6.011E+07 | 1.484E+08 | 5.653E+07 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 1.137E+03 |
| CF30 | Mean | 2.150E+04 | 3.468E+05 | 1.136E+06 | 9.710E+05 | 3.200E+03 | 3.200E+03 | 3.200E+03 | 3.200E+03 | 1.280E+04 |
| | Rank | 6 | 7 | 9 | 8 | 1 | 1 | 1 | 1 | 5 |
| | Std | 3.118E+04 | 3.876E+05 | 1.302E+06 | 5.338E+05 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 8.639E+02 |
| Sum | | 68 | 116 | 146 | 164 | 212 | 187 | 168 | 102 | 67 |
| Mean rank | | 2.267 | 3.867 | 4.867 | 5.467 | 7.067 | 6.233 | 5.600 | 3.400 | 2.233 |
| Final rank | | 2 | 4 | 5 | 6 | 9 | 8 | 7 | 3 | 1 |

*Note: The symbol* $(E \pm x)$ *is equal* $(10^{\pm x})$.

**Table 7**
$p_{value}$ of Wilcoxon rank sum test between KO and the other algorithms for CEC2014 benchmark functions at $\alpha = 0.05$, $D = 50$.

| Function | Different algorithms | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KO vs | PSO | Sig. | GWO | Sig. | WOA | Sig. | SMA | Sig. | AOA | Sig. | RSA | Sig. | AO | Sig. | LSHADE | Sig. |
| CF1 | 1.78E−04 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF2 | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF3 | 9.00E−01 | ∼ | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF4 | 3.52E−07 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF5 | 8.88E−06 | − | 9.92E−11 | − | 3.47E−10 | − | 1.21E−12 | + | 4.46E−04 | + | 8.31E−03 | + | 5.19E−02 | ∼ | 8.35E−08 | − |
| CF6 | 7.39E−11 | − | 5.01E−02 | ∼ | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 1.17E−09 | + | 3.02E−11 | − |
| CF7 | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF8 | 1.96E−10 | − | 3.02E−11 | + | 5.49E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 1.41E−09 | + | 3.02E−11 | − |
| CF9 | 4.80E−07 | − | 2.67E−09 | + | 3.69E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.83E−06 | + | 3.02E−11 | − |
| CF10 | 1.01E−08 | − | 1.17E−05 | + | 7.74E−06 | + | 1.21E−12 | + | 3.02E−11 | + | 8.10E−10 | + | 1.08E−02 | + | 8.89E−10 | − |
| CF11 | 1.68E−03 | − | 1.03E−02 | + | 1.95E−03 | + | 1.21E−12 | + | 2.03E−07 | + | 7.12E−09 | + | 9.03E−04 | + | 1.60E−03 | + |
| CF12 | 9.21E−05 | − | 2.67E−09 | − | 1.33E−04 | − | 1.21E−12 | + | 1.38E−02 | + | 4.92E−01 | ∼ | 9.76E−10 | − | 9.76E−10 | − |
| CF13 | 3.82E−09 | + | 3.34E−11 | + | 3.99E−04 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 5.09E−08 | + | 3.69E−11 | − |
| CF14 | 5.00E−09 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 1.33E−10 | + | 6.52E−01 | ∼ |
| CF15 | 1.17E−04 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF16 | 1.76E−01 | ∼ | 2.42E−02 | + | 4.08E−05 | + | 1.21E−12 | + | 1.49E−06 | + | 2.23E−09 | + | 6.91E−04 | + | 1.86E−06 | − |
| CF17 | 2.13E−05 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.34E−11 | + | 3.02E−11 | − |
| CF18 | 7.73E−02 | ∼ | 3.02E−11 | + | 4.50E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.34E−11 | + | 3.02E−11 | − |
| CF19 | 7.96E−01 | ∼ | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 4.18E−09 | + | 1.70E−08 | − |
| CF20 | 1.91E−02 | − | 4.08E−11 | + | 3.34E−11 | + | 1.21E−12 | + | 3.34E−11 | + | 3.34E−11 | + | 6.12E−10 | + | 3.02E−11 | − |
| CF21 | 4.46E−01 | ∼ | 2.67E−09 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 2.39E−08 | + | 3.02E−11 | − |
| CF22 | 3.59E−05 | − | 2.00E−06 | + | 1.31E−08 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 1.68E−04 | + | 4.18E−09 | − |
| CF23 | 1.21E−12 | + | 1.21E−12 | + | 1.66E−11 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF24 | 3.02E−11 | + | 3.02E−11 | ∼ | 3.02E−11 | + | 4.57E−12 | ∼ | 4.57E−12 | ∼ | 1.21E−12 | ∼ | 1.87E−10 | ∼ | 3.02E−11 | + |
| CF25 | 1.21E−12 | + | 2.73E−03 | ∼ | 7.56E−07 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF26 | 1.22E−09 | + | 4.19E−06 | + | 2.20E−02 | − | 1.46E−04 | + | 5.90E−04 | + | 2.79E−03 | + | 1.79E−01 | ∼ | 2.01E−10 | − |
| CF27 | 1.21E−12 | + | 1.21E−12 | + | 1.21E−12 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF28 | 1.21E−12 | + | 1.21E−12 | + | 1.21E−12 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF29 | 1.21E−12 | + | 1.21E−12 | + | 1.21E−12 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF30 | 3.99E−10 | + | 9.40E−12 | + | 9.40E−12 | + | 2.79E−03 | − | 6.39E−05 | − | 3.13E−04 | − | 3.13E−04 | − | 2.47E−02 | − |
| (+) | | **16** | | **25** | | **27** | | **23** | | **23** | | **22** | | **20** | | **7** |
| (−) | | **9** | | **2** | | **3** | | **1** | | **1** | | **1** | | **2** | | **22** |
| (∼) | | **5** | | **3** | | **0** | | **6** | | **6** | | **7** | | **8** | | **1** |

