Information Sciences 630 (2023) 463–481

Contents lists available at ScienceDirect

Information Sciences

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

# Self-adaptive teaching-learning-based optimizer with improved RBF and sparse autoencoder for high-dimensional problems [☆]

Jing Bi [a], Ziqi Wang [a], Haitao Yuan [b,*], Jia Zhang [c], MengChu Zhou [d]

[a] School of Software Engineering in Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[b] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[c] Department of Computer Science in the Lyle School of Engineering at Southern Methodist University, Dallas, TX 75205, USA
[d] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA

## ARTICLE INFO

*Keywords:*
Evolutionary algorithms
Swarm intelligence algorithms
Teaching-learning-based optimizer
Autoencoders
Radial basis function model

## ABSTRACT

Evolutionary algorithms and swarm intelligence ones are commonly used to solve many complex optimization problems in different fields. Yet, some of them have limited performance when dealing with high-dimensional complex problems because they often require enormous computational resources to yield desired solutions, and some of them may easily trap into local optima. To solve this problem, this work proposes a Self-adaptive Teaching-learning-based Optimizer with an improved Radial basis function model and a sparse Autoencoder (STORA). In STORA, a Self-adaptive Teaching-Learning-Based Optimizer (STLBO) is designed to dynamically adjust parameters for balancing exploration and exploitation abilities. Then, a sparse autoencoder (SAE) is adopted as a dimension reduction method to compress a search space into a lower-dimensional one for more efficiently guiding a population to converge towards global optima. Besides, an Improved Radial Basis Function model (IRBF) is designed as a surrogate one to balance training time and prediction accuracy. It is adopted to save computational resources for improving overall performance. In addition, a dynamic population allocation strategy is adopted to well integrate SAE and IRBF in STORA. We evaluate STORA by comparing it with several state-of-the-art algorithms through eight benchmark functions. We further test its actual performance by applying it to solve a real-world computation offloading problem.

## 1. Introduction

Evolutionary algorithms (EAs) and swarm intelligence (SI) ones have been widely applied to solve different types of benchmarks and real-world engineering optimization problems in a variety of fields, *e.g.*, image classification [1], scheduling problems [2], autonomous mobile robots [3], and power system planning [4]. Some practical problems have a large number of decision variables and can be characterized as high-dimensional problems [5]. These problems present a large and exponentially growing search space with many decision variables that bring big challenges for these metaheuristics to efficiently explore their search spaces with limited computational resources. In other words, they often require a large number of function evaluations (FEs) to yield satisfactory

☆ This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62173013 and 62073005, and the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1203.

* Corresponding author.
*E-mail address:* haitao.yuan@njit.edu (H. Yuan).

https://doi.org/10.1016/j.ins.2023.02.044
Received 27 September 2022; Received in revised form 8 February 2023; Accepted 9 February 2023

Available online 20 February 2023

solutions. However, FEs in many real-world problems can be computationally intensive or highly costly [6]. Some traditional EAs, *e.g.,* genetic algorithm (GA) [7] and some traditional SI algorithms, *e.g.,* grey wolf optimization (GWO) [8] and particle swarm optimization (PSO) [9] may easily trap into local optima because their strategies cannot well cover whole high-dimensional spaces.

To solve high-dimensional problems, several studies have been proposed, which can be divided into two types. The first type incorporates surrogate models into EAs. Surrogate-assisted EAs (SAEAs) have been considered as viable methods to deal with high-dimensional problems. For example, Cui et al. [10] employ a surrogate model to replace a part of a true model for evaluating individuals. Thus, it avoids some unnecessary FEs to save computational resources, which can be used for EAs to further explore a search space. Dong et al. [11] build a dynamically updated surrogate model to assist EAs. Specifically, subspaces and local surrogate models are constructed based on selected promising data points. Then, the most potential points are captured according to the prediction result of surrogate models for subsequent updates and optimization. However, surrogate models need to continue their updates that bring additional computational cost.

The second type belongs to the dimension reduction, which is widely adopted to deal with the huge amount of high-dimensional data because of the curse of dimensionality [12,13]. However, there are relatively few existing studies focusing on the combination of dimension reduction and metaheuristics. Yao et al. [14] propose a dimension reduction method of objective functions to reduce computational stress. A Spearman correlation coefficient is adopted to evaluate correlations among objective functions. Then, objective functions are clustered by the K-means algorithm based on their correlations. Tenne et al. [15] propose two complementary dimension reduction frameworks for EAs. One framework identifies an important subset of original variables by adopting selection of variables while the other uses topological mapping to project high-dimensional data to a lower-dimensional space.

However, some problems still exist and need to be solved. First, for the first type, SAEAs bring additional surrogate models into a structure that also brings additional training time especially for a high-dimensional training set. Moreover, an optimization process is partially guided by surrogate models whose accuracy has direct impact on the optimization direction. Inaccurate surrogate models may mislead the optimization direction and result in poor or inaccurate search results. Second, for the second type, although the feature data is extracted under a specific dimension reduction method, some data including important information for an optimization process may be lost. As a result, it is highly important to choose a proper method for dimension reduction.

In addition to two types of methods, it is important to consider real-word situations when designing algorithms. Tao et al. [16] point out that most modern existing data-driven fault diagnosis methods require sufficient training samples to achieve desired diagnostic effect. However, scarcity of fault samples in actual industrial environments leads to limitation of these methods. In that case, Tao et al. propose a model for a problem of fewer fault samples and cross-domain among data sets in real industrial environments. As a result, to improve feasibility in real-world problems, our work needs to focus on characteristics of real-world problems, *e.g.*, data volumes, and various uncertainties. Accordingly, similar to [17], our work adopts a combination of typical benchmark functions and a real-word problem to evaluate performance of our proposed algorithm.

Motivated by above analysis, this work proposes a novel Self-adaptive Teaching-learning-based Optimizer with an improved Radial basis function model and a sparse Autoencoder (STORA) to solve high-dimensional problems. Main contributions of this work are summarized as:

1. A Self-adaptive Teaching-Learning-Based Optimizer (STLBO) is proposed as an SI algorithm in STORA to solve high-dimensional problems. Its parameters are dynamically changed as the number of iterations increases to balance exploration and exploitation abilities in different stages.
2. An Improved Radial Basis Function model (IRBF) is proposed as a surrogate model in STORA to balance prediction accuracy and training time. We take advantage of IRBF to replace a part of true FEs. Those saved computational resources can be used for STLBO to better explore a search space.
3. A novel framework is proposed to integrate both dimension reduction and surrogate models into STLBO to solve high-dimensional problems. A sparse autoencoder (SAE) is adopted as a dimension reduction tool and IRBF as a surrogate model in STORA. Moreover, a dynamic population allocation strategy is adopted to allow SAE and IRBF to cooperate well. We compare STORA with state-of-the-art peers by using different unimodal and multimodal high-dimensional functions, and a real-world computation offloading problem in mobile edge computing to demonstrate its superior performance.

The rest of this work is organized as follows. Section 2 discusses background of TLBO, autoencoders and surrogate models. Section 3 describes an overall framework of STORA. Section 4 gives setting of experimental parameters and experimental results of STORA and its benchmark algorithms. Section 5 gives the conclusion.

## 2. Background

### 2.1. Teaching-learning-based optimization

Teaching-learning-based optimization (TLBO) is a swarm-based heuristic optimization algorithm, which mimics a traditional mode of classroom teaching [18]. In TLBO, learners represent solutions of a problem and knowledge levels of them mean fitness values of individuals in a population. The individual with the most knowledge is the teacher and other learners move towards it. Compared with other state-of-the-art EAs and SI algorithms, TLBO is not quite sensitive to dimensions of problems. Thus, it is suitable to solve high-dimensional problems [19]. However, its optimization speed and results are still affected as dimensions become bigger. It is assumed that there are $N$ random learners in a search space.

**Algorithm 1** Learning phase of TLBO.

```
1: for $j = 1{:}N$ do
2:   Randomly pick $k \in [1, N]$ where $k \neq j$
3:   for $d = 1{:}D$ do
4:     if $f(X^j) < f(X^k)$ then
5:       $X_d^j(t+1) = X_d^j(t) + r_d \cdot \left(X_d^j(t) - X_d^k(t)\right)$
6:     else
7:       $X_d^j(t+1) = X_d^j(t) + r_d \cdot \left(X_d^k(t) - X_d^j(t)\right)$
8:     end if
9:   end for
10: end for
```

$$\begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^N \end{bmatrix} = \begin{bmatrix} X_1^1, & X_2^1, & \cdots, & X_D^1 \\ X_1^2, & X_2^2, & \cdots, & X_D^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^N, & X_2^N, & \cdots, & X_D^N \end{bmatrix} \tag{1}$$

where $D$ denotes the number of decision variables, $\forall X_d^j \in [\check{b}_d, \hat{b}_d](1 \le d \le D)$, $\check{b}_d$ and $\hat{b}_d$ represent lower and upper bounds of dimension $d$. Besides, $X^j$ denotes solution $j$ where $j \in [1, 2, 3, \cdots, N]$ and $X_d^j$ denotes its dimension $d$. The search process of TLBO consists of two phases including teaching and learning ones.

In a teaching phase, a teacher shares the knowledge to learners for increasing the average knowledge level of the class. All learners learn from their teacher to improve their knowledge. Relevant formulas of the teaching phase are summarized as:

$$\begin{aligned} M_d &= \frac{1}{N} \cdot \sum_{j=1}^{N} X_d^j \\ T_F &= \mathbf{round}[1 + \mathbf{rand}(0, 1)] \\ X_d^j(t+1) &= X_d^j(t) + r_d \cdot \left(X_d^n(t) - T_F \cdot M_d(t)\right) \end{aligned} \tag{2}$$

where $X_d^j(t)$ ($X_d^j(t+1)$) denotes dimension $d$ of individual $j$ in iteration $t$ ($t+1$). Moreover, $X_d^n(t)$ denotes dimension $d$ of a teacher in iteration $t$, $r_d$ denotes a random variable ranging from $[0, 1]$, and $M_d(t)$ denotes the mean position of the population at dimension $d$ in iteration $t$. $T_F$ denotes a learning factor, which is a random integer in $[1, 2]$.

In a learning phase, learners learn from other ones to improve their knowledge, the procedure is shown in Algorithm 1, where $\mathbf{f}(\cdot)$ denotes FE. Given any two individuals $j$ and $k$, if individual $j$ performs better than individual $k$, the latter randomly moves to the former; otherwise, the former moves to the latter.

