2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Self-adaptive Teaching-learning-based Optimizer with Improved RBF and Sparse Autoencoder for Complex Optimization Problems

Jing Bi[1], Ziqi Wang[1], Haitao Yuan[2], Junfei Qiao[1], Jia Zhang[3] and MengChu Zhou[4]

Abstract— **Evolutionary algorithms are commonly used to solve many complex optimization problems in such fields as robotics, industrial automation, and complex system design. Yet, their performance is limited when dealing with high-dimensional complex problems because they often require enormous computational resources to yield desired solutions, and they 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 is designed to dynamically adjust parameters for balancing exploration and exploitation during its solution process. Then, a sparse autoencoder (SAE) is adopted as a dimension reduction method to compress search space into lower-dimensional one for more efficiently guiding population to converge towards global optima. Besides, an Improved Radial Basis Function model (IRBF) is designed as a surrogate model 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 it by comparing it with several state-of-the-art algorithms through six benchmark functions. We further test it by applying it to solve a real-world computational offloading problem.**

I. INTRODUCTION

Evolutionary algorithms (EAs) have been widely applied to solve different types of benchmarks and real-world engineering problems in a variety of fields, *e.g.*, computer vision [1], robots [2], cloud computing [3]–[5] and manufacturing scheduling problems [6]. Some practical problems have a large number of decision variables and can be characterized as high-dimensional problems [7]. These problems present a exponentially growing search space with many decision variables that bring big challenges for EAs to efficiently explore the search space. In other words, they often require a large number of function evaluations (FEs) to yield satisfactory solutions. However, FEs in many real-world problems can be computationally intensive or highly costly [8]. Moreover, some of EAs may easily trap into local optima when solving high-dimensional problems. As a result, it is important to balance exploration and exploitation abilities of EAs during their optimization process.

To solve high-dimensional and complex problems, a number of 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 [9]. A surrogate model can be employed to replace a part of a true model for evaluating individuals. It takes fewer computation resources than the true model. However, SAEAs bring additional surrogate models into the structure that also brings additional training time especially for a high-dimensional training set. Moreover, the 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.

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 [10], [11]. It aims to extract useful features of data to reduce the dimension of objective functions or the search space for reducing computational stress [12]. However, although the feature data is extracted under a specific dimension reduction method, some data including important information for the optimization process may be lost. As a result, it is highly important to choose a proper method and suitable time for dimension reduction [13].

Motivated by the 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. A Self-adaptive Teaching-learning-based Optimizer (STO) is proposed as EA 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. Similar to [14], [15], a sparse autoencoder (SAE) is adopted as a dimension reduction tool. SAE can well extract characteristics and the structure of samples with a large amount of high-dimensional data [16]. Moreover, an Improved Radial Basis Function model (IRBF) is proposed as a surrogate model in STORA to balance prediction accuracy and training time for reducing computational cost of SAEAs. Finally, a novel framework is proposed to integrate both dimension reduction and sur-

*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.

[1]J. Bi, Z. Wang and J. Qiao are with the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China. Email: bijing@bjut.edu.cn, wangziqi0312@163.com, junfeiq@bjut.edu.cn.

[2]H. Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. Email: yuan@buaa.edu.cn.

[3]J. Zhang is with the Department of Computer Science in the Lyle School of Engineering at Southern Methodist University, Dallas, TX 75205, USA. Email: jiazhang@smu.edu.

[4]M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA. Email: zhou@njit.edu.

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10160442

rogate models into STO to solve high-dimensional problems. 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 an energy-minimization problem in performing computation task offloading to demonstrate its superior performance.

II. PROPOSED FRAMEWORK

A. Self-adaptive Teaching-learning-based Optimizer

Our algorithm aims to find the global optima of a function $\mathbf{f}(x)$ where x is a vector of decision variables and $\mathbf{f}(x)$ is a function to evaluate an individual solution. In this work, STO is proposed as an EA for STORA. In STO, T_F is a learning factor and bigger T_F means better exploration ability. Moreover, T_F dynamically and linearly decreases as iterations continue. In the early stage of STO, T_F is assigned to a bigger value to enhance the exploration capability. On the contrary, T_F decreases to enhance the exploitation capability for obtaining high-precision solutions in the later stage. T_F is updated as:

$$T_F=\left(\frac{\hat{t}_1-t_1}{\hat{t}_1}\right)^2+2 \tag{1}$$

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

Learners adjust their learning progress from the teacher according to their own current state of knowledge. To reflect this phenomenon, the progress of each learner is represented as the step size. The step size of each learner is shown below.