**Table 8**
Comparison of CEC2014 benchmark functions for different optimization algorithms for $D = 100$.

| Function | Different algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | KO | PSO | GWO | WOA | SMA | AOA | RSA | AO | LSHADE |
| CF1 | Mean | 1.888E+08 | 3.989E+08 | 1.142E+09 | 1.316E+09 | 1.793E+10 | 8.954E+09 | 8.003E+09 | 6.313E+08 | 1.176E+07 |
| | Rank | 2 | 3 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 9.730E+06 | 8.346E+07 | 1.245E+08 | 8.570E+07 | 0.000E+00 | 2.086E+09 | 1.330E+09 | 1.293E+08 | 2.841E+06 |
| CF2 | Mean | 4.747E+08 | 2.902E+10 | 9.331E+10 | 7.414E+10 | 3.416E+11 | 3.153E+11 | 2.778E+11 | 5.126E+10 | 1.195E+05 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 5.152E+06 | 4.547E+09 | 6.512E+09 | 3.165E+09 | 3.104E−05 | 1.659E+10 | 1.210E+10 | 7.369E+09 | 5.277E+04 |
| CF3 | Mean | 2.248E+05 | 2.776E+05 | 3.057E+05 | 5.065E+05 | 2.401E+08 | 5.951E+05 | 3.002E+05 | 3.160E+05 | 2.440E+03 |
| | Rank | 2 | 3 | 5 | 7 | 9 | 8 | 4 | 6 | 1 |
| | S. Deviation | 1.143E+04 | 2.243E+04 | 1.281E+04 | 1.884E+04 | 3.637E−07 | 9.421E+05 | 1.487E+04 | 2.652E+04 | 6.320E+02 |
| CF4 | Mean | 1.063E+03 | 3.129E+03 | 1.046E+04 | 9.555E+03 | 1.421E+05 | 1.088E+05 | 8.146E+04 | 7.158E+03 | 6.368E+02 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 5.337E+01 | 2.763E+02 | 1.110E+03 | 4.896E+02 | 1.480E−11 | 8.911E+03 | 1.292E+04 | 1.600E+03 | 3.135E+01 |
| CF5 | Mean | 5.214E+02 | 5.213E+02 | 5.211E+02 | 5.212E+02 | 5.217E+02 | 5.213E+02 | 5.214E+02 | 5.214E+02 | 5.213E+02 |
| | Rank | 7 | 4 | 1 | 2 | 9 | 5 | 8 | 6 | 3 |
| | Std | 5.507E−02 | 6.447E−02 | 1.325E−01 | 8.033E−02 | 3.469E−13 | 7.806E−02 | 2.686E−02 | 5.406E−02 | 1.210E−01 |
| CF6 | Mean | 7.188E+02 | 6.889E+02 | 7.088E+02 | 7.572E+02 | 7.814E+02 | 7.596E+02 | 7.579E+02 | 7.352E+02 | 6.332E+02 |
| | Rank | 4 | 2 | 3 | 6 | 9 | 8 | 7 | 5 | 1 |
| | Std | 5.636E+00 | 3.644E+00 | 8.418E+00 | 4.230E+00 | 0.000E+00 | 3.294E+00 | 3.073E+00 | 6.327E+00 | 3.319E+00 |
| CF7 | Mean | 7.048E+02 | 1.028E+03 | 1.579E+03 | 1.401E+03 | 4.167E+03 | 3.900E+03 | 3.505E+03 | 1.244E+03 | 7.001E+02 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 4.186E−02 | 4.864E+01 | 5.102E+01 | 4.092E+01 | 2.313E−12 | 1.940E+02 | 9.742E+01 | 5.837E+01 | 4.895E−02 |
| CF8 | Mean | 1.448E+03 | 1.344E+03 | 1.876E+03 | 1.850E+03 | 2.567E+03 | 2.466E+03 | 2.236E+03 | 1.614E+03 | 1.203E+03 |
| | Rank | 3 | 2 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 3.824E+01 | 2.651E+01 | 3.011E+01 | 6.176E+01 | 0.000E+00 | 4.265E+01 | 3.249E+01 | 5.064E+01 | 1.656E+01 |
| CF9 | Mean | 1.696E+03 | 1.693E+03 | 1.805E+03 | 2.103E+03 | 2.601E+03 | 2.505E+03 | 2.351E+03 | 1.898E+03 | 1.429E+03 |
| | Rank | 3 | 2 | 4 | 6 | 9 | 8 | 7 | 5 | 1 |
| | Std | 4.331E+01 | 6.548E+01 | 4.518E+01 | 6.791E+01 | 4.625E−13 | 7.766E+01 | 3.373E+01 | 5.780E+01 | 3.714E+01 |
| CF10 | Mean | 1.938E+04 | 1.415E+04 | 2.431E+04 | 2.546E+04 | 3.761E+04 | 3.332E+04 | 3.078E+04 | 2.119E+04 | 1.715E+04 |
| | Rank | 3 | 1 | 5 | 6 | 9 | 8 | 7 | 4 | 2 |
| | Std | 2.656E+03 | 6.964E+02 | 8.628E+02 | 8.507E+02 | 0.000E+00 | 6.464E+02 | 9.559E+02 | 1.865E+03 | 6.572E+02 |
| CF11 | Mean | 1.904E+04 | 2.507E+04 | 2.468E+04 | 2.808E+04 | 3.775E+04 | 3.136E+04 | 3.111E+04 | 2.252E+04 | 2.566E+04 |
| | Rank | 1 | 4 | 3 | 6 | 9 | 8 | 7 | 2 | 5 |
| | Std | 2.944E+03 | 1.315E+03 | 1.262E+03 | 1.170E+03 | 0.000E+00 | 7.524E+02 | 9.070E+02 | 1.898E+03 | 1.689E+03 |
| CF12 | Mean | 1.205E+03 | 1.204E+03 | 1.201E+03 | 1.204E+03 | 1.214E+03 | 1.205E+03 | 1.205E+03 | 1.203E+03 | 1.203E+03 |
| | Rank | 6 | 4 | 1 | 4 | 9 | 7 | 8 | 3 | 2 |
| | Std | 9.422E−01 | 5.334E−01 | 3.947E−01 | 5.828E−01 | 4.625E−13 | 8.554E−01 | 2.798E−01 | 6.553E−01 | 1.045E+00 |
| CF13 | Mean | 1.301E+03 | 1.302E+03 | 1.305E+03 | 1.304E+03 | 1.310E+03 | 1.310E+03 | 1.309E+03 | 1.304E+03 | 1.300E+03 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 8.688E−02 | 5.924E−01 | 1.063E+00 | 9.442E−01 | 2.313E−13 | 5.278E−01 | 2.320E−01 | 3.676E−01 | 7.047E−02 |
| CF14 | Mean | 1.401E+03 | 1.498E+03 | 1.644E+03 | 1.581E+03 | 2.442E+03 | 2.338E+03 | 2.238E+03 | 1.540E+03 | 1.400E+03 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 6.518E−02 | 2.109E+01 | 1.888E+01 | 8.787E+00 | 6.938E−13 | 4.419E+01 | 4.681E+01 | 2.122E+01 | 4.682E−02 |
| CF15 | Mean | 1.748E+03 | 3.570E+04 | 3.546E+05 | 4.333E+05 | 6.099E+07 | 3.608E+07 | 1.511E+07 | 8.122E+04 | 1.556E+03 |
| | Rank | 2 | 3 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 9.470E+00 | 5.690E+03 | 5.434E+04 | 1.740E+04 | 0.000E+00 | 8.484E+06 | 4.077E+06 | 4.349E+04 | 4.304E+00 |
| CF16 | Mean | 1.646E+03 | 1.646E+03 | 1.646E+03 | 1.647E+03 | 1.650E+03 | 1.647E+03 | 1.647E+03 | 1.647E+03 | 1.646E+03 |
| | Rank | 4 | 2 | 2 | 6 | 9 | 7 | 8 | 5 | 1 |
| | Std | 4.090E−01 | 3.703E−01 | 4.024E−01 | 5.218E−01 | 9.250E−13 | 3.766E−01 | 2.864E−01 | 5.844E−01 | 1.059E+00 |
| CF17 | Mean | 3.245E+07 | 4.195E+07 | 1.541E+08 | 1.303E+08 | 3.590E+09 | 1.456E+09 | 1.271E+09 | 1.402E+08 | 8.937E+04 |
| | Rank | 2 | 3 | 6 | 4 | 9 | 8 | 7 | 5 | 1 |
| | Std | 1.708E+06 | 7.489E+06 | 2.367E+07 | 7.272E+07 | 1.940E−06 | 2.537E+08 | 4.473E+08 | 4.898E+07 | 3.372E+04 |
| CF18 | Mean | 5.053E+05 | 7.762E+08 | 3.228E+09 | 5.965E+08 | 5.828E+10 | 4.515E+10 | 3.508E+10 | 3.047E+08 | 2.610E+03 |
| | Rank | 2 | 5 | 6 | 4 | 9 | 8 | 7 | 3 | 1 |
| | Std | 6.679E+05 | 2.426E+08 | 4.424E+08 | 9.769E+07 | 7.760E−06 | 5.945E+09 | 7.902E+09 | 5.742E+08 | 1.136E+02 |
| CF19 | Mean | 2.094E+03 | 2.401E+03 | 2.733E+03 | 2.724E+03 | 1.788E+04 | 1.151E+04 | 9.749E+03 | 2.610E+03 | 2.003E+03 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 2.826E+01 | 6.957E+01 | 4.415E+01 | 7.767E+01 | 0.000E+00 | 1.476E+03 | 1.652E+03 | 1.607E+02 | 3.044E+00 |
| CF20 | Mean | 1.474E+05 | 1.541E+05 | 5.992E+05 | 5.105E+05 | 1.313E+09 | 1.417E+06 | 7.932E+05 | 4.119E+05 | 2.527E+03 |
| | Rank | 2 | 3 | 6 | 5 | 9 | 8 | 7 | 4 | 1 |
| | Std | 1.110E+04 | 1.935E+04 | 3.810E+04 | 1.273E+06 | 0.000E+00 | 1.839E+06 | 1.875E+05 | 1.240E+05 | 7.105E+01 |
| CF21 | Mean | 1.213E+07 | 1.510E+07 | 5.253E+07 | 1.074E+08 | 1.409E+09 | 6.258E+08 | 4.318E+08 | 3.877E+07 | 2.168E+04 |
| | Rank | 2 | 3 | 5 | 6 | 9 | 8 | 7 | 4 | 1 |
| | Std | 8.871E+05 | 4.524E+06 | 8.042E+06 | 1.132E+07 | 7.275E−07 | 1.106E+08 | 8.439E+07 | 1.420E+07 | 5.768E+03 |

(*continued on next page*)

**Table 8** (*continued*).

| Function | Different algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | KO | PSO | GWO | WOA | SMA | AOA | RSA | AO | LSHADE |
| CF22 | Mean 5.708E+03 | 4.931E+03 | 6.410E+03 | 8.177E+03 | 2.169E+06 | 2.889E+05 | 1.533E+05 | 6.369E+03 | 5.267E+03 |
| | Rank 3 | 1 | 5 | 6 | 9 | 8 | 7 | 4 | 2 |
| | Std 2.834E+02 | 3.241E+02 | 3.076E+02 | 6.442E+02 | 3.789E−09 | 4.049E+05 | 1.011E+05 | 7.109E+02 | 3.783E+02 |
| CF23 | Mean 2.500E+03 | 2.800E+03 | 3.127E+03 | 2.573E+03 | 2.500E+03 | 2.500E+03 | 2.500E+03 | 2.500E+03 | 2.649E+03 |
| | Rank 1 | 8 | 9 | 6 | 1 | 1 | 1 | 1 | 7 |
| | Std 0.000E+00 | 3.676E+01 | 5.461E+01 | 1.185E+02 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 1.582E−01 |
| CF24 | Mean 2.600E+03 | 2.884E+03 | 2.600E+03 | 2.601E+03 | 2.600E+03 | 2.600E+03 | 2.600E+03 | 2.600E+03 | 2.795E+03 |
| | Rank 1 | 9 | 1 | 7 | 1 | 1 | 1 | 1 | 8 |
| | Std 2.749E−05 | 9.808E+00 | 9.503E−04 | 2.004E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 1.933E−05 | 2.932E+00 |
| CF25 | Mean 2.700E+03 | 2.800E+03 | 2.700E+03 | 2.700E+03 | 2.700E+03 | 2.700E+03 | 2.700E+03 | 2.700E+03 | 2.703E+03 |
| | Rank 1 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| | Std 0.000E+00 | 6.366E+00 | 4.050E−13 | 1.734E+01 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 4.292E+00 |
| CF26 | Mean 2.800E+03 | 2.817E+03 | 2.800E+03 | 2.800E+03 | 2.800E+03 | 2.800E+03 | 2.800E+03 | 2.800E+03 | 2.801E+03 |
| | Rank 1 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| | Std 4.417E+01 | 2.491E+01 | 3.334E+01 | 3.738E+01 | 0.000E+00 | 8.800E+00 | 0.000E+00 | 0.000E+00 | 1.650E−02 |
| CF27 | Mean 2.900E+03 | 5.931E+03 | 6.207E+03 | 7.332E+03 | 2.900E+03 | 2.900E+03 | 2.900E+03 | 2.900E+03 | 3.367E+03 |
| | Rank 1 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 6 |
| | Std 1.388E−12 | 2.668E+02 | 1.821E+02 | 1.330E+02 | 1.388E−12 | 1.388E−12 | 2.313E−12 | 2.313E−12 | 4.573E+01 |
| CF28 | Mean 3.000E+03 | 1.118E+04 | 1.521E+04 | 2.008E+04 | 3.000E+03 | 3.000E+03 | 3.000E+03 | 3.000E+03 | 5.257E+03 |
| | Rank 1 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 6 |
| | Std 1.388E−12 | 7.972E+02 | 5.818E+02 | 1.835E+03 | 1.388E−12 | 1.388E−12 | 2.313E−12 | 2.313E−12 | 5.958E+01 |
| CF29 | Mean 3.100E+03 | 5.595E+08 | 1.191E+09 | 5.595E+08 | 3.100E+03 | 3.100E+03 | 3.100E+03 | 3.100E+03 | 1.294E+04 |
| | Rank 1 | 7 | 9 | 8 | 1 | 1 | 1 | 1 | 6 |
| | Std 0.000E+00 | 6.011E+07 | 1.484E+08 | 5.653E+07 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 1.392E+03 |
| CF30 | Mean 1.249E+05 | 2.330E+06 | 1.715E+07 | 2.173E+07 | 3.200E+03 | 3.200E+03 | 3.200E+03 | 3.200E+03 | 1.493E+04 |
| | Rank 6 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 5 |
| | Std 3.118E+04 | 3.876E+05 | 1.302E+06 | 5.338E+05 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 0.000E+00 | 1.325E+03 |
| Sum | 67 | 119 | 141 | 156 | 205 | 178 | 161 | 99 | 80 |
| Mean rank | 2.233 | 3.967 | 4.700 | 5.200 | 6.833 | 5.933 | 5.367 | 3.300 | 2.667 |
| Final rank | 1 | 4 | 5 | 6 | 9 | 8 | 7 | 3 | 2 |

*Note: The symbol* $(E \pm x)$ *is equal* $(10^{\pm x})$.