## *2.2. Autoencoder in dimension reduction*

Autoencoder (AE) is a type of unsupervised neural networks with three layers. It takes an input vector to the next layer by nonlinear mapping that achieves dimension reduction [20]. It tries to approximate the same function by minimizing an expected reconstruction error, and therefore, an output value is close to an input one. As a result, AE is an effective tool to reduce dimensions. It learns inherent features with reduced hidden nodes, which are expected to completely reconstruct an original function and this representation is presented in a lower-dimensional manner. Chen et al. [21] adopt an AE to reduce features of data, and keep critical information of data to facilitate information retention. In this way, important features of data are kept while others are removed. However, it still has possibility that some important features are removed because of an inaccurate AE. Gao et al. [22] adopt an AE to convert an original large-dimensional seismic inversion problem into a compressed one that can be effectively solved by global optimization. In addition, there are several studies on integrating AEs with metaheuristics. Miranda et al. [23] apply AEs as reversible mapping between an original search space and a compressed one. Then, they organize evolutionary searches in an equivalent reduced dimension search space to speed up convergence of metaheuristics. Cui et al. [24] pioneer in an autoencoder-embedded evolutionary optimization framework to solve high-dimensional expensive problems. AE is adopted to extract important topographic information of a search space. Thus, a lower-dimensional informative space is constructed based on a trained AE where individuals generate offspring. However, just like other neural networks, performance of AE drops rapidly when there is a large amount of high-dimensional training data.

We also adopt AE to compress an original space into a reduced one that facilitates evolution. We intend to allow some individuals to produce offspring in a lower-dimensional space. In that case, individuals may have higher possibility to quickly produce promising solutions than those in an original space. To fit large amount of high-dimensional training data, we choose SAE in our structure to be introduced later. With the help of the trained SAE, a population is encoded into a low-dimensional space where offspring generation is performed. The trained SAE can capture useful landscape information about promising areas as data samples used to train it become closer to the optimal case. Guided by this information, metaheuristics are easier to locate promising areas. Then, those offspring in the compressed space can be decoded to an original space for FEs in a decoding phase of SAE.

**Table 1**
Main notations.

| Notation | Definition |
|---|---|
| $N$ | Population size |
| $D$ | Dimension of a problem |
| $A_1(A_2)$ | Knowledge acquisition factor in a teaching (learning) phase |
| $q$ | Number of samples used to train SAE |
| $h$ | Number of hidden neurons in SAE |
| $p$ | Number of samples used to train IRBF |
| $m$ | Number of basis function centers |
| $\gamma$ | Trained IRBF |
| $M$ | Maximum number of re-evaluations |
| $P_1(P_2)$ | Sub-populations assisted by SAE (IRBF) |
| $B_1(B_2)$ | Database used to train SAE (IRBF) |

### 2.3. Surrogate models

SAEAs have been widely applied to tackle high-dimensional continuous or combinatorial problems [25]. Surrogate models aim to discover main characteristics of a true model, and therefore, they can be adopted to substitute a part of the true model. These surrogate models use fewer computational resources to evaluate individuals and they can be used to prescreen candidate solutions. Liu et al. [26] propose a Gaussian process surrogate model to assist EAs, and prediction result of a surrogate model is adopted to search a promising subregion. Thus, more individuals evolve towards a promising subregion and further explore a search space. However, once a surrogate model is inaccurate, the whole search direction is misguided. To solve this problem, Pan et al. [27] adopt a surrogate model to predict dominance relationship between candidate solutions and reference ones rather than predicting their function values. In that case, it alleviates an approximation error of a surrogate model. It is worth noting that when using a surrogate model to prescreen offspring, additional methods are required to ensure accuracy due to an approximation error of a surrogate model. For example, Li et al. [28] select individuals whose predicted fitness values are better than previous ones for true model evaluations because those individuals have greater chances to locate better solutions. Thus, re-evaluating those individuals helps to improve accuracy. Wang et al. [29] put more emphasis on solutions with the highest predicted uncertainty defined by variance of prediction values at mid-late stages of evolution. Thus, re-evaluating them alleviates inaccuracy of surrogates to ensure accuracy of search.

Several machine learning methods can be adopted as surrogate models, among which Gaussian processes (GPs) [30], random forests (RFs) [31], radial basis functions (RBFs) [32] and support vector machines (SVMs) [33] are widely adopted. GPs have lower prediction accuracy and higher computation complexity for high-dimensional problems. Thus, it is more suitable for low-dimensional problems. RFs have poor performance when solving continuous problems because it cannot give continuous output. In addition, SVMs and RBFs are suitable for solving high-dimensional problems. However, SVMs have poor performance when processing large amount of data. Among existing models, RBFs work well for problems with a high-dimensional space given large amount of data. RBFs often adopt linear activation functions with low complexity, which are summarized as:

$$\begin{aligned} \hat{f}(X) &= \varepsilon^{\mathsf{T}}\psi = \sum_{i=1}^{m}\varepsilon_i\psi(X-c_i) \\ \psi_{ij} &= \psi(\|X^i-X^j\|) = \|X^i-X^j\|, i,j=1,\cdots,p \\ \varepsilon &= \psi^{-1}Y \end{aligned} \tag{3}$$

where $\hat{f}(X)$ denotes an RBF model, $X$ denotes an input data point, $m$ denotes the number of basis function centers, $c_i$ denotes a center point $i$, $p$ denotes the number of samples used to train RBF, $\psi(\cdot)$ denotes a radial basis function, $\|\cdot\|$ denotes Euclidean norm, and $\varepsilon$ denotes a vector of weight coefficients that are estimated by data samples $X$ and their corresponding function values $Y=(Y^1,Y^2,\cdots,Y^p)^{\mathsf{T}}$.

## 3. Overall framework

This section introduces an overall framework of STORA. For clarity, main notations of STORA are summarized in Table 1.

### 3.1. Self-adaptive teaching-learning-based optimizer

STLBO is proposed to speed up an optimization process and balance exploration and exploitation capabilities of TLBO for solving high-dimensional problems. In TLBO, a learning factor $T_F$ is a random integer ranging from $[1,2]$, which determines improvement of an average knowledge level. However, learners can obtain any proportion of knowledge, and $T_F$ does not have to be an integer. Bigger $T_F$ means that the average knowledge level increases faster, which leads to better exploration ability. On the contrary, lower $T_F$ means better exploitation ability. To enhance exploration ability of TLBO, $T_F$ ranges from $[2,3)$. Moreover, $T_F$ dynamically and linearly decreases as iterations continue in STLBO. In the early stage of STLBO, $T_F$ is assigned to a bigger value to enhance exploration capability. On the contrary, $T_F$ decreases to enhance exploitation capability for obtaining high-precision solutions in a later stage. $T_F$ is updated as:

$$T_F=\left(\frac{\hat{t}_1-t_1}{\hat{t}_1}\right)^2+2 \tag{4}$$

where $\hat{t}_1$ is the maximum number of iterations, and $t_1$ is current iteration.

Learners adjust their learning progress from the teacher according to their own current state of knowledge. When the level gap between learners and teacher is large, learners make great progress after learning. On the contrary, after learners reach a high level of knowledge, their progress becomes slower. To reflect this phenomenon, the progress is represented as a step size. The step size of each learner is shown as:

$$S^j(t)=\frac{\mathbf{f}(X^n(t))}{\mathbf{f}\left(X^j(t)\right)}, j\in[1,2,3,\cdots,N] \tag{5}$$

where $S^j(t)$ denotes a step size of individual $j$ in iteration $t$, $X^n(t)$ denotes the teacher in iteration $t$, and $X^j(t)$ denotes individual $j$ in iteration $t$.

TLBO considers what learners have learned is completely correct. If some of them trap into local optima, some knowledge they learned is in a wrong direction. If all learners gain this wrong knowledge, an optimization direction is misguided. As a result, a knowledge acquisition factor ($A$) is introduced to solve this problem. There is a certain probability that learners can fully grasp acquired knowledge controlled by $A$. Otherwise, learners cannot gain it. In addition, $A$ has two different values $A_1$ and $A_2$ in teaching and learning phases, respectively. $A_1$ is slightly bigger than $A_2$ because accuracy of knowledge imparted by the teacher is higher than that learned from other learners.

Knowledge level of the teacher is very important because other individuals are approaching it. Once the teacher (the currently best solution) falls into local optima, learners (solutions) may also be easy to fall into local optima. Therefore, in mid-late stages of an optimization process, the teacher and learners are disturbed to prevent them from falling into local optima. Specifically, some parts of the teacher's knowledge are randomly changed with a random learner. It is worth noting that to ensure accuracy of the final search, disturbance can only occur in mid-late stages. The teacher is disturbed as:

$$X_d^n(t)=X_d^n(t)+r\cdot\left(X_d^n(t)-X_d^j(t)\right) \tag{6}$$

where $X_d^n(t)$ denotes dimension $d$ of the current teacher in iteration $t$, $X_d^j(t)$ denotes dimension $d$ of individual $j$ in iteration $t$. $j$ denotes a random number in $\{1,2,3,\cdots,N\}$ and $j\neq n$. $r$ is a random number in $[0,1]$.

Thus, in the teaching phase of STLBO, individuals are updated as:

$$X_d^j(t+1)=A_1\cdot X_d^j(t)+S^j(t)\cdot\left(X_d^n(t)-T_F\cdot M_d(t)\right) \tag{7}$$

where $X_d^j(t+1)$ denotes dimension $d$ of individual $j$ in iteration $t+1$.

Knowledge level of learners is improved with the help of their peers in a learning stage of TLBO. In STLBO, in addition to learners learning from each other, they also learn through their teacher. In that case, learners progress under guidance of their peers and the teacher, which further speeds up an optimization process. In a learning phase of STLBO, if $f(X^j)<f(X^k)$, $X_d^j(t+1)$ is updated with (8); otherwise, it is updated with (9).

$$X_d^j(t+1)=A_2\cdot X_d^j(t)+S^j(t)\cdot\left(X_d^j(t)-X_d^k(t)\right)+S^j(t)\cdot\left(X_d^n(t)-T_F\cdot X_d^j(t)\right) \tag{8}$$

$$X_d^j(t+1)=A_2\cdot X_d^j(t)+S^j(t)\cdot\left(X_d^k(t)-X_d^j(t)\right)+S^j(t)\cdot\left(X_d^n(t)-T_F\cdot X_d^j(t)\right) \tag{9}$$

The overall process of STLBO is given in Algorithm 2.

### 3.2. Sparse autoencoder assisted by STLBO

We adopt an AE to compress a high-dimensional space into a reduced one for facilitating evolution. In STORA, its initial generations are conducted by STLBO for providing samples of training AE. As a population is evolving towards better regions, a trained AE can extract some important information of promising evolution directions to compress dimension of individuals. Besides, a small step of evolution in a compressed space represents a large one in an original space for speeding up an optimization process. When the termination condition is reached, AE is trained and used in the next stage. It is worth noting that low-dimensional offspring cannot be evaluated by functions because of dimensional mismatch. Each offspring has to be decoded to an original space for FEs.