**Table 9**
$p_{value}$ of Wilcoxon rank sum test between KO and the other algorithms for CEC2014 benchmark functions at $\alpha = 0.05$ for $D = 100$.

| Function | Different algorithms | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KO vs | PSO | Sig. | GWO | Sig. | WOA | Sig. | SMA | Sig. | AOA | Sig. | RSA | Sig. | AO | Sig. | LSHADE | Sig. |
| CF1 | 1.07E−07 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.69E−11 | − |
| CF2 | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF3 | 3.83E−06 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF4 | 1.46E−10 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF5 | 5.97E−09 | − | 3.02E−11 | − | 3.34E−11 | − | 1.21E−12 | + | 4.08E−05 | − | 8.53E−01 | ∼ | 4.86E−03 | − | 7.70E−08 | − |
| CF6 | 3.69E−11 | − | 1.33E−04 | − | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.47E−10 | + | 3.02E−11 | − |
| CF7 | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF8 | 1.75E−05 | − | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 2.02E−08 | + | 3.02E−11 | − |
| CF9 | 5.01E−01 | ∼ | 1.39E−06 | + | 3.02E−11 | + | 1.21E−12 | + | 3.01E−11 | + | 3.02E−11 | + | 3.34E−11 | + | 3.34E−11 | − |
| CF10 | 1.33E−10 | − | 2.19E−08 | + | 1.56E−08 | + | 1.21E−12 | + | 3.02E−11 | + | 1.09E−10 | + | 2.88E−06 | + | 2.01E−01 | ∼ |
| CF11 | 7.12E−09 | + | 1.29E−09 | + | 5.57E−10 | + | 1.21E−12 | + | 3.16E−10 | + | 5.57E−10 | + | 4.18E−09 | + | 5.57E−10 | + |
| CF12 | 1.34E−05 | − | 3.02E−11 | − | 3.57E−06 | − | 1.21E−12 | + | 5.79E−01 | ∼ | 3.03E−03 | + | 7.70E−08 | − | 6.52E−09 | − |
| CF13 | 1.46E−10 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 7.39E−11 | − |
| CF14 | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 9.83E−08 | − |
| CF15 | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF16 | 4.20E−01 | ∼ | 4.04E−01 | ∼ | 6.57E−02 | ∼ | 1.21E−12 | + | 1.68E−04 | + | 3.32E−06 | + | 1.58E−01 | ∼ | 9.21E−05 | − |
| CF17 | 1.03E−02 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF18 | 1.16E−07 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF19 | 4.18E−09 | + | 3.02E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF20 | 6.95E−01 | ∼ | 3.02E−11 | + | 3.69E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | + | 3.02E−11 | − |
| CF21 | 3.04E−01 | ∼ | 4.08E−11 | + | 3.02E−11 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 3.69E−11 | + | 3.02E−11 | − |
| CF22 | 6.77E−05 | − | 8.15E−05 | + | 5.07E−10 | + | 1.21E−12 | + | 3.02E−11 | + | 3.02E−11 | + | 8.15E−05 | + | 5.26E−04 | − |
| CF23 | 1.21E−12 | + | 1.21E−12 | + | 4.06E−08 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF24 | 3.02E−11 | + | 3.02E−11 | ∼ | 3.02E−11 | + | 1.66E−11 | ∼ | 1.66E−11 | ∼ | 4.57E−12 | ∼ | 8.37E−09 | ∼ | 3.02E−11 | + |
| CF25 | 1.21E−12 | + | 6.37E−13 | ∼ | 1.14E−05 | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF26 | 1.21E−12 | + | 4.64E−13 | ∼ | 1.38E−04 | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF27 | 1.21E−12 | + | 1.21E−12 | + | 1.21E−12 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF28 | 1.21E−12 | + | 1.21E−12 | + | 1.21E−12 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF29 | 1.21E−12 | + | 1.21E−12 | + | 1.21E−12 | + | NaN | ∼ | NaN | ∼ | NaN | ∼ | NaN | ∼ | 1.21E−12 | + |
| CF30 | 4.33E−10 | + | 3.16E−12 | + | 3.16E−12 | + | 8.15E−02 | ∼ | 8.15E−02 | ∼ | 4.19E−02 | − | 4.19E−02 | − | 3.73E−07 | − |
| (+) | **20** | | **23** | | **25** | | **22** | | **20** | | **21** | | **19** | | **8** | |
| (−) | **6** | | **3** | | **2** | | **0** | | **1** | | **1** | | **3** | | **21** | |
| (∼) | **4** | | **4** | | **3** | | **8** | | **9** | | **8** | | **8** | | **1** | |

**Table 10**
The best result for tension/compression spring design using KO.

| Design variables | $x_1$ 0.051696456046163 | $x_2$ 0.356893641428733 | $x_3$ 11.279414562853750 | |
|---|---|---|---|---|
| The value of constrained functions | $g_1(x)$ −6.0325E−05 | $g_2(x)$ −4.5715E−06 | $g_3(x)$ −4.0466E+00 | $g_4(x)$ −7.2761E−01 |
| The best value | 0.012665994155756 | | | |
| The mean value | 0.012917291568222 | | | |
| The standard deviation value | 0.000301392430207736 | | | |

**Table 11**
The results of spring design problem by different algorithms.

| Different algorithms | The best design variables | | | The best value fitness |
|---|---|---|---|---|
| | $x_1$ | $x_2$ | $x_3$ | $f(x)$ |
| PSO [50] | 0.05172800 | 0.35764400 | 11.24454300 | 0.01267470 |
| ES [51] | 0.355360 | 0.051643 | 11.397926 | 0.012698 |
| GSA [43] | 0.050276 | 0.323680 | 13.525410 | 0.012702 |
| GA [52] | 0.05148000 | 0.35166100 | 11.63220100 | 0.01270480 |
| GWO [5] | 0.05169 | 0.356737 | 11.28885 | 0.0126660 |
| RO [53] | 0.051370 | 0.349096 | 11.762790 | 0.012679 |
| CSA [47] | 0.05168903 | 0.35671695 | 11.28901180 | 0.01266523 |
| SCA [54] | 0.052160 | 0.368159 | 10.648442 | 0.012669 |
| WOA [32] | 0.051207 | 0.345215 | 12.004032 | 0.012676 |
| DE [55] | 0.05160900 | 0.35471400 | 11.41083100 | 0.01267020 |
| HS [56] | 0.051154 | 0.349871 | 12.076432 | 0.012671 |
| SSA [57] | 0.051207 | 0.345215 | 12.004032 | 0.0126763 |
| EO [49] | 0.05161991 | 0.35505438 | 11.3879676 | 0.012666 |
| MPA [43] | 0.05172448 | 0.35757003 | 11.2391955 | 0.012665 |
| AVOA [33] | 0.05166983 | 0.35625535 | 11.316126 | 0.01266524 |
| SO [58] | 0.0511 | 0.3418 | 12.2222 | 0.01267254 |
| HHO [48] | 0.05179639 | 0.35930536 | 11.138859 | 0.01266544 |
| Constraint correction [59] | 0.05000000 | 0.31590000 | 14.25000000 | 0.01283340 |

**Table 12**
The best result for pressure vessel design using KO.

| Design variables | $x_1$ 0.8125 | $x_2$ 0.4375 | $x_3$ 42.0984449696433 | $x_4$ 176.636618401874 |
|---|---|---|---|---|
| The value of constrained functions | $g_1(x)$ −1.2086E−08 | $g_2(x)$ −3.5881E−02 | $g_3(x)$ −8.2401E−02 | $g_4(x)$ −6.3363E+01 |
| The best value | 6059.71475731827 | | | |
| The mean value | 6059.72453197228 | | | |
| The standard deviation value | 0.00594200957704173 | | | |

**Table 13**
The results of spring design problem by different algorithms.

| Different algorithms | The best design variables | | | | The best fitness |
|---|---|---|---|---|---|
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $f(x)$ |
| PSO [50] | 0.812500 | 0.437500 | 42.091266 | 176.7465 | 6061.0777 |
| ES [51] | 0.812500 | 0.437500 | 42.098087 | 176.640518 | 6059.745605 |
| HS [60] | 1.125 | 0.625 | 58.2789 | 43.7549 | 7198.433 |
| GSA [43] | 1.1250000 | 0.625000 | 55.9886598 | 84 0.4542025 | 85380.8359 |
| GA [52] | 0.812500 | 0.437500 | 40.3239 | 200.000000 | 6288.7445 |
| GWO [5] | 0.812500 | 0.434500 | 42.089181 | 176.758731 | 6051.5639 |
| ABC [61] | 0.812500 | 0.437500 | 42.098446 | 176.636596 | 6059.714339 |
| CSA [47] | 0.812500 | 0.437500 | 42.09844539 | 176.6365986 | 6059.714363 |
| WOA [32] | 0.812500 | 0.437500 | 42.0982699 | 176.638998 | 6059.7410 |
| DE [55] | 0.812500 | 0.437500 | 42.098411 | 176.637690 | 6059.7340 |
| ACO [62] | 0.812500 | 0.437500 | 42.103624 | 176.572656 | 6059.0888 |
| MVO [63] | 0.8125 | 0.4375 | 42.090738 | 176.73869 | 6060.8066 |
| EO [49] Mixed variable | 0.8125 | 0.4375 | 42.0984456 | 176.6365958 | 6059.7143 |
| MPA [43] Mixed variable | 0.8125 | 0.4375 | 42.098445 | 176.636607 | 6059.7144 |