SAE is a kind of autoencoders that achieve sparse effect by suppressing hidden layer neurons. Training data for AE is position information of a population, which is large and high-dimensional. Feature extraction of large samples by suppressing a part of hidden layer neurons has better performance [34]. Thus, adding extra sparse penalties can enhance ability of AE when dealing with high-dimensional problems. As shown in Fig. 1, SAE is parameterized by $(W,b)=(W^{(1)},W^{(2)},b^{(1)},b^{(2)})$. $W^{(i)}$ denotes a weight matrix between layers $i$ and $i+1$, and $b^{(i)}$ denotes a bias in layer $i$. We adopt the sigmoid logistic function as an activation function, *i.e.*,

$$\mathbf{f}(x)=\frac{1}{1+\mathbf{exp}(-x)} \tag{10}$$

**Algorithm 2** STLBO.

**Input:** Maximum number of iterations ($\hat{t}_1$), $A_1$ and $A_2$
**Output:** New population $P'$

```
1:  Initialize the population ($P$)
2:  for $t_1 = 1{:}\hat{t}_1$ do
3:      $T_F = \left(\frac{\hat{t}_1 - t_1}{\hat{t}_1}\right)^2 + 2$
4:      %Teaching phase
5:      if $t_1$ in mid-late stages then
6:          for $i = 1{:}N$ do
7:              Calculate a fitness value $f(X^i)$
8:              Choose the best individual as the teacher $X^n(t_1)$
9:              Randomly choose $d \in [1, D]$, $j \in [1, N]$, and $j \neq n$
10:             $X^n_d(t_1) = X^n_d(t_1) + r \cdot \left(X^n_d(t_1) - X^j_d(t_1)\right)$
11:         end for
12:     else
13:         for $i = 1{:}N$ do
14:             Calculate a fitness value $f(X^i)$
15:             Choose the best individual as the teacher $X^n(t_1)$
16:         end for
17:     end if
18:     for $j = 1{:}N$ do
19:         $S^j(t) = \frac{f(X^n(t_1))}{f(X^j(t_1))}$
20:         for $d = 1{:}D$ do
21:             $M_d(t_1) = \frac{1}{N}\sum_{j=1}^{N} X^j_d(t_1)$
22:             Update $X^j_d(t_1+1)$ with (7)
23:         end for
24:     end for
25:     %Learning phase
26:     for $j = 1{:}N$ do
27:         Randomly pick $k \in [1, N]$ where $k \neq j$
28:         for $d = 1{:}D$ do
29:             if $f(X^j) < f(X^k)$ then
30:                 Update $X^j_d(t_1+1)$ with (8)
31:             else
32:                 Update $X^j_d(t_1+1)$ with (9)
33:             end if
34:         end for
35:     end for
36: end for
37: return $P'$
```

Thus, given a single input $X$, its output is given as:

$$X' = h_{W,b}(X) = \mathbf{f}\left(W^{(2)}\mathbf{f}\left(W^{(1)}X + b^{(1)}\right) + b(2)\right) \tag{11}$$

where $\mathbf{f}(\cdot)$ denotes an activation function.

Given a training dataset $[X^i]$ where $i = 1, 2, \cdots, q$, where $q$ denotes the number of samples used to train SAE. To minimize discrepancy between input and output of SAE, a cost function of SAE is defined as:

$$\begin{aligned} J_s(W, b) = &\frac{1}{2q}\sum_{i=1}^{q} \|X^i - h_{W,b}(X^i)\|^2 + \frac{\lambda}{2}\|W\|^2 \\ &+ \beta \sum_{j=1}^{h} \mathbf{KL}(\rho \| \hat{\rho}_j) \end{aligned} \tag{12}$$

The first term is the average sum-of-squares error that describes discrepancy over training data. The second term is a regularization term that decreases magnitude of weights where $\lambda$ is a coefficient of weight. The third term is sparsity penalty where $\rho$ is a sparse parameter that is generally close to zero. $\beta$ is a random variable in $[0, 1]$, which is used to control weight of sparse penalty. $h$ is the number of hidden neurons. $\hat{\rho}$ represents an average activation value of all neurons in a hidden layer of all training samples for ensuring sparsity, and it evolves toward $\rho$. $\mathbf{KL}(\cdot)$ denotes Kullback-Leibler divergence between two different distributions, *i.e.*, if $\rho = \hat{\rho}$, $\mathbf{KL}(\rho \| \hat{\rho}_j) = 0$.

## 3.3. STLBO-assisted improved radial basis function

Training time of surrogate models is usually long because of large amount of high-dimensional training data. In addition, the number of center points in traditional RBFs equals that of training samples, and it makes network overly complex. In this case, it leads to an overfitting problem of neural networks, which results in low prediction accuracy [35]. This work proposes an IRBF as a surrogate model to solve this problem. We extract characteristics of data to construct neural networks for simplifying a network

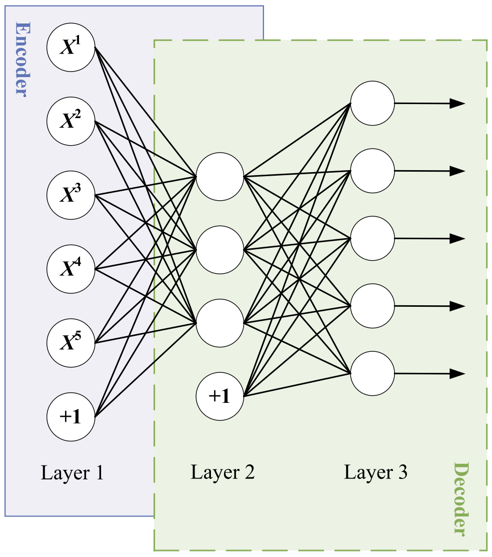


**Fig. 1.** Structure of SAE.

structure. To realize it, K-means clustering is widely adopted in data mining and compression. Given a set of data points and the number ($K$) of clusters, according to a specific distance function, the K-means algorithm divides data points into $K$ clusters repeatedly. It has advantages of fast convergence and strong local search ability [36]. In our structure, we take its advantages to select centers of a basis function, which locate important areas of an input space after the clustering, and improve prediction accuracy of the model. The number of center points in RBF is significantly reduced by the K-means algorithm, the model structure is simplified and it results to less training time.

However, the clustering result of the K-means algorithm is easily affected by initial clustering centers [37]. Different initial center points may cause different clustering results, and obtained results are unstable and volatile that affect prediction accuracy of constructed RBF. This work takes advantage of excellent global optimization ability of some metaheuristics to choose initial clustering centers. The goal of the metaheuristic in this problem is to find $K$ center points in samples to minimize the sum of distances from all points to a category to which they belong. Then, $K$ data points are initial centers of the K-means algorithm. Genetic learning particle swarm optimization (GLPSO) [38] is adopted in our structure. It combines PSO with GA in a cascade manner to avoid premature convergence of GA and enhance exploitation ability of PSO. Because of a selection operation in GA, selected particles are also of high quality. They can provide guidance for particles for increasing a searching rate of PSO. On the other hand, search experience of PSO provides promising information back to GA which helps GA to yield better solutions. In this way, GLPSO achieves high global optimization ability and robust performance.

Based on above analysis, we integrate GLPSO and the K-means algorithm work in a cascade manner. Specifically, GLPSO first finds initial clustering centers for the K-means algorithm, which divides a training dataset into $m$ groups, and $m$ equals the number of basis function centers. Then, RBF is built based on chosen center points. Fig. 2 shows a construction process of IRBF. The cooperation of GLPSO and the K-means algorithm chooses proper centers of RBF, which leads a model to have less training time and better prediction accuracy. In addition, IRBF is trained only when enough data is collected to ensure its accuracy. Before training IRBF, all positions and fitness values of individuals are collected. Then, IRBF is trained based on the collected data. The pseudo code of **IRBFtraining** is shown in Algorithm 3 where $p_i=[p_{i,1},p_{i,2},\cdots,p_{i,D}]$ denotes the locally best position of particle $i$, and $g=[g_1,g_2,\cdots,g_D]$ denotes the globally best position of the population. Besides, $v_i=[v_{i,1},v_{i,2},\cdots,v_{i,D}]$ and $x_i=[x_{i,1},x_{i,2},\cdots,x_{i,D}]$ represent the velocity and position of particle $i$.

## *3.4. Self-adaptive TLBO with improved RBF and SAE*

At the beginning of STORA, a population $P$ is initialized randomly in a decision space by Latin hypercube sampling (LHS) [39] for well covering a search space. Then, several generations of evolution are carried out by STLBO to collect data samples for training of SAE. Once a preset condition is reached, SAE is constructed based on accumulated data samples. After SAE training, a population is split into two sub-populations ($P_1$ and $P_2$) with a dynamic population allocation strategy introduced next. Then, $P_1$ and $P_2$ coevolve in a distributed manner to ensure diversity. $P_1$ is assisted by SAE to find promising solutions rapidly and $P_2$ is guided by STLBO (possibly assisted by IRBF) in an original space. The diversity of a population helps STORA to jump out of local optima that are imperative to an optimization process.

In SAE-assisted STLBO, $P_1$ is first encoded by trained SAE into a lower-dimensional space. Then, STLBO is adopted to generate offspring. Individuals have higher possibility to find promising offspring in a relatively low-dimensional space to speed up optimization. Due to dimensional mismatch, FEs cannot be completed in a low-dimensional space. After a decoding phase of the SAE, a population is in an original and high-dimensional space, its individuals can be directly evaluated by a fitness function. Finally,

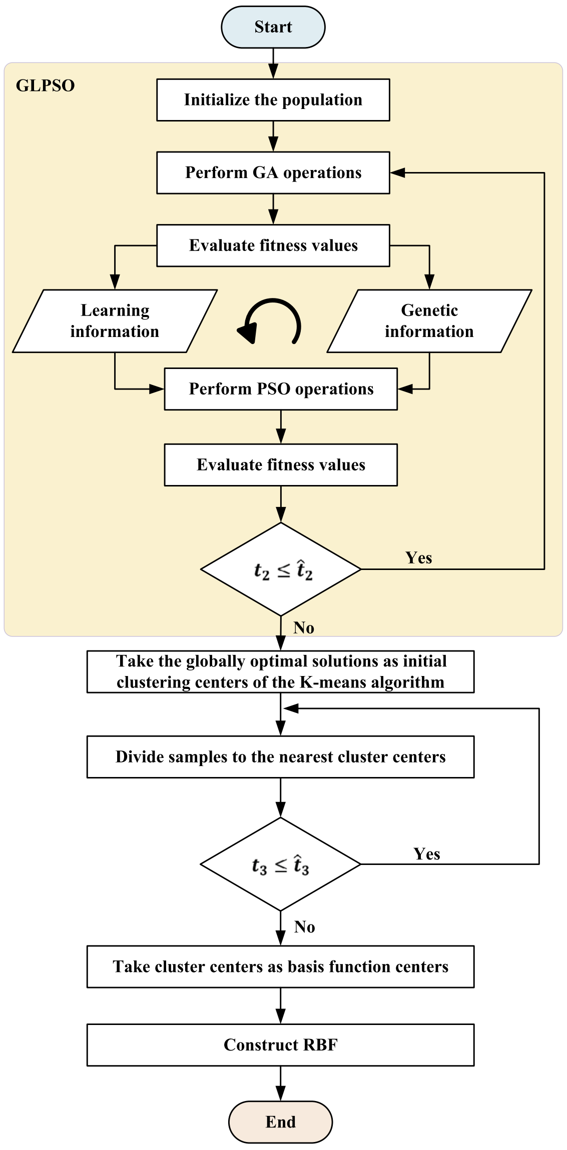

Start
GLPSO
Initialize the population
Perform GA operations
Evaluate fitness values
Learning information
Genetic information
Perform PSO operations
Evaluate fitness values
$t_2 \le \hat{t}_2$
Yes
No
Take the globally optimal solutions as initial clustering centers of the K-means algorithm
Divide samples to the nearest cluster centers
$t_3 \le \hat{t}_3$
Yes
No
Take cluster centers as basis function centers
Construct RBF
End

**Fig. 2.** Construction process of IRBF.

new $P_1$ is updated for the next generation. Furthermore, in IRBF-assisted STLBO, to ensure accuracy of IRBF, a training process of IRBF is only activated after enough samples are collected. $P_2$ evolves with STLBO before the activation of IRBF. In addition, all positions and their fitness values in previous iterations are stored in a database for later training of IRBF. Once an activation condition is met, IRBF is trained based on collected data samples, and it is adopted to prescreen individuals in the rest of an optimization process. After STLBO generates the offspring, positions of the offspring are input of IRBF that outputs predicted fitness values of those individuals. Furthermore, to ensure search accuracy, some individuals still need to be selected for true model evaluation. In STORA, individuals are sorted based on their predicted fitness values, the top $M$ individuals are selected for the true model evaluation because they have higher possibility to find optima quickly. Then, new $P_2$ is updated for the next generation. New $P_1$ and $P_2$ are combined together to form a new population $P$ after each iteration. The whole process continues until a termination condition is met. The flowchart of STORA is shown in Fig. 3 and its pseudo code is shown in Algorithm 4. Here, **SAEtraining(·)** and **IRBFtraining(·)** denote a training process of SAE and IRBF, respectively. In addition, **encode(·)** and **decode(·)** denote encoding and decoding phases of SAE, respectively. Finally, **STLBO(·)** means a process of generating offspring.

### *3.5. Dynamic population allocation strategy*

A dynamic population allocation strategy mainly includes two parts. The first one determines the number of individuals in each sub-population, which is called dynamic adjustment. The second one determines selected individuals assigned to each sub-population.

**Algorithm 3** IRBF training.

**Input:** Training database ($B$): a set of fitness values of all particles ($\digamma$) and a set of positions of all particles ($P$), maximum number of iterations in GLPSO ($\hat{t}_2$), maximum number of iterations in K-means ($\hat{t}_3$), inertia weight ($\omega$), stopping gap ($sg$), accelerate coefficient ($c$), and mutation rate ($p_m$)
**Output:** Trained IRBF model ($\gamma$): basis function center ($C$) and weight of output layer ($\Theta$)

```
1: while $t_2 \le \hat{t}_2$ do
2:     Evaluate a fitness value of each particle in $P$
3:     for $i=1:m$ do
4:         for $d=1:D$ do
5:             Randomly select a particle $k \in \{1,2,\cdots,m\}$
6:             if $f(p_i) < f(p_k)$ then
7:                 $o_{i,d} = r_d \cdot p_{i,d} + (1-r_d) \cdot g_d$
8:             else
9:                 $o_{i,d} = p_{k,d}$
10:            end if
11:            if rand$(0,1) < p_m$ then
12:                $o_{i,d} =$ rand$(\check{b}_d, \hat{b}_d)$
13:            end if
14:        end for
15:        if $f(o_i) < f(e_i)$ then
16:            $e_i = o_i$
17:        end if
18:        if $f(e_i)$ ceases improving for $sg$ generations then
19:            Select $e_j$ by %20$N$ tournament
20:            $e_i = e_j$
21:        end if
22:        for $d=1:D$ do
23:            $v_{i,d} = \omega \cdot v_{i,d} + c \cdot r_d \cdot (e_{i,d} - x_{i,d})$
24:            $x_{i,d} = x_{i,d} + v_{i,d}$
25:        end for
26:        Select and update $p_i$ and $g$
27:    end for
28:    $t_2 = t_2 + 1$
29: end while
30: Adopt $g$ as initialize centers of K-means
31: while $t_3 \le \hat{t}_3$ do
32:     Assign each point to a cluster to which it is closest based on the Euclidean distance
33:     Update a center of each cluster
34:     $t_3 = t_3 + 1$
35: end while
36: Adopt the clustering result as a basis function center $C$
37: Calculate an activation function value matrix $F$ of the population with (3)
38: Calculate the weight of output layer $\Theta = F^{-1}\digamma$
39: Return $\gamma$
```

Each individual is allocated to a unique sub-population based on their fitness value, and the number of finished iteration in a dynamic manner. At beginning of evolution, STORA aims to locate promising areas quickly. A sub-population $P_1$ is assisted by SAE that compresses an original decision space to a reduced one, which is beneficial to explore a promising region. As a result, more individuals are assigned to $P_1$ at beginning. On the other hand, as promising areas are gradually explored, further compression to lower dimensions may lose important area information and affect optimization accuracy. Accordingly, a sub-population $P_2$ evolves at an original space (possibly helped by IRBF) have more assigned individuals. As a result, the number of individuals in $P_1$ decreases and that of $P_2$ increases as the number of iterations increases. The individuals with worse fitness values are assigned to $P_1$ because they are difficult to evolve towards promising areas due to a high-dimensional search space. However, they may have higher possibility to produce better offspring in a compressed space. Two sub-populations are combined to a whole population again after each iteration. Details of dynamic adjustment are given as:

$$\begin{aligned} P_1 &= P \cdot \left( \frac{\hat{t}_5 - t_5}{\hat{t}_5} \right)^3 \\ P_2 &= P - P_1 \end{aligned} \tag{13}$$

where $\hat{t}_5$ is the number of maximum iterations for STORA and $t_5$ is the current iteration number. $(\frac{\hat{t}_5 - t_5}{\hat{t}_5})^3$ controls a decreasing rate of $P_1$ for SAE, and an increasing rate of $P_2$ for IRBF. Then, more individuals are assigned to $P_2$ in a later stage to further exploit more promising areas.

## 3.6. Computational complexity

Total computation expense ($T$) of STORA is given as:

$$T = T_{SAE} + T_{IRBF} + T_{FEs} \cdot FEs + \sum_{i=1}^{I} STLBO^i \tag{14}$$

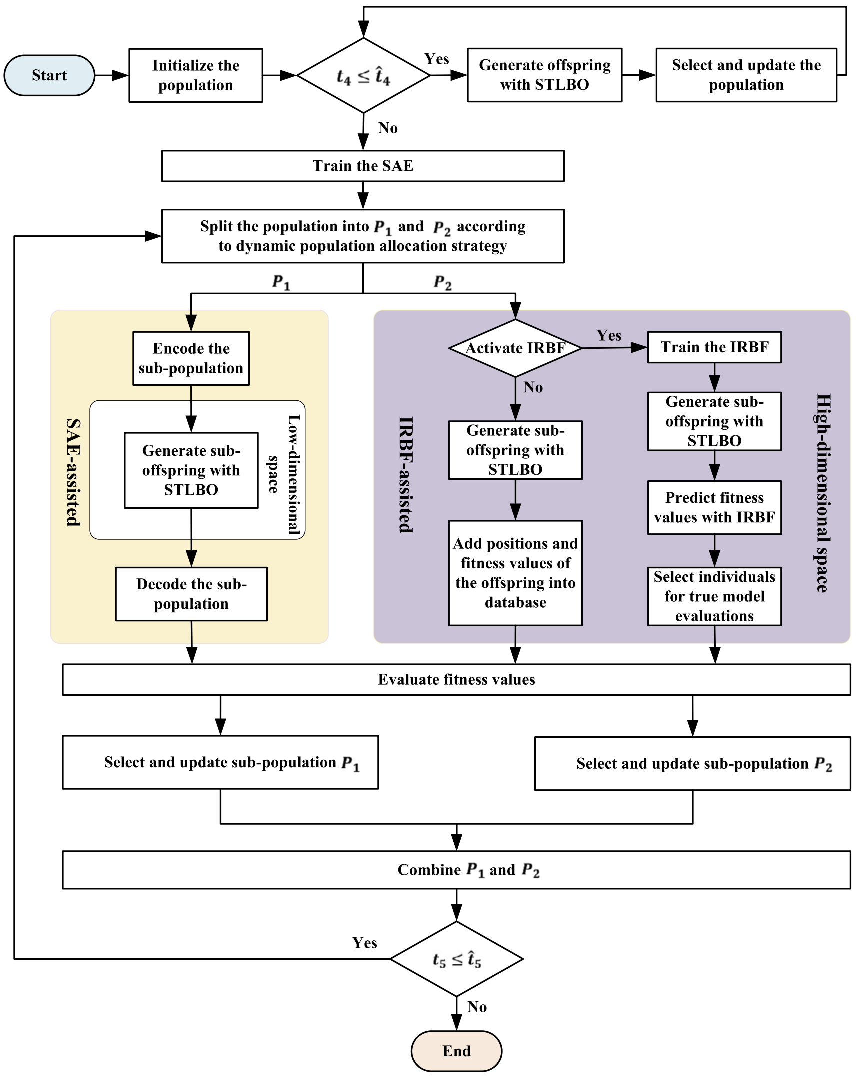

Start
Initialize the population
$t_4 \le \hat{t}_4$
Yes
Generate offspring with STLBO
Select and update the population
No
Train the SAE
Split the population into $P_1$ and $P_2$ according to dynamic population allocation strategy
$P_1$
$P_2$
SAE-assisted
Encode the sub-population
Low-dimensional space
Generate sub-offspring with STLBO
Decode the sub-population
IRBF-assisted
Activate IRBF
Yes
No
Train the IRBF
Generate sub-offspring with STLBO
Generate sub-offspring with STLBO
Predict fitness values with IRBF
Add positions and fitness values of the offspring into database
Select individuals for true model evaluations
High-dimensional space
Evaluate fitness values
Select and update sub-population $P_1$
Select and update sub-population $P_2$
Combine $P_1$ and $P_2$
Yes
$t_5 \le \hat{t}_5$
No
End

**Fig. 3.** Framework of STORA.

where $T_{SAE}$ denotes training time of SAE and $T_{IRBF}$ denotes that of IRBF. $T_{FEs}$ denotes time for each FE and $FEs$ denotes total number of FEs. $I$ denotes maximum number of iterations for STLBO, and $STLBO^i$ denotes time of STLBO in iteration $i$.