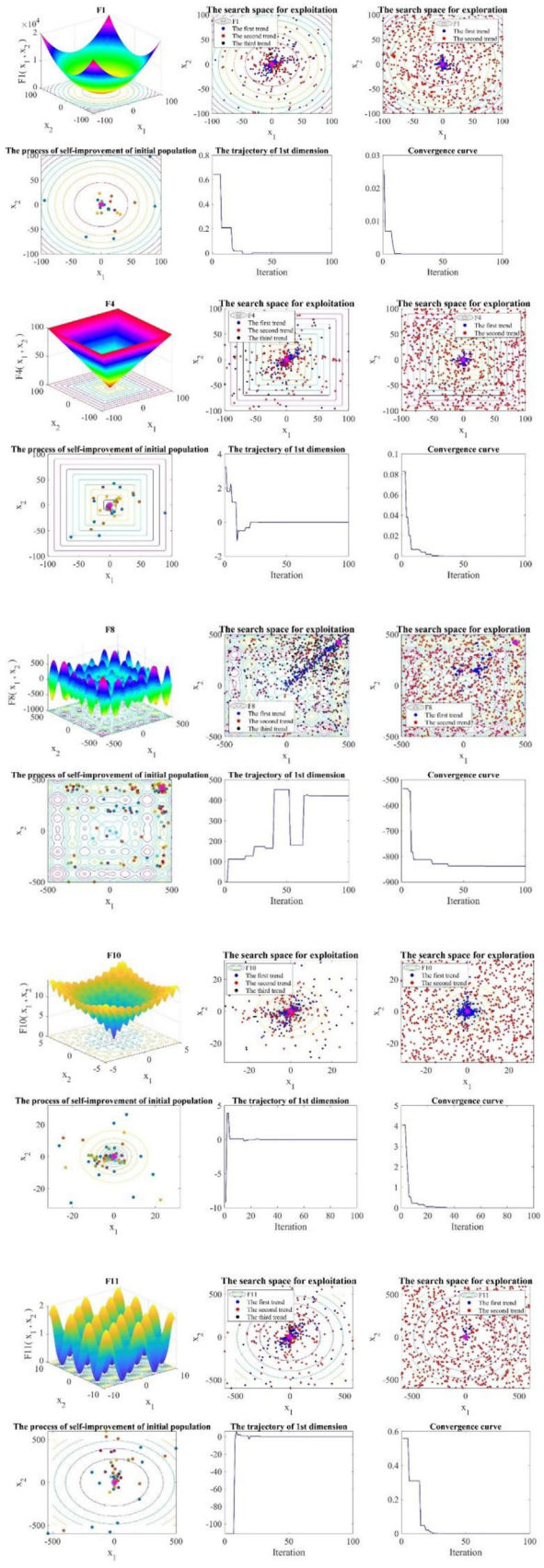


**Fig. 13.** Quantitative results of KO for the performance of the ability of exploration, exploitation, self-improvement of the initial population, the trajectory, and the convergence rate for typical functions.

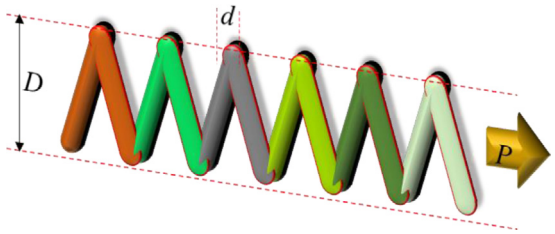


**Fig. 14.** Tension/compression spring design problem.

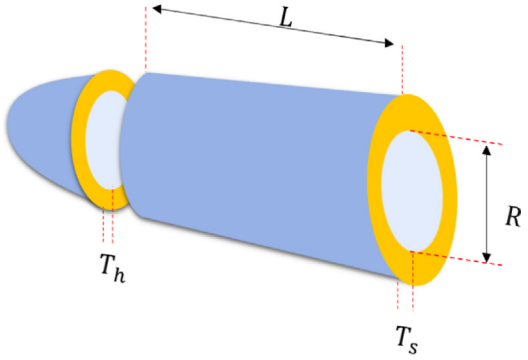


**Fig. 15.** Pressure vessel design problem.

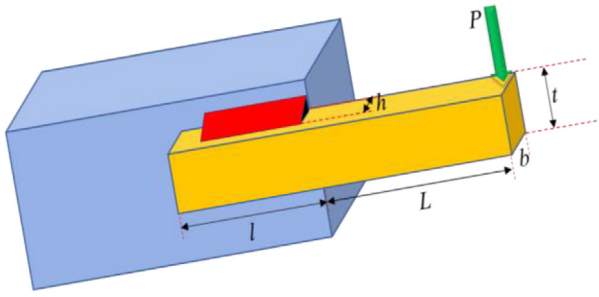


**Fig. 16.** Welded beam design problem.

$5 \le 0$, $g_5(x) = 0.125 - x_1 \le 0$, $g_6(x) = \delta(x) - \delta_{\max} \le 0$, $g_7(x) = P - P_c(x) \le 0$

where

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\ \tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},$$

$$M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},\ J = 2\left[\sqrt{2}x_1x_2\left\{\frac{x_2^2}{4}\right\} + \left(\frac{x_1 + x_3}{2}\right)^2\right],$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2},\ \delta(x) = \frac{4PL}{Ex_3^3x_4};$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\ P = 6000\,\text{lb},$$

$L = 14\,\text{in},\ E = 30 \times 10^6\,\text{psi},\ G = 12 \times 10^6\,\text{psi}$

$\tau_{\max} = 13600\,\text{psi},\ \sigma_{\max} = 30000\,\text{psi},\ \delta_{\max} = 0.25\,\text{in}$

$0.1 \le x_1 \le 2.0 \quad 0.1 \le x_2, x_3 \le 10 \quad 0.1 \le x_4 \le 2.0$

It is obvious that the results obtained by KO as shown in Table 14 with $f(x) = 1.725344871759735$. It can be concluded

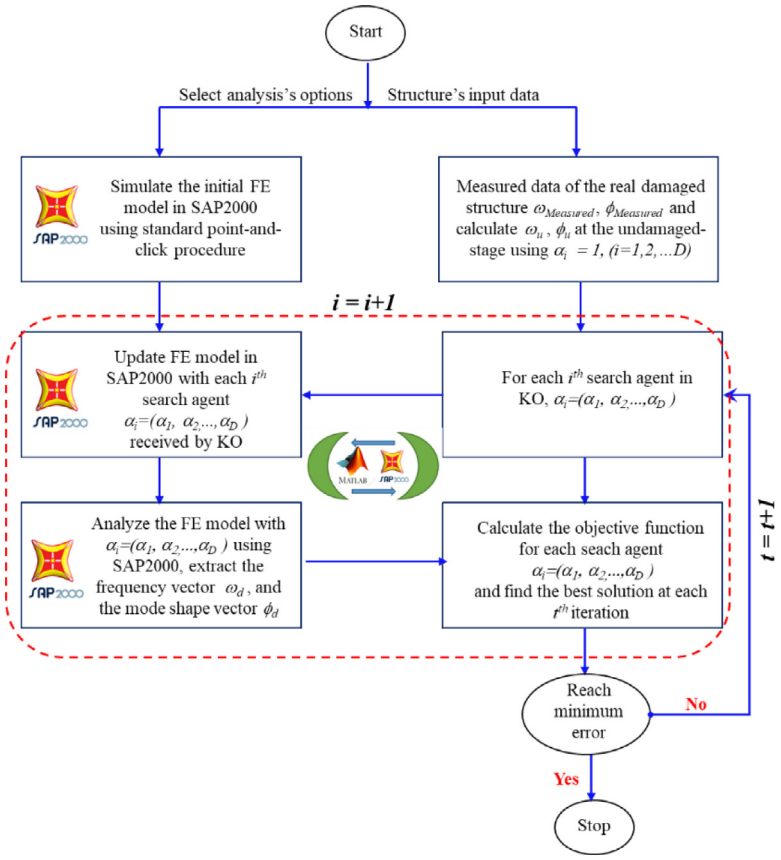


**Fig. 17.** The process for structural damage identification use KO combined with SAP-2000 through OAPI library.

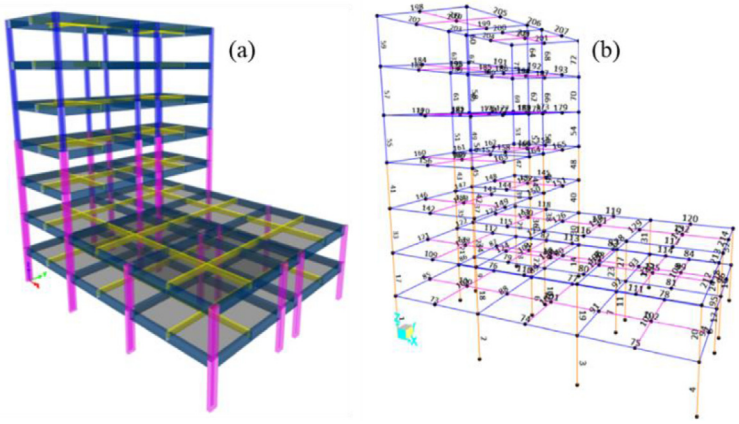


**Fig. 18.** (a) The 3D FE model of structure in SAP2000 and (b) the element numbers.

that KO can exploit the best possible optimal solution for solving the welded beam, the results obtained using KO is a little bit lower than that of MPA [43], AVOA [33], SSA [57], EO [49] as given in Table 15. Similar to the two examples' trends, KO has performed better than the rest of other algorithms. It can be concluded that KO can exploit the best possible optimal solution for solving the welded beam. The value of constrained functions calculated by KO can reach $-2.58 \times 10^{-4}$ as given in $g_3(x)$ function. Thus, KO has a good chance of success in this problem.