The computing complexity of STORA is given as follows. The complexity of SAE training is $O(qDE)$, where $E$ denotes the number of epochs. The complexity of IRBF training comprises the complexity of GLPSO, K-means and RBF. The complexity of GLPSO is $O[(D+N)\cdot\hat{t}_2]$, and that of K-means is $O(pm\hat{t}_3)$. The complexity of RBF is $O(p)$. $p$ is much larger than $D$ or $\hat{t}_2$, and therefore, the overall complexity of IRBF is mainly determined by that of K-means, *i.e.*, it is $O(pm\hat{t}_3)$. The complexity of generating offspring with STLBO is $O(IDN)$. Besides, sub-populations $P_1$ and $P_2$ co-evolve in a parallel way. According to above analysis, the complexity of STORA is $O(IDN)+O(qDE+pm\hat{t}_3)$.

It is worth noting that STORA adopts two neural networks (a sparse autoencoder and a radial basis function model). Although this work adopts a sparse autoencoder and an improved radial basis function model for reducing training time, they still need time for training, which inevitably increases time overhead of STORA. Moreover, it is shown that time complexity is higher than that of traditional evolutionary algorithms or swarm intelligence ones. As a result, when STORA is applied in real applications, it needs to be deployed in high-performance base stations to exert advantages of STORA.

## 4. Experimental results and discussion

### 4.1. Benchmark functions and comparative experiments

We compare STORA with two SI algorithms (TLBO [18] and GWO [8]) and one recently proposed algorithm (SAEO [10]), which is suitable to solve high-dimensional problems. We choose eight different benchmark functions adopted by many researchers [40].

**Algorithm 4** STORA.

```
Input: Maximum iterations of STLBO for SAE ($\hat{t}_4$), maximum number of iterations in STORA ($\hat{t}_5$), database to train SAE ($B_1$), database to train IRBF ($B_2$), $\hat{t}_2$, $\hat{t}_3$, $\omega$, $sg$, $c$, $p_m$ and $M$
Output: $x_{best}$ and $f_{best}$
1: Initialize $P$, $B_1$=∅, and $B_2$=∅
2: while $t_4 \le \hat{t}_4$ do
3:     $P' =$ STLBO($P$)
4:     Evaluate a fitness value of each individual in $P'$
5:     $B_1 = B_1 \cup P'$
6:     Select $P'$ as $P$ for the next generation
7:     $t_4 = t_4 + 1$
8: end while
9: $S =$ SAEtraining($B_1$)
10: while $t_5 \le \hat{t}_5$ do
11:     Split $P$ into $P_1$ and $P_2$ according to a dynamic population allocation strategy
12:     $\hat{P}_1 =$ encode($S, P_1$)
13:     $\hat{P}_1{}' =$ STLBO($\hat{P}_1$)
14:     $P_1{}' =$ decode($S, \hat{P}_1{}'$)
15:     Select $P_1{}'$ as $P_1$ for the next generation
16:     if an activation condition of IRBF is not met then
17:         $P_2' =$ STLBO($P_2$)
18:         Evaluate a fitness value of each individual in $P_2'$
19:         $B_2 = B_2 \cup (f(P_2'), P_2')$
20:         Select $P_2{}'$ as $P_2$ for the next generation
21:         $t_5 = t_5 + 1$
22:     else
23:         $\gamma$=IRBFtraining($B_2, \hat{t}_2, \hat{t}_3, \omega, sg, c, p_m$)
24:         $P_2' =$ STLBO($P_2$)
25:         $f(P_2') =$ IRBFpredict($\gamma, P_2'$)
26:         Sort individuals in $P_2'$ by their predicted fitness values in an ascending order
27:         Select top $M$ individuals for true model evaluations
28:         Select $P_2{}'$ as $P_2$ for the next generation
29:         $t_5 = t_5 + 1$
30:     end if
31:     $P = P_1 \cup P_2$
32:     Update $f_{best}$ and $x_{best}$
33: end while
34: Return $f_{best}$ and $x_{best}$
```

They include unimodal and multimodal functions with dimensions of 100 and 200. Details of benchmark functions are shown in Table 2. For benchmark algorithms, the population size is set to 50 and the number of re-evaluation in SAEO is set to five. For each benchmark function, 30 independent runs are performed and we record the best, the worst, the average values, and the standard deviations of optimal solutions. Moreover, a non-parametric statistical test (Wilcoxon-ranked sum test) is performed to prove that there are significant differences among STORA and compared algorithms. To be specific, each algorithm is statistically compared with STORA with a significance level of 0.05, *i.e.*, $\alpha = 0.05$. Besides, the logical *p*-value of 1 represents that STORA is significantly different from compared algorithms. The calculated *p*-value of each compared algorithm is shown in the supplementary file. For STORA, its population size is set to 50. $\hat{t}_2$ and $\hat{t}_3$ are both set to 100. $\hat{t}_4$ and $\hat{t}_5$ are set to 50 and 1000, respectively. $A_1$ and $A_2$ are set to 0.8 and 0.7, respectively. $h$ is set to 50. Parameters of GLPSO are set as suggested in [38], $\omega$=0.7298, $sg$=7, $c = 1.49618$ and $p_m$=0.01. $m$ is set to 125, and $M$ is set to five. IRBF is activated when 500 FEs are executed as recommended in [41] to balance training time and accuracy of IRBF. Moreover, sensitivity analysis of $A_1$, $A_2$ and $m$ are discussed in the supplementary file. All these algorithms are implemented in a computer with an Intel(R) Core(TM) i7 CPU 10750H at 2.60 GHz with 16 GB of RAM.

## 4.2. Experimental results

Table 3 provides statistical results of F1, F3, F5 and F6 after 1000 iterations with 100-D cases and Fig. 4 shows corresponding convergence curves. F2, F4, F7 and F8 with 100-D problems are shown in Fig. 1 and Table 1 in the supplementary file. It is shown that STORA's final fitness value is the least among these algorithms. Moreover, *p*-values for all compared algorithms are all one, which proves that STORA is significantly different from them. As for unimodal problems, *e.g.*, F1 in Fig. 4(a), STORA has faster optimization speed and better search result after 1000 iterations than other algorithms because of better exploration ability of STLBO and usage of SAE. Search results of GWO and TLBO are worse for F3 due to complexity of a high-dimensional space in Fig. 4(b). Compared with GWO and TLBO, SAEO can further explore the search space. However, STORA has better search performance over SAEO. Results of F2 and F4 have similar trend. For multimodal problems, *e.g.*, F5 in Fig. 4(c), TLBO and SAEO both find global optima due to great performance of TLBO. Among three algorithms including TLBO, SAEO, and STORA, STORA still has the steepest slope on its iterative curve. For F6 in Fig. 4(d), STORA rapidly converges to high-quality solutions within fewer iterations and it further exploits a search space to find better solutions. As shown in Table 3, STORA achieves better average results for all benchmark functions. In addition, standard deviation of STORA is particularly small, which indicates that STORA has stable performance and great robustness. Thus, STORA achieves the best search result for 100-D problems over other algorithms.

**Table 2**
Benchmark functions.

| Functions | $D$ | Range | Property | $\check{f}$ |
|---|---|---|---|---|
| $F1(x)=\sum_{i=1}^{N}\left(\lvert x_i+0.5\rvert\right)^2$ | 100 | $[-100,100]$ | Unimodal | 0 |
| $F2(x)=\sum_{i=1}^{N}\lvert x_i\rvert+\prod_{i=1}^{N}\lvert x_i\rvert$ | 100 | $[-10,10]$ | Unimodal | 0 |
| $F3(x)=\max_i\{\lvert x_i\rvert,1\leq i\leq N\}$ | 100 | $[-100,100]$ | Unimodal | 0 |
| $F4(x)=\left(\sum_{i=1}^{N} ix_i^4\right)+\textbf{random}[0,1)$ | 100 | $[-1.28,1.28]$ | Unimodal | 0 |
| $F5(x)=\sum_{i=1}^{N}[x_i^2-10\cos(2\pi x_i)+10]$ | 100 | $[-5.12,5.12]$ | Multimodal | 0 |
| $F6(x)=-20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right)-\exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right)+20+e$ | 100 | $[-32,32]$ | Multimodal | 0 |
| $F7(x)=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ | 100 | $[-600,600]$ | Multimodal | 0 |
| $F8(x)=418.9829D-\sum_{i=1}^{N}x_i sin\sqrt{\lvert x_i\rvert}$ | 100 | $[-500,500]$ | Multimodal | 0 |

**Table 3**
Results of F1, F3, F5 and F6 with 100-D problems.

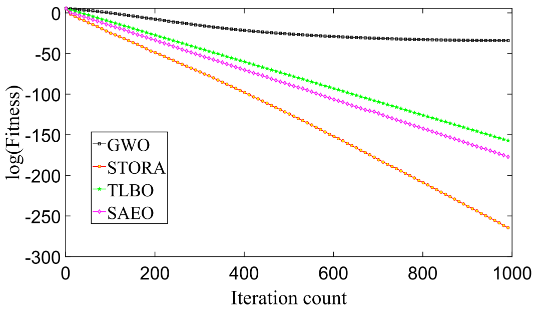
| Functions | Algorithms | Best | Mean | Worst | Std | $p$-value |
|---|---|---|---|---|---|---|
| F1 | **STORA** | $1.96\times10^{-271}$ | $\mathbf{1.90\times10^{-269}}$ | $9.98\times10^{-268}$ | $2.61\times10^{-270}$ | N/A |
| | SAEO | $5.26\times10^{-182}$ | $2.13\times10^{-179}$ | $6.62\times10^{-176}$ | $7.96\times10^{-179}$ | 1 |
| | GWO | $5.32\times10^{-35}$ | $1.53\times10^{-34}$ | $4.36\times10^{-34}$ | $6.64\times10^{-34}$ | 1 |
| | TLBO | $1.11\times10^{-168}$ | $3.73\times10^{-166}$ | $9.67\times10^{-158}$ | $2.62\times10^{-165}$ | 1 |
| F3 | **STORA** | $1.36\times10^{-133}$ | $\mathbf{3.22\times10^{-131}}$ | $7.32\times10^{-130}$ | $3.22\times10^{-130}$ | N/A |
| | SAEO | $3.26\times10^{-26}$ | $3.12\times10^{-25}$ | $5.52\times10^{-23}$ | $2.35\times10^{-25}$ | 1 |
| | GWO | $0.11\times10^{+00}$ | $0.25\times10^{+00}$ | $0.88\times10^{+00}$ | $0.63\times10^{+00}$ | 1 |
| | TLBO | $1.16\times10^{+00}$ | $9.65\times10^{+00}$ | $3.46\times10^{+01}$ | $2.63\times10^{+01}$ | 1 |
| F5 | **STORA** | $0.00\times10^{+00}$ | $\mathbf{0.00\times10^{+00}}$ | $0.00\times10^{+00}$ | $0.00\times10^{+00}$ | N/A |
| | **SAEO** | $0.00\times10^{+00}$ | $\mathbf{0.00\times10^{+00}}$ | $0.00\times10^{+00}$ | $0.00\times10^{+00}$ | 1 |
| | GWO | $1.12\times10^{-13}$ | $1.36\times10^{-13}$ | $3.11\times10^{-13}$ | $1.93\times10^{-13}$ | 1 |
| | **TLBO** | $0.00\times10^{+00}$ | $\mathbf{0.00\times10^{+00}}$ | $0.00\times10^{+00}$ | $0.00\times10^{+00}$ | 1 |
| F6 | **STORA** | $1.26\times10^{-15}$ | $\mathbf{2.46\times10^{-15}}$ | $5.31\times10^{-15}$ | $1.46\times10^{-15}$ | N/A |
| | SAEO | $3.25\times10^{-15}$ | $8.99\times10^{-15}$ | $9.34\times10^{-15}$ | $2.61\times10^{-15}$ | 1 |
| | GWO | $2.38\times10^{-14}$ | $3.41\times10^{-14}$ | $6.27\times10^{-14}$ | $3.61\times10^{-14}$ | 1 |
| | TLBO | $8.26\times10^{-15}$ | $9.78\times10^{-15}$ | $9.99\times10^{-15}$ | $2.31\times10^{-15}$ | 1 |

For 200-D problems, this work selects two unimodal problems (F2 and F3) and two multimodal problems (F6 and F8). Final statistical results and convergence curves of F3 and F8 are shown in Table 4 and Fig. 5, respectively. F2 and F6 with 200-D problems are shown in Fig. 2 and Table 2 in the supplementary file. Final results on $p$-value for all compared algorithms are all one, which proves that STORA is significantly different from them in 200-D problems. Moreover, final results on all algorithms are influenced by the increase of dimensions except STORA for F3. It is worth noting that STORA's performance is better for F3 with 200-D than that for F3 with 100-D. The reason is that SAE extracts more useful features of a 200-D search space, and better guides STORA to further explore it. Fig. 5(a) shows that benchmark algorithms cannot well explore a high-dimensional search space for F3, yet STORA shows the best performance on this problem. For F2, STORA has faster optimization speed and better search result than other benchmark algorithms. For F6, STORA further exploits a high-dimensional search space to find better solutions, and it also converges faster than benchmark algorithms. For F8 in Fig. 5(b), TLBO and GWO trap into local optima in an early stage due to a high-dimensional search space. SAEO and STORA maintain their performance because of AE. However, STORA still has faster convergence speed and better search results than SAEO. This is mainly caused by efficiency of SAE and exploitation ability of STLBO.

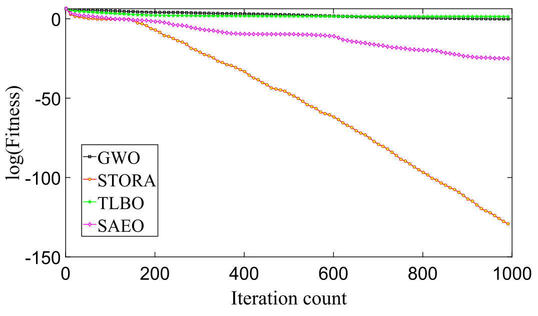
## 4.3. *Effectiveness of IRBF*

IRBF prescreens fitness values of individuals, and avoids unnecessary FEs to save more computational resources especially for those problems that have optimization potentials. This is significant to computationally expensive optimization problems. To verify effectiveness of IRBF, we show results of STORA and TLBO under the same computational resources (1000 FEs) for one unimodal problem (F2) and one multimodal problem (F6), which have different characteristics. Other results on benchmark functions show similar phenomenon.

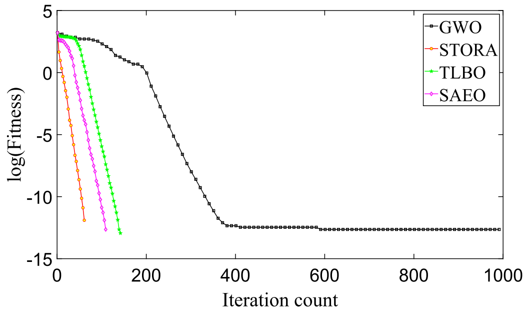
Fig. 6 shows results after activation of IRBF when 500 FEs are performed. It is shown that STORA has faster optimization speed than a case when 500 FEs are not reached. This is because some unnecessary FEs are saved for STLBO to further explore a high-dimensional search space. In addition, prediction accuracy of surrogate models is important. Large prediction errors lead to



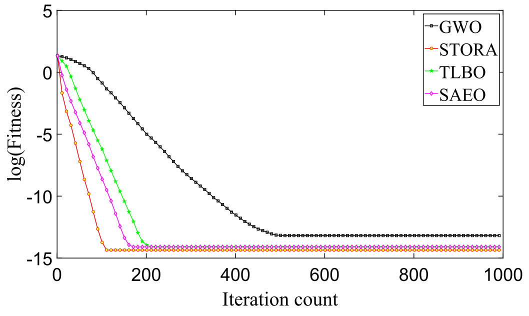
(a) F1



(b) F3



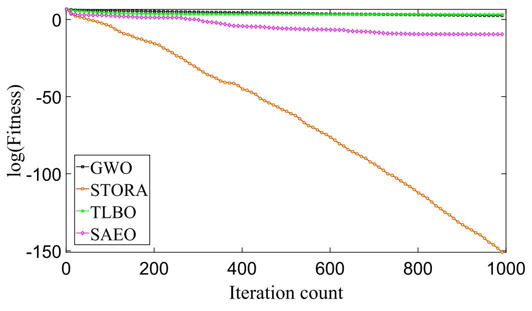
(c) F5



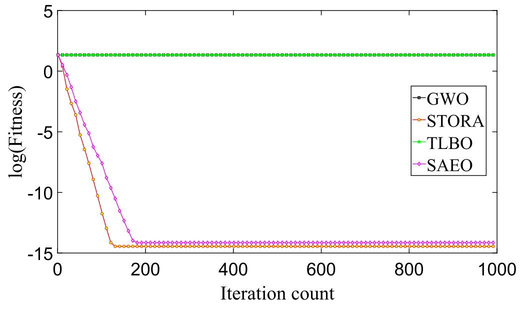
(d) F6

**Fig. 4.** Results of F1, F3, F5 and F6 with 100-D problems.

wrong individuals evaluated by the true model that influences accuracy of final search results. Moreover, if training time of surrogate models is too long, their use becomes meaningless. As a result, we compare prediction accuracy and training time of IRBF, RBF and other three common surrogate models including back propagation neural network (BPNN) [42], SVM [43], RF [44] under the same circumstance. We adopt the mean-square error (MSE) and the mean absolute error (MAE) to measure their prediction error, defined as:



(a) F3 (200D)



(b) F8 (200D)

**Fig. 5.** Results of F3 and F8 with 200-D problems.

**Table 4**
Results of F3 and F8 with 200-D problems.

| Functions | Algorithms | Best | Mean | Worst | Std | *p*-value |
|---|---|---|---|---|---|---|
| F3 | **STORA** | $1.23\times10^{-153}$ | $\mathbf{3.62\times10^{-153}}$ | $5.01\times10^{-153}$ | $1.61\times10^{-153}$ | N/A |
| | SAEO | $9.66\times10^{-13}$ | $1.97\times10^{-10}$ | $3.78\times10^{-09}$ | $9.93\times10^{-10}$ | 1 |
| | GWO | $1.13\times10^{+01}$ | $1.89\times10^{+01}$ | $2.32\times10^{+03}$ | $5.67\times10^{+01}$ | 1 |
| | TLBO | $4.24\times10^{+02}$ | $5.97\times10^{+02}$ | $8.47\times10^{+02}$ | $7.78\times10^{+02}$ | 1 |
| F8 | **STORA** | $2.15\times10^{-15}$ | $\mathbf{6.93\times10^{-15}}$ | $7.73\times10^{-15}$ | $6.47\times10^{-16}$ | N/A |
| | SAEO | $9.22\times10^{-15}$ | $7.11\times10^{-14}$ | $9.36\times10^{-14}$ | $2.24\times10^{-15}$ | 1 |
| | GWO | $1.99\times10^{+01}$ | $2.14\times10^{+01}$ | $2.45\times10^{+01}$ | $1.17\times10^{-01}$ | 1 |
| | TLBO | $1.98\times10^{+01}$ | $2.11\times10^{+01}$ | $2.94\times10^{+01}$ | $1.37\times10^{-02}$ | 1 |

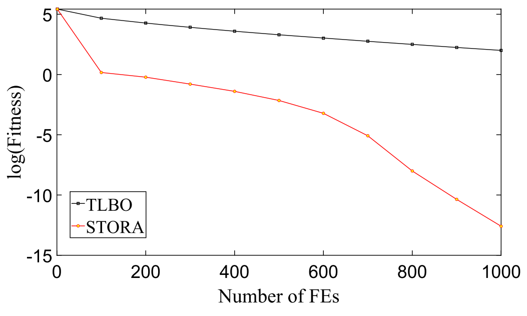
$$\begin{aligned}\text{MSE}&=\frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i-x_i)^2\\ \text{MAE}&=\frac{1}{n}\sum_{i=1}^{n}|\hat{x}_i-x_i|\end{aligned}\tag{15}$$

where $x_i$ denotes an actual fitness value of individual $i$ calculated by a true model, and $\hat{x}_i$ denotes its predicted fitness value. $n$ denotes the number of predicted values by IRBF. Besides, smaller metrics indicate better prediction accuracy.