**Table 14**
The best result for welded beam design using KO.

| Design variables | $x_1$ 0.205465996226745 | $x_2$ 3.475892988253460 | $x_3$ 9.037870569365802 | $x_4$ 0.205724000148857 |
|---|---|---|---|---|
| The value of constrained functions | $g_1(x)$ −6.2996E−01 | $g_2(x)$ −7.4535E+00 | $g_3(x)$ −2.5800E−04 | $g_4(x)$ −3.2747E+00 |
| | $g_5(x)$ −8.0466E−02 | $g_6(x)$ −2.3555E−01 | $g_7(x)$ −5.0699E−02 | |
| The best fitness | 1.725344871759735 | | | |
| The mean value | 1.757279329313681 | | | |
| The standard deviation value | 0.029124629426684 | | | |

**Table 15**
The results of welded beam design by different algorithms.

| Different algorithms | The best design variables | | | | The best fitness |
|---|---|---|---|---|---|
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $f(x)$ |
| HS [60] | 0.2442 | 6.2231 | 8.2915 | 0.2400 | 2.3807 |
| GA [64] | 0.205986 | 3.471328 | 9.020224 | 0.20648 | 1.728226 |
| GSA [43] | 0.182129 | 3.856979 | 10.0000 | 0.202376 | 1.879952 |
| SA [65] | 0.2471 | 6.1451 | 8.2721 | 0.2495 | 2.4148 |
| DE [55] | 0.203137 | 3.542998 | 9.033498 | 0.206179 | 1.733462 |
| PSO [50] | 0.202369 | 3.544214 | 9.04821 | 0.205723 | 1.728024 |
| GWO [5] | 0.205676 | 3.478377 | 9.03681 | 0.205778 | 1.72624 |
| MVO [63] | 0.205463 | 3.473193 | 9.044502 | 0.205695 | 1.72645 |
| IPSO [66] | 0.2444 | 6.2175 | 8.2915 | 0.2444 | 2.3810 |
| RO [53] | 0.203687 | 3.528467 | 9.004233 | 0.207241 | 1.735344 |
| HS [56] | 0.2442 | 6.2231 | 8.2915 | 0.2400 | 2.3807 |
| DE [55] | 0.203137 | 3.542998 | 9.033498 | 0.206179 | 1.733462 |
| EO [49] | 0.2057 | 3.4705 | 9.03664 | 0.2057 | 1.7249 |
| HHO [48] | 0.204039 | 3.531061 | 9.027463 | 0.206147 | 1.731991 |
| SSA [57] | 0.2057 | 3.4714 | 9.0366 | 0.2057 | 1.72491 |
| MPA [43] | 0.205728 | 3.470509 | 9.036624 | 0.20573 | 1.724853 |
| AVOA [33] | 0.20573 | 3.470474 | 9.036621 | 0.20573 | 1.724852 |

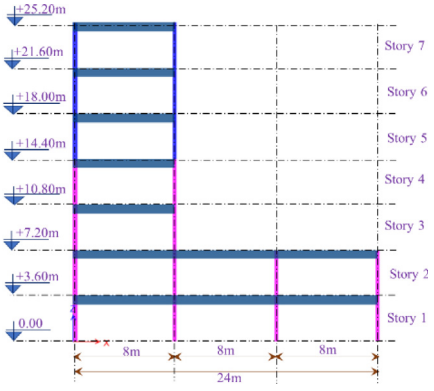


**Fig. 19.** Dimensions of the left-side view.

## 7. Application for structural Damage Identification in 3*D* complex concrete structures

Structural health monitoring (SHM) has gotten a lot of attention and research in the last two decades because of its importance in guaranteeing building safety. Structures will decrease in strength and will be damaged as the service life increases. It will be easier to maintain and repair structures if we can detect structural problems early on. SDI (structural damage identification) plays a key role in recent research [67–69]. A typical technique called vibration-based damage assessment has been strongly performed in SDI. The background of this method is based on the changes in the dynamic characteristics such as frequencies and mode shapes at two stages, i.e. damaged structure and undamaged structure [70]. Thus, based on these changes, an objective function is established. The objective function considers the change of frequencies or mode shapes, or both of them. This objective function is responsible for evaluating the correlation of the dynamic characteristics of the structure at two stages; the damaged stage and the undamaged stage. In other words, when the data at both stages is available, changes in the dynamic characteristics of the structures can be used to detect the damage location and severity.

### 7.1. The objective function

The first step toward solving a real-optimization problem is to determine the objective function. Then, an algorithm will be used to optimize this objective function. We assume that the structures have a total of $D$ elements. By converting to a vector with $D$ dimension, we will have a vector representing the damage severity of the structures denoted by $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_D)$, where $\alpha_i$ $(i = 1, 2, \ldots, D)$ represents the reduction stiffness in each element in the structures. And the value of this vector varies from zero to one. The value of one indicates a complete damage state, whereas a value of zero indicate an intact state. The process of finding the damaged vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_D)$, which makes the objective function reaching an acceptable error, is the main goal of SDI. As mentioned in the previous section, the mode shapes and the frequencies are two main parameters used for the objective function. In the first step, we must determine the frequencies and the mode shapes of the undamaged state of the structure. This is achieved by assigning the damage severities $\alpha_i = 1$, $(i = 1, 2, \ldots, D)$ to elements. And the frequencies and mode shapes are extracted using a linear equation of undammed free vibration, as given in Eq. (26).

$$\left[K_u - \omega_u^2 M\right] \phi_u = \{0\} \tag{26}$$

Where $K_u$ and $M$ are the global stiffness matrix and global mass matrix, respectively. The number of rows and columns in these matrices depends on the number of degrees of freedom of nodes

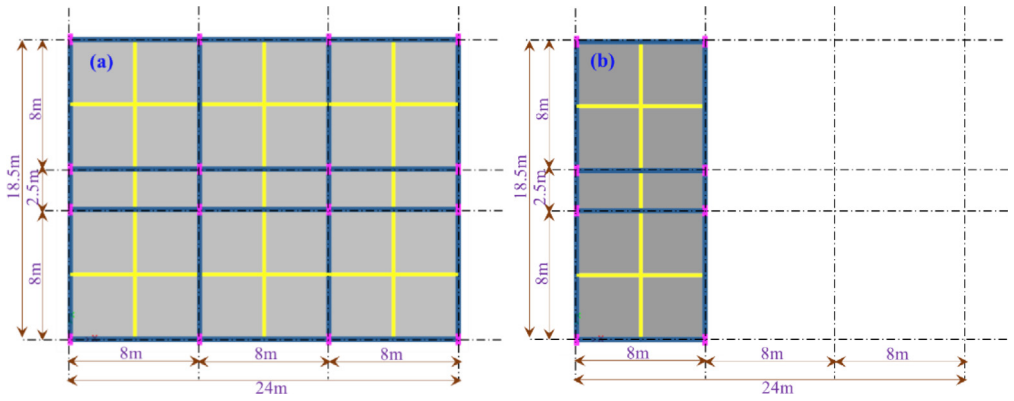


**Fig. 20.** Structural layout plan: (a) at level of +3.60 m and +7.20 m, and (b) at level of +10.80 m, +14.40 m, +18.00 m, +21.60 m, +25.20 m.

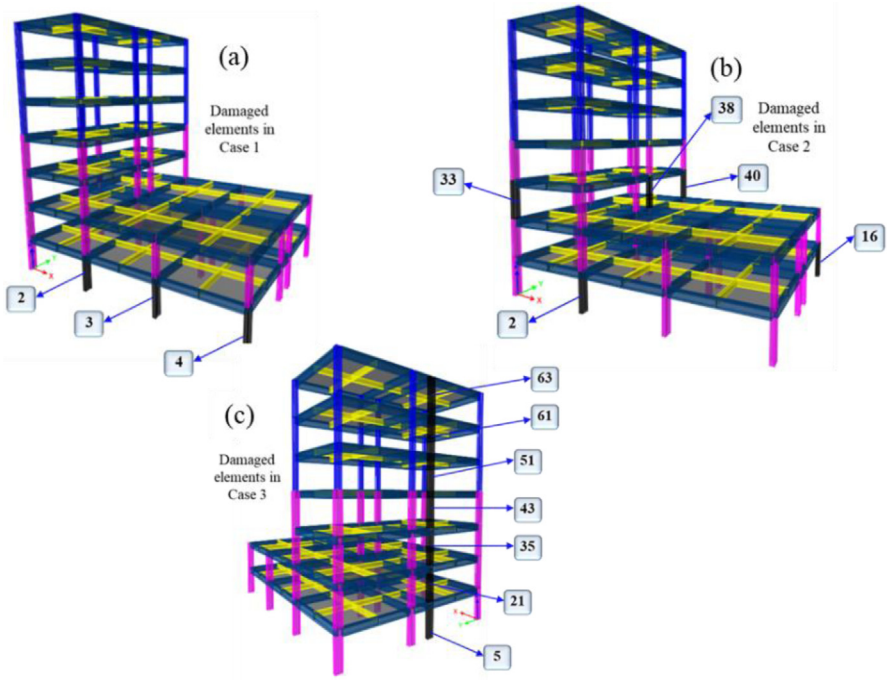


**Fig. 21.** The location of damaged elements; (a) case 1, (b) case 2 and (c) case 3.

in the structure. And $\omega_u$ and $\phi_u$ are the natural frequencies and the mode shapes for the undamaged stage, respectively.

At the state of damage $0 < \alpha_i,\ (i = 1, 2, \ldots, D) < 1$, the reduction in stiffness for each element of the structures is given by Eq. (27).

$$\begin{aligned} [k_e]_d &= (1 - \alpha_i)\,[k_e]_u \\ [k_e]_d &= (\alpha_r)\,[k_e]_u \end{aligned} \tag{27}$$

Where $[k_e]_d$ and $[k_e]_u$ are the stiffness matrix in $e$th element at damaged stage, and undamaged stage, respectively. And $(\alpha_r)$ is the remaining stiffness of $e$th element. Due to the damaged elements, the dynamic characteristics including natural frequencies and mode shapes are changed. And the new natural frequencies and mode shapes will also extract by Eq. (28).

$$\left[K_d - \omega_d^2\,(\alpha)\,M\right]\phi_d\,(\alpha) = \{0\} \tag{28}$$

Where $K_d$ is global stiffness matrix updated by Eq. (7), and $\omega_d$ and $\phi_d\,(\alpha)$ are the natural frequencies and the mode shapes at damaged stage, which is determined by $0 < \alpha_i,\ (i = 1, 2, \ldots, D) < 1$, respectively.

We assume that, two items, including the natural frequencies $\omega_{Measured}$ and mode shapes $\phi_{Measure}$, which are measured for the damaged structure, are available. Normally, these items will be measured by sensors that are attached to the structures during construction. Thus, we will have a set of values defined as follows $(\omega_u,\ \phi_u,\ \omega_d\,(\alpha)\,,\ \phi_d\,(\alpha)\,,\ \omega_{Measure},\ \phi_{Measure})$. The objective function in the SDI problem will be determined based on the correlation between these values. In this work, we used two functions, namely Multiple Damage Location Assurance Criterion Algorithm (*MDLAC*) [71] and Modal Assurance Criterion (*MAC*) [72,73] to establish the objective function.

*MDLAC* focuses on the change in frequencies by determining the level of correlation between measured the natural frequencies of undamaged structure $\omega_u$ and natural frequencies of damaged structure $\omega_d\,(\alpha)$ calculated by the assignment of the damaged

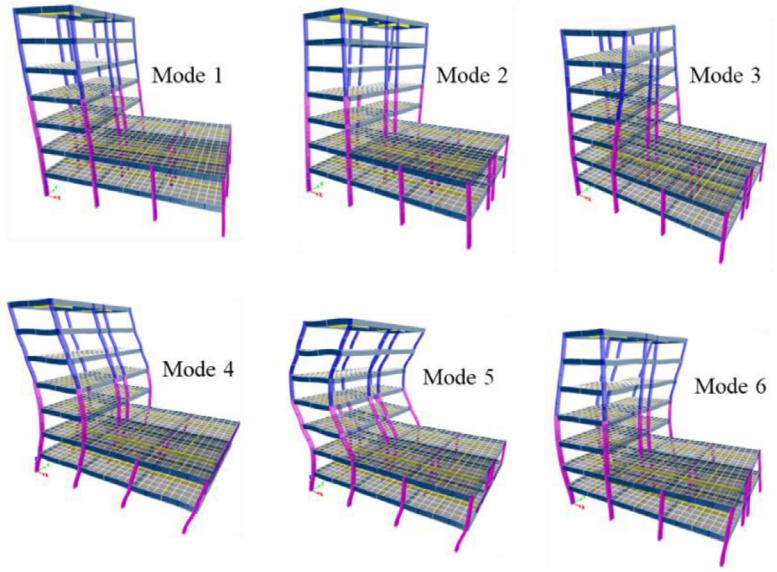


**Fig. 22.** The first six mode shapes of the 3D concrete structure.

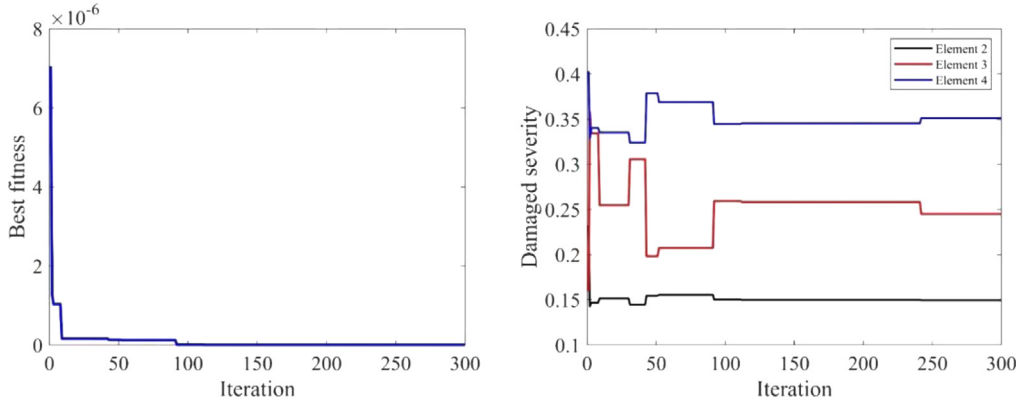


**Fig. 23.** The convergence rate of the objective function and the historical search of the damaged elements in Case 1.

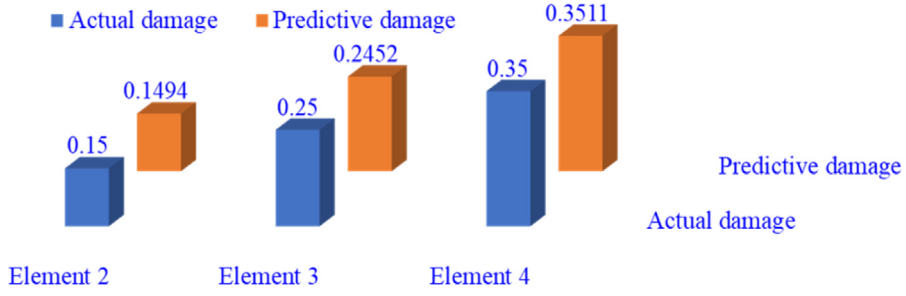


**Fig. 24.** The comparison between actual damage and predictive damage in Case 1.

vector $0 < \alpha_i,\ (i = 1, 2, \ldots, D) < 1$ as shown in Eq. (29), and the level of correlation between $\omega_u$ and measured natural frequencies $\omega_{Measured}$ at the real-damaged stage of the structures as given in Eqs. (29) and (30)

$$\delta\omega(\alpha) = \frac{\omega_u - \omega_d(\alpha)}{\omega_u} \tag{29}$$

$$\Delta\omega = \frac{\omega_u - \omega_{Measured}}{\omega_u} \tag{30}$$

Given a pair of parameter vectors, one can estimate the level of correlation in several ways. An efficient way is to evaluate a correlation-based index, termed the multiple damage location assurance criterion (*MDLAC*), which is expressed in Eq. (31).

$$MDLAC(\alpha) = \frac{\left|\Delta\omega^T \delta\omega(\alpha)\right|^2}{\left(\Delta\omega^T \Delta\omega\right)\left(\delta\omega^T(\alpha)\,\delta\omega(\alpha)\right)} \tag{31}$$

*MAC* focuses on the change in the mode shapes by determining the level of correlation between measured mode shapes for damaged stage $\phi_{Measure}$ and the mode shape $\phi_d(\alpha)$ extracted from Eq. (28), in which the assignment the damaged vector $0 < \alpha_i,\ (i = 1, 2, \ldots, D) < 1$ is considered. *MAC at* $j$th mode shape is

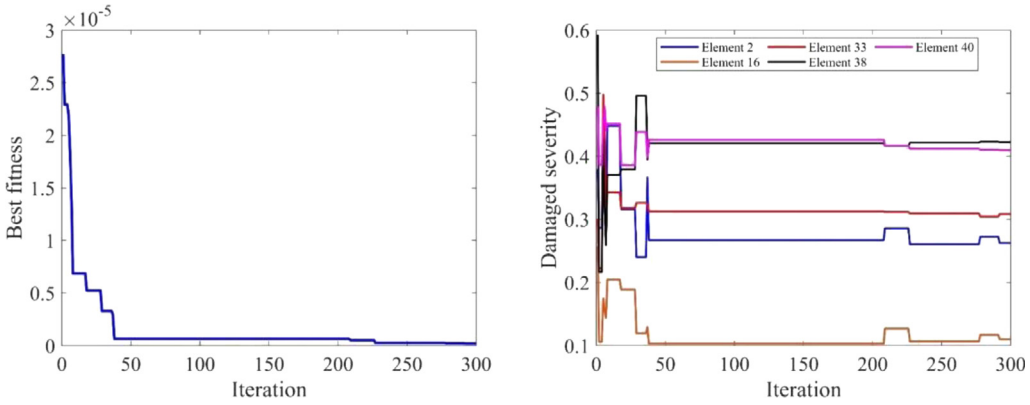


**Fig. 25.** The convergence rate of the objective function and the historical search of the damaged elements for case 2.

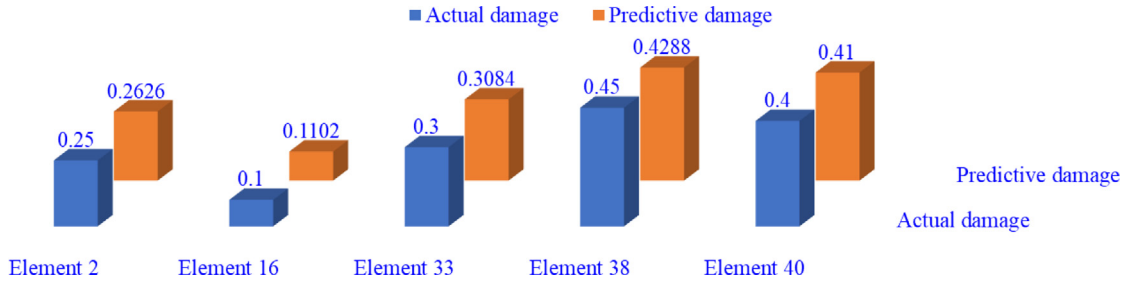


**Fig. 26.** The comparison between actual damage and predictive damage for case 2.

given in Eq. (32).

$$MAC_j(\alpha) = \frac{\left|\{\phi_{j,Measured}\}^T \{\phi_{j,d}(\alpha)\}\right|^2}{\left(\{\phi_{j,Measured}\}^T \{\phi_{j,Measured}\}\right)\left(\{\phi_{j,d}(\alpha)\}^T \{\phi_{j,d}(\alpha)\}\right)} \quad (32)$$

It can be seen that the two terms MAC and MDLAC represent the change of frequencies and mode shapes. They vary from a minimum value of zeros to a maximum value of one. They have maximum values, when the vector of analytical frequencies $\omega_d(\alpha)$ and mode shapes $\phi_d(\alpha)$ extracted from Eq. (28) becomes identical to $\omega_{Measured}$ and $\phi_{Measure}$, respectively. By combining these two terms, we propose an objective function as shown in Eq. (33).

$$\Gamma(x) = w_1 \frac{1}{n_\omega} \sum_{j=1}^{n_\omega} \left(1 - \sqrt{MDLAC(\alpha)}\right) + w_2 \frac{1}{n_\phi} \sum_{j=1}^{n_\phi} \left(1 - \sqrt{MAC_j(\alpha)}\right) \quad (33)$$

Where $n_\omega$ and $n_\phi$ are the numbers of frequencies and mode shapes, respectively, and $w_1$, $w_2$ are weighting factors. In general, the weighting factor reflects the relative importance of each variable and it can be adjusted for a particular structure.

Based on the application of the metaheuristic optimization algorithm to minimize the objective function given Eq. (33), we can find the best solution $0 < \alpha_i, (i = 1, 2, \ldots, D) < 1$, which considers the damage severities in the structural elements.

### 7.2. Damage identification

The inverse method using FE model updating combined with the reliability optimization algorithm is an effective method for determining the structural damage identification. This is a process to adjust an initial FE model to reach an acceptable agreement between damaged structure and FE model [74,75]. Based on FE model analysis, Eq. (33) can be implemented to calculate the objective function. In previous studies, solving this equation is usually done by FE method, which is conducted by programming languages such as MATLAB, Python, Visual Basic.NET, Visual Basic for Applications (VBA), C#, Visual C++, Visual Fortran. Therefore, these previous studies only focused on simple models with few elements or with simple shapes [76–78]. For the structures having complexity, and large number of elements, using the above programming languages for simulation is difficult and sometimes impossible. To overcome this limitation, in this work, for the first time, we have successfully developed a sub-program, which make the link between SAP2000 and MATLAB language. This program can allow the user to adjust the initial model's parameters in SAP2000 to create a continuous two ways data exchange between SAP2000 and MATLAB.