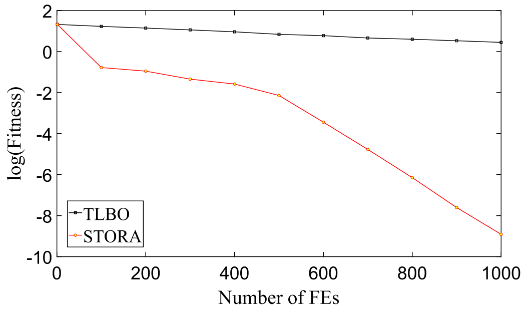
Table 5 shows that IRBF has the shortest training time especially compared with traditional RBF and BPNN. This is because the K-means algorithm reduces the number of centers of basis functions. Thus, the structure of the neural network is greatly simplified to significantly reduce training time of IRBF. Furthermore, RBF and BPNN choose each known sample in the training dataset as a center point, which leads a network overly complex and causes a problem of overfitting. Thus, RBF and BPNN have lower prediction accuracy than IRBF. GLPSO cooperates with the K-means algorithm locates centers of basis functions in promising areas of training data, which helps to enhance prediction ability of IRBF. The comparison of prediction results of different models demonstrates that prediction error of IRBF is the best among these methods.

## *4.4. Real-world computation offloading problem*

We apply STORA to solve a real-world computation offloading problem in mobile edge computing [45]. This problem considers to migrate a part of data processing of mobile applications from resource-constrained smart mobile devices (SMDs) to high-performing



(a) F2 (100D)



(b) F6 (100D)

**Fig. 6.** Effectiveness of IRBF in TLBO and STORA.

**Table 5**
Statistical results of surrogate models.

| Functions | Surrogate models | MSE | MAE | Time |
|---|---|---|---|---|
| F2 | **IRBF** | $\mathbf{2.2307\times10^{-13}}$ | $\mathbf{4.7230\times10^{-07}}$ | **3.21** |
| | RBF | $8.0570\times10^{-11}$ | $8.9761\times10^{-06}$ | 13.65 |
| | BPNN | $8.8152\times10^{-05}$ | $9.4632\times10^{-03}$ | 15.69 |
| | RF | $8.6475\times10^{-05}$ | $9.3243\times10^{-03}$ | 7.24 |
| | SVM | $2.5952\times10^{-05}$ | $5.1362\times10^{-03}$ | 4.59 |
| F6 | **IRBF** | $\mathbf{1.4719\times10^{-08}}$ | $\mathbf{3.8365\times10^{-03}}$ | **2.85** |
| | RBF | $1.8182\times10^{-03}$ | $1.4723\times10^{-01}$ | 14.29 |
| | BPNN | $3.7012\times10^{-01}$ | $6.0842\times10^{-01}$ | 18.63 |
| | RF | $2.2747\times10^{-02}$ | $1.5082\times10^{-01}$ | 6.41 |
| | SVM | $2.0841\times10^{-05}$ | $4.6219\times10^{-03}$ | 5.76 |

platforms in a network edge, which is known as computation offloading [46]. Therefore, the network edge can share part of computational pressures to enhance capabilities of SMDs. Furthermore, this problem focuses on applications that support data partitioning where the amount of data to be processed is known beforehand, *e.g.*, virus scan applications. In addition, this problem assumes that data of applications in SMDs can be partitioned into any size of subsets. Optimized decision variables include a computational speed of each SMD, its data transmission power, and task offloading ratio. Moreover, constraints include maximum latency for executing applications, maximum transmission power and maximum computational speed of each SMD. The objective of the problem is to minimize total energy consumed by all SMDs and edge servers while guaranteeing above-mentioned constraints for prolonging battery life. It is worth noting that this problem is a high-dimensional and single-objective problem that is fit for STORA. A constrained mixed-integer nonlinear program is formulated. A penalty function method is used to handle these constraints and integrate them into an unconstrained optimization problem. Each constraint is transformed into a non-negative penalty. For example, zero penalty means all the constraints are strictly met. Parameter setting of the problem is the same as that in [45].

We compare STORA with GWO, TLBO, SAEO and GLPSO by applying them to solve the real-world optimization problem. Fig. 7 shows their total energy consumption curves. It shows that total energy consumption of STORA is the least among all algorithms. In addition, STORA needs less than 150 iterations to converge to its final value, which is faster than its compared algorithms. Fig. 8 shows their final energy consumption given different numbers of SMDs. It is shown that total energy consumption of all algorithms increases with the number of SMDs. Among all algorithms, STORA achieves the least energy consumption as the number of SMDs increases. Fig. 9 shows their final energy consumption with different distances between each SMD and its nearest edge server. It is shown that final energy consumption of STORA is still the least among all algorithms as distances increase. Fig. 10 presents

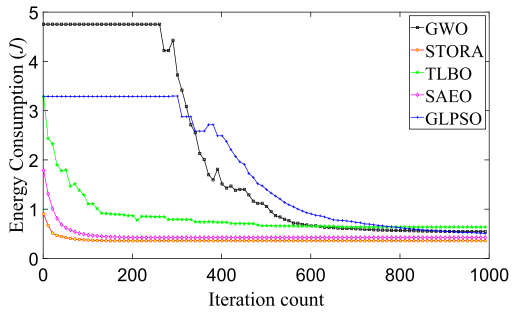


**Fig. 7.** Energy consumption in each iteration for each algorithm.

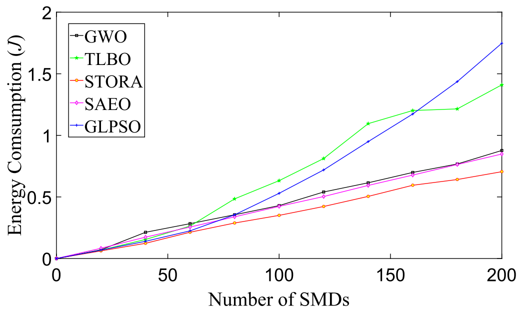


**Fig. 8.** Energy consumption v.s. number of SMDs for each algorithm.

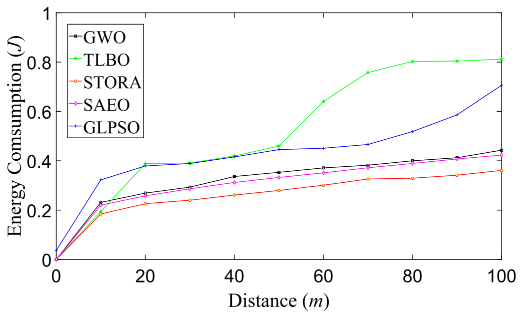


**Fig. 9.** Energy consumption v.s. distance for each algorithm.

comparison of penalty values of STORA and its peers. It is shown that penalty of STORA is quite small at beginning and it keeps the least during iterations. Furthermore, final penalty of STORA is zero, which proves that STORA produces a high-quality solution meeting all constraints in this problem. Fig. 11 shows final energy consumption of each SMD, which is a function of $\lambda$ under several simulation settings. The minimum energy consumption can be obtained by adjusting $\lambda$ and the final values of $\lambda$ under different conditions are all 0.6. Moreover, Fig. 12 shows final energy consumption of STORA given different numbers of SMDs and values of maximum latency ($L_{max}$). It demonstrates that STORA always finds final solutions under different latency requirements.

## 5. Conclusions

This work proposes a novel Self-adaptive Teaching-learning-based Optimizer with an improved Radial basis function model and a sparse Autoencoder (STORA) for high-dimensional problems. First, to trade off exploration and exploitation abilities, a Self-adaptive Teaching-Learning-Based Optimizer (STLBO) is designed to adjust parameters in a search process. Second, a sparse autoencoder (SAE) is adopted to speed up optimization in a high-dimensional space and give more possibility to worse individuals evolving towards promising areas. Third, an Improved Radial Basis Function model (IRBF) is designed as a surrogate model to find better solutions with fewer computing resources and less training time. Then, a dynamic population allocation strategy is designed to enhance integration of SAE and IRBF for improved performance of STORA. Finally, STORA is compared against its state-of-the-art peers on eight high-dimensional benchmark functions. Experimental results demonstrate that STORA yields the best search results with the least time among all compared algorithms. Furthermore, we apply STORA to solve a real-world computation offloading problem in mobile edge computing, and results demonstrate that STORA yields higher-quality solutions meeting all constraints than its typical peers. Our next work will extend it to solve many-objective optimization high-dimensional problems with discrete and

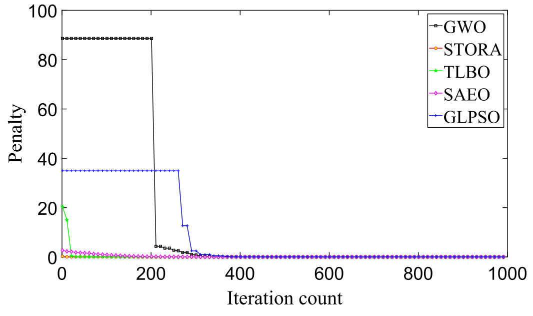


**Fig. 10.** Penalty in each iteration for each algorithm.

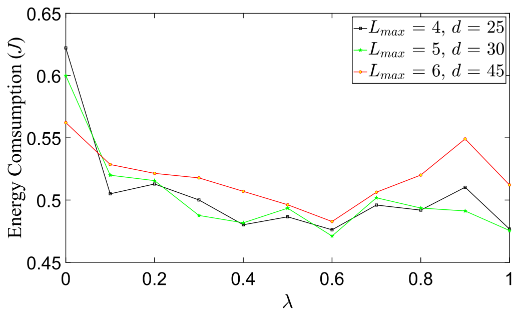


**Fig. 11.** Energy consumption v.s. $\lambda$ of STORA.

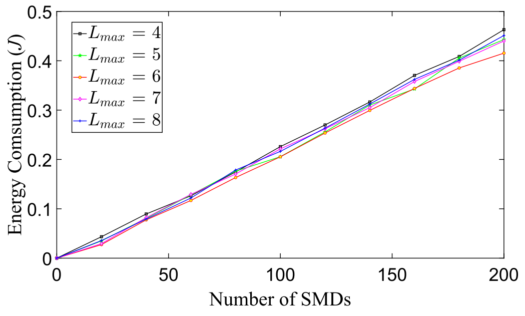


**Fig. 12.** Energy consumption with different values of $L_{max}$ and numbers of SMDs of STORA.

continuous parameters. In addition, other advanced surrogate models can be applied to better solve these problems, thus further improving performance of STORA.

## CRediT authorship contribution statement

**Jing Bi:** Conceptualization, Funding acquisition, Supervision, Writing – original draft. **Ziqi Wang:** Data curation, Formal analysis, Methodology, Software, Validation. **Haitao Yuan:** Investigation, Project administration, Resources, Visualization. **Jia Zhang:** Investigation, Writing – original draft, Writing – review & editing. **MengChu Zhou:** Writing – review & editing.