Computers and Structures, Inc (https://www.csiamerica.com/products/sap2000) created SAP2000, a well-known commercial software that is widely used in the field of civil engineering. Based on strong tools, this software is used to simulate the FE models of real-structures ranging from simple to complicated geometries. The definition of materials, elements, geometry, and analysis types are the basis steps in the FE simulation in SAP2000. The OAPI is a programming tool that allows a third-party to change the analysis option and design topology of a FE model in SAP2000 using command codes of programming languages instead of using point-and-click procedure through the SAP2000's interface. It leads to the creation of a custom SAP2000 interface, which may be adjusted to the needs of the user or embedded with other programming applications. The developed sub-program controls the structural model in SAP2000 for the stiffness updating process for each element, analyzes the model and provides the results. Consequently, these results are exchanged to KO program to calculate the objective function and identify the damage severities (see Fig. 17).

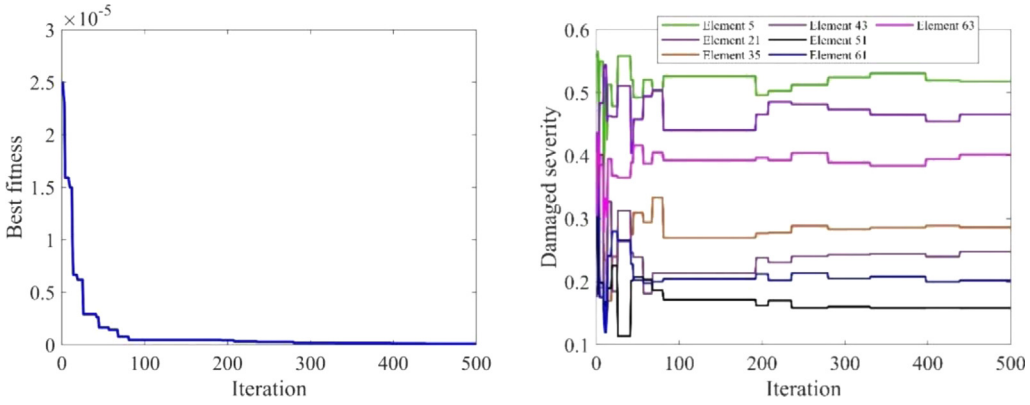


**Fig. 27.** The convergence rate of the objective function and the historical search of the damaged elements for case 3.

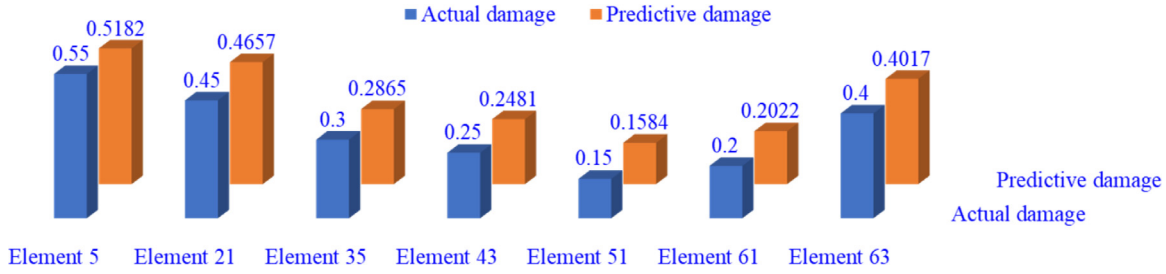


**Fig. 28.** The comparison between actual damage and predictive damage for case 3.

**Table 16**
Geometric and material properties for the 3D concrete structure.

| | |
|---|---|
| Geometric parameters | |
| @span in short direction | 8000 mm/2500 mm/8000 mm |
| @span in long direction | 8000 mm/8000 mm/8000 mm |
| Inter-story height | 3600 mm |
| Sections properties | |
| Columns (width × height) from story 1 to story 4 | (300 × 900) mm |
| Columns (width × height) from story 5 to story 7 | (300 × 500) mm |
| Main beams (width × height) for all stories | (300 × 700) mm |
| Sub-beams (width × height) for all stories | (200 × 500) mm |
| Slabs (thickness) for all stories | 180 mm |
| Material | |
| Columns for all stories | C30/35 |
| Slabs and beams for all stories | C20/25 |

### *7.3. The process of damage identification in a complex 3D concrete structure using KO*

In this section, a complex-3D concrete structure is used to assess the performance of KO for application to SDI. The structure has seven stories and a total height of 25.2 m. Its total long span is 24 m, and its total short span is 18.5 m. The 3D FE model of this structure is constructed using SAP2000 software, which consists of 214 elements as shown in Fig. 18. The geometry, cross-section properties, and material properties of the investigated concrete structure are given in Table 16. The dimensions of the left-side view and structural layout plans with respect to the other levels of this structure are illustrated in Figs. 19 and 20, respectively.

The columns are assumed to have different damage severities, which are distributed from the 1st story to the 7th story. Three damage cases are used to valid the KO's performance. The number of damaged columns increases with respect to case 1, case 2 and case 3 as shown in Table 17. Case 1 has three damaged columns in the 1st story. Case 2 has five damaged columns in the 1st and 2nd story. And finally, in case 3, the damaged columns are in the 1st to the 7th story at the same straight line. The location of damaged elements is illustrated in Fig. 21. For this test, the size of the initial population and the number of search agents are selected as $N = 30$. The maximum number of iterations is $T_{max} = 300$. For case 3, the number of iterations is increased to $T_{\max} = 500$. The first six frequencies $n_\omega = 6$, and the first six mode shapes $n_\phi = 6$ as shown in Fig. 22, are used to establish the objective function.

The results in the three cases are illustrated in Table 17:

- The objective function's convergence trend: This curve demonstrates the capacity to exploit the new best global optima of KO.
- The historical search for damaged elements assumed in each case: These curves indicate the tendency of finding the best solution, as well as the algorithm's quick or slow convergence rate.
- The severity of the damage in each case: This is shown by the final values obtained during the course of iterations, which indicate the difference between the true damaged

**Table 17**
Reduction in stiffness in elements of the 3*D* concrete structure for different damage cases.

| Case study | Damaged elements assumption | Damage severity assumption |
|---|---|---|
| Case 1 | Element 2 | 0.15 |
| | Element 3 | 0.25 |
| | Element 4 | 0.35 |
| Case 2 | Element 2 | 0.25 |
| | Element 16 | 0.1 |
| | Element 33 | 0.3 |
| | Element 38 | 0.45 |
| | Element 40 | 0.4 |
| Case 3 | Element 5 | 0.55 |
| | Element 21 | 0.45 |
| | Element 35 | 0.3 |
| | Element 43 | 0.25 |
| | Element 51 | 0.15 |
| | Element 61 | 0.2 |
| | Element 63 | 0.4 |

**Table 18**
The predictive results for case 1, case 2 and case 3 of 3D concrete structure using KO.

| Case | Element | Actual damage | Predictive damage | Absolute error |
|---|---|---|---|---|
| Case 1 | Element 2 | 0.15 | 0.1494 | 0.0006 |
| | Element 3 | 0.25 | 0.2452 | 0.0048 |
| | Element 4 | 0.35 | 0.3511 | 0.0011 |
| Case 2 | Element 2 | 0.25 | 0.2626 | 0.0126 |
| | Element 16 | 0.1 | 0.1102 | 0.0102 |
| | Element 33 | 0.3 | 0.3084 | 0.0084 |
| | Element 38 | 0.45 | 0.4288 | 0.0212 |
| | Element 40 | 0.4 | 0.41 | 0.01 |
| Case 3 | Element 5 | 0.55 | 0.5182 | 0.0318 |
| | Element 21 | 0.45 | 0.4657 | 0.0157 |
| | Element 35 | 0.3 | 0.2865 | 0.0135 |
| | Element 43 | 0.25 | 0.2481 | 0.0019 |
| | Element 51 | 0.15 | 0.1584 | 0.0084 |
| | Element 61 | 0.2 | 0.2022 | 0.0022 |
| | Element 63 | 0.4 | 0.4017 | 0.0017 |

values assumed in structural elements and the predictive damage values found by KO.

With the number of iteration $T_{max} = 300$, KO can predict the damage severity with good accuracy for case 1 and case 2 with respect to three and five damaged elements. The damage severity can be exploited by KO with $T_{max} = 500$ when the damaged elements increase to seven elements in case 3. As illustrated in Figs. 23, 25 and 27, the convergence of KO reaches a stable rate after only a few iterations. This demonstrates KO's effective movement strategy for exploitation. Thus, KO can move very quickly to potential areas during the first few iterations to improve the convergence rate. Especially, the ability to escape the local optima of KO can be seen during the last iterations, KO can still continue to find the better solution. KO achieves the result with very small error, as shown in Fig. 24, Fig. 26, Fig. 28 and Table 18.

## 8. Conclusion and future works

K-Mean Optimizer (KO) is a new metaheuristic optimization technique introduced in this article to tackle a wide range of optimization problems, from numerical functions to real-design challenges. While the movement methods in other optimization algorithms are based on natural principles, this can result in significant limitations. Because the algorithms attempt to imitate these laws in unexpected ways, KO produces a radically different trend than all other algorithms. In KO, mathematical formulas are used to find the optimal solution instead of natural laws. For the first time, K-Mean clustering algorithm is used to establish the foundation of a new optimization based on the centroid vector position in each cluster region. Thus, the movement strategies for exploitation in KO are based on high quality search spaces, which is established by the centroid vector positions and the current best solution found at each iteration. In this way, the outstanding advantages of the K-Mean clustering algorithm have been fully exploited to bring promising results for KO. At the same time, the ability to explore the new search spaces in KO is also created in a different way. Thus, the new search spaces in KO are secured by modifying the step length $S$ and replacing the old search space by new search spaces. Compared to other algorithms, this algorithm shows a significant advantage in solving problems with a large number of local optima. To evaluate the effectiveness of KO, five examples including 23 classical benchmarks, CEC2005, CEC2014 and three real-world design problems are presented. The optimal results obtained by KO are compared with the original PSO and seven well-known algorithms including GWO, WOA, SMA, AOA, RSA, AO and LSHASE. The statistical results in the classical benchmark functions and engineering problems show that KO performs better in exploitation and exploration compared to the other algorithms (except LSHADE). Finally, the performance of KO in solving the SDI problem is evaluated using a real-life complex 3D concrete structure. For this purpose, a sub-program has been successfully developed to link SAP2000 and MATLAB software. The results obtained show that KO is highly appreciated in solving SDI problems with very high accuracy and reliability. Among current state-of-the-art optimization algorithms, especially for SDI application, KO is recommended as a viable alternative strategy for solving optimization issues in general.

The application of KO for solving complicated test functions and real-world problems can be researched in further future studies. The turning parameters can be investigated by an improved version of KO to achieve a full evaluation of its performance. At the same time, based on the advantages of the program, which can link MATLAB and SAP2000, a promising research topic for topology optimization using KO can be developed in the future.

## CRediT authorship contribution statement

**Hoang-Le Minh:** Conceptualization, Software, validation, Writing – original draft. **Thanh Sang-To:** Validation. **Magd Abdel Wahab:** Supervision, Funding acquisition, Writing – review & editing. **Thanh Cuong-Le:** Editing, Methodology, Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

The authors acknowledge the financial support of the VLIR-UOS TEAM Project, VN2018TEA479A103, 'Damage assessment tools for Structural Health Monitoring of Vietnamese infrastructures' funded by the Flemish Government. The authors wish to express their gratitude to Van Lang University, Vietnam for financial support for this research.

The authors would like to acknowledge the support from Ho Chi Minh City Open University under the basic research fund (No. **E2021.05.2**).

## References


[1] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.

[2] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, 1995.

[3] J. Jin, et al., A new cluster analysis based on combinatorial particle swarm optimization algorithm, in: Proceedings of the 2016 International Conference on Education, Management, Computer and Society, Shenyang, China, 2016.

[4] A. Ahmadyfard, H. Modares, Combining PSO and K-means to enhance data clustering, in: 2008 International Symposium on Telecommunications, IEEE, 2008.

[5] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Software 69 (2014) 46–61.

[6] L. Korayem, M. Khorsid, S. Kassem, A hybrid K-means metaheuristic algorithm to solve a class of vehicle routing problems, Adv. Sci. Lett. 21 (12) (2015) 3720–3722.

[7] F. Glover, Tabu search: A tutorial, Interfaces 20 (4) (1990) 74–94.

[8] H.R. Lourenço, O.C. Martin, T. Stützle, Iterated local search, in: Handbook of Metaheuristics, Springer, 2003, pp. 320–353.

[9] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (11) (1997) 1097–1100.

[10] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.

[11] B. Basturk, An artificial bee colony (ABC) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium, Indianapolis, in, USA, 2006, 2006.

[12] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing, NaBIC, Ieee, 2009.

[13] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inform. Sci. 237 (2013) 82–117.

[14] S. Mahdavi, M.E. Shiri, S. Rahnamayan, Metaheuristics in large-scale global continues optimization: A survey, Inform. Sci. 295 (2015) 407–428.

[15] H. Poorzahedy, O.M. Rouhani, Hybrid meta-heuristic algorithms for solving network design problem, European J. Oper. Res. 182 (2) (2007) 578–596.

[16] M. Moussaid, et al., Collective information processing and pattern formation in swarms, flocks, and crowds, Top. Cogn. Sci. 1 (3) (2009) 469–497.

[17] S. Camazine, et al., Self-Organization in Biological Systems, Princeton University Press, 2020.

[18] E. Bonabeau, et al., Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, 1999.

[19] A. Chakraborty, A.K. Kar, Swarm intelligence: A review of algorithms, in: Nature-Inspired Computing and Optimization, 2017, pp. 475–494.

[20] S. Patnaik, X.-S. Yang, K. Nakamatsu, Nature-Inspired Computing and Optimization, Vol. 10, Springer, 2017.

[21] D.E. Goldberg, J.H. Holland, Genetic Algorithms and Machine Learning, 1988.

[22] R. Storn, K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.

[23] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.

[24] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[25] B. Webster, P.J. Bernhard, A local search optimization algorithm based on natural principles of gravitation, 2003.

[26] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.

[27] A. Kaveh, S. Talatahari, A novel heuristic optimization method: Charged system search, Acta Mech. 213 (3) (2010) 267–289.

[28] R. Formato, Central force optimization: A new metaheuristic with applications in applied electromagnetics, Prog. Electromagn. Res. 77 (2007) 425–491.

[29] P.C. Pinto, T.A. Runkler, J.M. Sousa, Wasp swarm algorithm for dynamic MAX-SAT problems, in: International Conference on Adaptive and Natural Computing Algorithms, Springer, 2007.

[30] C. Yang, X. Tu, J. Chen, Algorithm of marriage in honey bees optimization based on the wolf pack search, in: The 2007 International Conference on Intelligent Pervasive Computing, IPC 2007, IEEE, 2007.

[31] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2) (2010) 78–84.

[32] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.

[33] B. Abdollahzadeh, F.S. Gharehchopogh, S. Mirjalili, African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems, Comput. Ind. Eng. 158 (2021) 107408.

[34] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Comput. Aided Des. 43 (3) (2011) 303–315.

[35] A.H. Kashan, League championship algorithm: A new algorithm for numerical function optimization, in: 2009 International Conference of Soft Computing and Pattern Recognition, IEEE, 2009.

[36] A. Ahmadi-Javid, Anarchic society optimization: A human-inspired method, in: 2011 IEEE Congress of Evolutionary Computation, CEC, IEEE, 2011.

[37] W. Zhao, L. Wang, S. Mirjalili, Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications, Comput. Methods Appl. Mech. Engrg. 388 (2022) 114194.

[38] H.-L. Minh, et al., A variable velocity strategy particle swarm optimization algorithm (VVS-PSO) for damage assessment in structures, Eng. Comput. (2021) 1–30.

[39] H.-L. Minh, et al., An enhancing particle swarm optimization algorithm (EHVPSO) for damage identification in 3D transmission tower, Eng. Struct. 242 (2021) 112412.

[40] E.W. Forgy, Cluster analysis of multivariate data: Efficiency versus interpretability of classifications, Biometrics 21 (1965) 768–769.

[41] L. Abualigah, et al., The arithmetic optimization algorithm, Comput. Methods Appl. Mech. Eng. 376 (2021) 113609.

[42] S. Li, et al., Slime mould algorithm: A new method for stochastic optimization, Future Gener. Comput. Syst. 111 (2020) 300–323.

[43] A. Faramarzi, et al., Marine predators algorithm: A nature-inspired metaheuristic, Expert Syst. Appl. 152 (2020) 113377.

[44] L. Abualigah, et al., Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer, Expert Syst. Appl. 191 (2022) 116158.

[45] L. Abualigah, et al., Aquila optimizer: A novel meta-heuristic optimization algorithm, Comput. Ind. Eng. 157 (2021) 107250.

[46] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2014.

[47] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, Comput. Struct. 169 (2016) 1–12.

[48] A.A. Heidari, et al., Harris hawks optimization: Algorithm and applications, Future Gener. Comput. Syst. 97 (2019) 849–872.

[49] A. Faramarzi, et al., Equilibrium optimizer: A novel optimization algorithm, Knowl. -Based Syst. 191 (2020) 105190.

[50] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (1) (2007) 89–99.

[51] E. Mezura-Montes, C.A.C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (4) (2008) 443–473.

[52] C.A.C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2) (2000) 113–127.

[53] A. Kaveh, M. Khayatazad, A new meta-heuristic method: Ray optimization, Comput. Struct. 112 (2012) 283–294.

[54] T. Ray, K.-M. Liew, Society and civilization: An optimization algorithm based on the simulation of social behavior, IEEE Trans. Evol. Comput. 7 (4) (2003) 386–396.

[55] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (1) (2007) 340–356.

[56] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice, Comput. Methods Appl. Mech. Engrg. 194 (36–38) (2005) 3902–3933.

[57] S. Mirjalili, et al., Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Software 114 (2017) 163–191.

[58] F.A. Hashim, A.G. Hussien, Snake optimizer: A novel meta-heuristic optimization algorithm, Knowl.-Based Syst. (2022) 108320.

[59] J.S. Arora, Introduction to Optimum Design, Elsevier, 2004.

[60] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice, Comput. Methods Appl. Mech. Engrg. 194 (36) (2005) 3902–3933.

[61] B. Akay, D. Karaboga, Artificial bee colony algorithm for large-scale problems and engineering design optimization, J. Intell. Manuf. 23 (4) (2012) 1001–1014.

[62] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. (2010).

[63] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: A nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2) (2016) 495–513.
[64] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inform. 16 (3) (2002) 193–203.
[65] M.M. Atiqullah, S. Rao, Simulated annealing and parallel processing: An implementation for constrained global design optimization, Eng. Optim.+ A35 32 (5) (2000) 659–685.
[66] S. He, E. Prempain, Q. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Eng. Optim. 36 (5) (2004) 585–605.
[67] W.-X. Ren, G. De Roeck, Structural damage identification using modal data. I: Simulation verification, J. Struct. Eng. 128 (1) (2002) 87–95.
[68] J.N. Yang, et al., An adaptive extended Kalman filter for structural damage identification, Struct. Control Health Monitor.: Official J. Int. Assoc. Struct. Control Monitor. Eur. Assoc. Control Struct. 13 (4) (2006) 849–867.
[69] U. Lee, J. Shin, A frequency response function-based structural damage identification method, Comput. Struct. 80 (2) (2002) 117–132.
[70] C.R. Farrar, S.W. Doebling, D.A. Nix, Vibration–based structural damage identification, Phil. Trans. R. Soc. A 359 (1778) (2001) 131–149.
[71] A. Messina, E. Williams, T. Contursi, Structural damage detection by a sensitivity and statistical-based method, J. Sound Vib. 216 (5) (1998) 791–808.
[72] S.W. Doebling, C.R. Farrar, M.B. Prime, A summary review of vibration-based damage identification methods, Shock Vib. Digest 30 (2) (1998) 91–105.
[73] S.W. Doebling, et al., Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in their Vibration Characteristics: A Literature Review, Los Alamos National Lab. NM (United States), 1996.
[74] T. Marwala, Finite-element-model updating using computional intelligence techniques: Applications to structural dynamics, 2010.
[75] N.F. Alkayem, et al., Structural damage detection using finite element model updating with evolutionary algorithms: A survey, Neural Comput. Appl. 30 (2) (2018) 389–411.
[76] S. Tiachacht, et al., Damage assessment in structures using combination of a modified Cornwell indicator and genetic algorithm, Eng. Struct. 177 (2018) 421–430.
[77] H.-L. Minh, et al., A two-step approach for damage detection in a real 3D tower using the reduced-order finite element model updating and atom search algorithm (ASO), in: Proceedings of the 2nd International Conference on Structural Damage Modelling and Assessment, Springer, 2022.
[78] C. Le Thanh, et al., Combination of intermittent search strategy and an improve particle swarm optimization algorithm (IPSO) for damage detection of steel frame, Frattura ed Integr. Strut. 16 (59) (2022) 141–152.