## 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.

## Data availability

Data will be made available on request.

## Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ins.2023.02.044.

## References

[1] T. Hassanzadeh, D. Essam, R. Sarker, EvoDCNN: an evolutionary deep convolutional neural network for image classification, Neurocomputing 488 (Jun. 2022) 271–283.

[2] M. Qin, R. Wang, Z. Shi, L. Liu, L. Shi, A genetic programming-based scheduling approach for hybrid flow shop with a batch processor and waiting time constraint, IEEE Trans. Autom. Sci. Eng. 18 (1) (Jan. 2021) 94–105.

[3] A. Favaro, A. Segato, F. Muretti, E.D. Momi, An evolutionary-optimized surgical path planner for a programmable bevel-tip needle, IEEE Trans. Robot. 37 (4) (Aug. 2021) 1039–1050.

[4] Y. Chi, Y. Xu, R. Zhang, Many-objective robust optimization for dynamic VAR planning to enhance voltage stability of a wind-energy power system, IEEE Trans. Power Deliv. 36 (1) (Feb. 2021) 30–42.

[5] W. Du, W. Zhong, Y. Tang, W. Du, Y. Jin, High-dimensional robust multi-objective optimization for order scheduling: a decision variable classification approach, IEEE Trans. Ind. Inform. 15 (1) (Jan. 2019) 293–304.

[6] S. Zhang, Y. Xia, Solving nonlinear optimization problems of real functions in complex variables by complex-valued iterative methods, IEEE Trans. Cybern. 48 (1) (Jan. 2018) 277–287.

[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (Apr. 2002) 182–197.

[8] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (10) (Mar. 2014) 46–61.

[9] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang, W.A. Chaovalitwongse, Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions, IEEE Trans. Evol. Comput. 23 (4) (Aug. 2019) 718–731.

[10] M. Cui, L. Li, M. Zhou, A. Abusorrah, Surrogate-assisted autoencoder-embedded evolutionary optimization algorithm to solve high-dimensional expensive problems, IEEE Trans. Evol. Comput. (May 2021) 1–15.

[11] H. Dong, P. Wang, X. Yu, B. Song, Surrogate-assisted teaching-learning-based optimization for high-dimensional and computationally expensive problems, Appl. Soft Comput. J. 99 (5) (Feb. 2021) 1–21.

[12] R. Lu, Y. Cai, J. Zhu, F. Nie, H. Yang, Dimension reduction of multimodal data by auto-weighted local discriminant analysis, Neurocomputing 461 (Oct. 2021) 27–40.

[13] X. Xu, T. Liang, J. Zhu, D. Zheng, T. Sun, Review of classical dimensionality reduction and sample selection methods for large-scale data processing, Neurocomputing 328 (Feb. 2019) 5–15.

[14] X. Yao, Q. Zhao, D. Gong, S. Zhu, Solution of large-scale many-objective optimization problems based on dimension reduction and solving knowledge guided evolutionary algorithm, IEEE Trans. Evol. Comput. (Sept. 2021) 1–15.

[15] Y. Tenne, K. Izui, S. Nishiwaki, Dimensionality-reduction frameworks for computationally expensive problems, in: IEEE Congress on Evolutionary Computation, Sept. 2010, pp. 1–8.

[16] H. Tao, L. Cheng, J. Qiu, V. Stojanovic, Few shot cross equipment fault diagnosis method based on parameter optimization and feature mertic, Meas. Sci. Technol. 33 (11) (Aug. 2022).

[17] X. Xin, Y. Tu, V. Stojanovic, H. Wang, K. Shi, S. He, T. Pan, Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems, Appl. Math. Comput. 412 (Jan. 2022).

[18] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput. Aided Des. 43 (5) (Mar. 2011) 303–315.

[19] F. Li, X. Cai, L. Gao, W. Shen, A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems, IEEE Trans. Cybern. 51 (3) (Mar. 2021) 1390–1402.

[20] J. Zabalza, J. Ren, J. Zheng, H. Zhao, C. Qing, Z. Yang, P. Du, S. Marshall, Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging, Neurocomputing 185 (Apr. 2016) 1–10.

[21] C.-Y. Chen, J.-S. Leu, S.W. Prakosa, Using autoencoder to facilitate information retention for data dimension reduction, in: 2018 3rd International Conference on Intelligent Green Building and Smart Grid, IGBSG, Apr. 2018, pp. 1–5.

[22] Z. Gao, C. Li, N. Liu, Z. Pan, J. Gao, Z. Xu, Large-dimensional seismic inversion using global optimization with autoencoder-based model dimensionality reduction, IEEE Trans. Geosci. Remote Sens. 59 (2) (Feb. 2021) 1718–1732.

[23] V. Miranda, J. da Hora Martins, V. Palma, Optimizing large scale problems with metaheuristics in a reduced space mapped by autoencoders—application to the wind-hydro coordination, IEEE Trans. Power Syst. 29 (6) (Nov. 2014) 3078–3085.

[24] M. Cui, L. Li, M. Zhou, An autoencoder-embedded evolutionary optimization framework for high-dimensional problems, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics, SMC, Mar. 2020, pp. 1046–1051.

[25] D. Lim, Y. Jin, Y. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comput. 14 (3) (Jun. 2010) 329–355.

[26] B. Liu, Q. Zhang, G.G.E. Gielen, A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, IEEE Trans. Evol. Comput. 18 (2) (Apr. 2014) 180–192.

[27] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, Y. Jin, A classification based surrogate-assisted evolutionary algorithm for expensive many objective optimization, IEEE Trans. Evol. Comput. 23 (1) (Mar. 2018) 74–88.

[28] F. Li, W. Shen, X. Cai, L. Gao, G.G. Wang, A fast surrogate-assisted particle swarm optimization algorithm for computationally expensive problems, Appl. Soft Comput. J. (Mar. 2020) 1–18.

[29] Y. Wang, D.-Q. Yin, S. Yang, G. Sun, Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints, IEEE Trans. Cybern. 49 (5) (May 2019) 1642–1656.

[30] J. Luo, A. Gupta, Y.-S. Ong, Z. Wang, Evolutionary optimization of expensive multiobjective problems with co-sub-Pareto front Gaussian process surrogates, IEEE Trans. Cybern. 49 (5) (May 2019) 1708–1721.

[31] X. Zhang, T. Yu, B. Yang, L. Jiang, A random forest-assisted fast distributed auction-based algorithm for hierarchical coordinated power control in a large-scale PV power plant, IEEE Trans. Sustain. Energy 12 (4) (Oct. 2021) 2471–2481.

[32] T. Henneron, A. Pierquin, S. Clenet, Surrogate model based on the POD combined with the RBF interpolation of nonlinear magnetostatic FE model, IEEE Trans. Magn. 56 (1) (Jan. 2020) 1–4.

[33] F. Shen, R. Yan, A new intermediate-domain SVM-based transfer model for rolling bearing RUL prediction, IEEE/ASME Trans. Mechatron. 27 (3) (Jun. 2022) 1357–1369.

[34] Y. Tian, C. Lu, X. Zhang, K.C. Tan, Y. Jin, Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks, IEEE Trans. Cybern. 51 (6) (Jun. 2021) 3115–3128.

[35] A. Ghasemian, H. Hosseinmardi, A. Clauset, Evaluating overfit and underfit in models of network community structure, IEEE Trans. Knowl. Data Eng. 32 (9) (Sept. 2020) 1722–1735.

[36] M.-S. Yang, K.P. Sinaga, A feature-reduction multi-view K-means clustering algorithm, IEEE Access 7 (2) (Mar. 2019) 114472–114486.
[37] H. Xiong, J. Wu, J. Chen, K-means clustering versus validation measures: a data-distribution perspective, IEEE Trans. Syst. Man Cybern. 39 (2) (Apr. 2009) 318–331.
[38] Y.-J. Gong, et al., Genetic learning particle swarm optimization, IEEE Trans. Cybern. 46 (10) (Oct. 2016) 2277–2290.
[39] P.S. Shin, S.H. Woo, Y. Zhang, C.S. Koh, An application of Latin hypercube sampling strategy for cogging torque reduction of large-scale permanent magnet motor, IEEE Trans. Magn. 44 (11) (Nov. 2008) 4421–4424.
[40] S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer function for binary particle swarm optimization, Swarm Evol. Comput. 9 (Apr. 2013) 1–14.
[41] H. Wang, Y. Jin, J. Doherty, Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems, IEEE Trans. Cybern. 47 (9) (Sept. 2017) 2664–2677.
[42] S. Siu, S. Yang, C. Lee, C. Ho, Improving the back-propagation algorithm using evolutionary strategy, IEEE Trans. Circuits Syst. 54 (2) (Feb. 2007) 171–175.
[43] S. Khandelwal, L. Garg, D. Boolchandani, Reliability-aware support vector machine-based high-level surrogate model for analog circuits, IEEE Trans. Device Mater. Reliab. 15 (3) (Sept. 2015) 461–463.
[44] H. Wang, Y. Jin, A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems, IEEE Trans. Cybern. 50 (2) (Feb. 2020) 536–549.
[45] Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing: partial computation offloading using dynamic voltage scaling, IEEE Trans. Commun. 64 (10) (Oct. 2016) 4268–4282.
[46] J. Zhao, Q. Li, Y. Gong, K. Zhang, Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks, IEEE Trans. Veh. Technol. 68 (8) (Aug. 2019) 7944–7956.

**Jing Bi** is currently an Associate Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She has over 80 publications including journal and conference papers. Her research interests include distributed computing, cloud computing, large-scale data analysis, machine learning and performance optimization. Dr. Bi was the recipient of the IBM Fellowship Award and the recipient of the Best Paper Award-Finalist in the 16th IEEE International Conference on Networking, Sensing and Control. She is now an Associate Editor of IEEE ACCESS. She is a senior member of the IEEE.

**Ziqi Wang** is currently a Master student in the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. Before that, he received his B.E. degree in Internet of Things from Beijing University of Technology in 2022. His research interests include cloud computing, task scheduling, intelligent optimization algorithms and machine learning.

**Haitao Yuan** received the Ph.D. degree in Computer Engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA in 2020. He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning and optimization algorithms. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC.

**Jia Zhang** received the PhD degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, Professor of Department of Computer Science in the Lyle School of Engineering at Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science. She is a senior member of the IEEE.

**MengChu Zhou** received his Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY in 1990 and then joined New Jersey Institute of Technology where he is now a Distinguished Professor. His interests are in Petri nets, automation, Internet of Things, and big data. He has over 900 publications including 12 books, 600+ journal papers (450+ in IEEE transactions), 28 patents and 29 book-chapters. He is Fellow of IFAC, AAAS, CAA and NAI.