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

where $S^j(t)$ denotes the 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.

Besides, a knowledge acquisition factor (A) is introduced to avoid STO falling into local optima. There is a certain probability that learners can fully grasp the acquired knowledge controlled by A. Otherwise, learners cannot gain this knowledge. 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 the accuracy of knowledge imparted by teachers is higher than that learned from others.

The knowledge level of the teacher is very important because other individuals are approaching it. Therefore, some parts of the teacher's knowledge are randomly disturbed with a random learner. Specifically, the teacher is disturbed as:

$$X_d^n(t)\leftarrow X_d^n(t)+r\cdot\left(X_d^n(t)-X_d^j(t)\right) \tag{3}$$

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 STO, individuals are updated as:

$$X_d^j(t+1)\leftarrow A_1\cdot X_d^j(t)+S^j(t)\cdot\left(X_d^n(t)-T_F\cdot M_d(t)\right) \tag{4}$$

where $X_d^j(t+1)$ denotes dimension d of individual j in iteration $t+1$. $M_d(t)$ denotes the mean position of the population at dimension d in iteration t.

The knowledge level of learners is improved with the help of their peers and the teacher in the learning phase of STO which further speeds up the optimization process. In the learning phase of STO, $X_d^j(t+1)$ is updated as:

$$\begin{aligned}X_d^j(t+1)\leftarrow A_2\cdot X_d^j(t)+S^j(t)\cdot\left(X_d^j(t)-X_d^k(t)\right)+\\ S^j(t)\cdot\left(X_d^n(t)-T_F\cdot X_d^j(t)\right)\end{aligned} \tag{5}$$

$$\begin{aligned}X_d^j(t+1)\leftarrow A_2\cdot X_d^j(t)+S^j(t)\cdot\left(X_d^k(t)-X_d^j(t)\right)+\\ S^j(t)\cdot\left(X_d^n(t)-T_F\cdot X_d^j(t)\right)\end{aligned} \tag{6}$$

If $\mathbf{f}(X^j)<\mathbf{f}(X^k)$, $X_d^j(t+1)$ is updated with (5); otherwise, it is updated with (6).

B. Sparse Autoencoder Training

SAE is a kind of autoencoders (AEs) that achieve sparse effect by suppressing hidden layer neurons. The training data for AE is the position information of the population, which is large and high-dimensional. The feature extraction of large samples by suppressing part of hidden layer neurons has better performance [17]. Thus, adding extra sparse penalties can enhance the ability of AE when dealing with high-dimensional problems. We adopt SAE to compress a high-dimensional space into a reduced one for facilitating the evolution. In STORA, its initial generations are conducted by STO for providing samples to train the SAE. As the population is evolving towards better regions, the trained SAE can extract some important information of promising evolution directions to compress the dimension of individuals. When the termination condition is reached, SAE is trained and used in the next stage. Moreover, the training time of AE is much lower than that of a surrogate model and can be neglected.

C. STO-assisted Improved Radial Basis Function

The number of center points in traditional RBFs equals that of the training samples which makes a neural network overly complex. In this case, it leads to long training time and overfitting problem of neural networks [18]. This work proposes an IRBF as a surrogate model to solve this problem. We extract characteristics of data to construct neural networks for simplifying the network structure. To realize it, we adopt the K-means algorithm to select centers of basis function, which locate important areas of the input space after the clustering. Thus, it can improve the prediction accuracy of the model [19], [20]. Moreover, the number of center points in RBF is significantly reduced by the K-means algorithm. Therefore, the model structure is simplified, thus demanding less training time.

However, the clustering result of the K-means algorithm is easily affected by the initial clustering centers [21]. This work takes advantage of the excellent global optimization ability of EAs to choose the initial clustering centers. The goal of EA in this problem is to find K center points in the samples to minimize the sum of distances from all points to the category to which they belong. Then, the K data points are initial centers of the K-means algorithm. Genetic learning particle swarm optimization (GLPSO) [22] is adopted in our structure. It combines particle swarm optimization (PSO) with genetic algorithm (GA) to avoid premature convergence of GA and enhance the exploitation ability of PSO. GLPSO achieves high global optimization ability and robust performance.

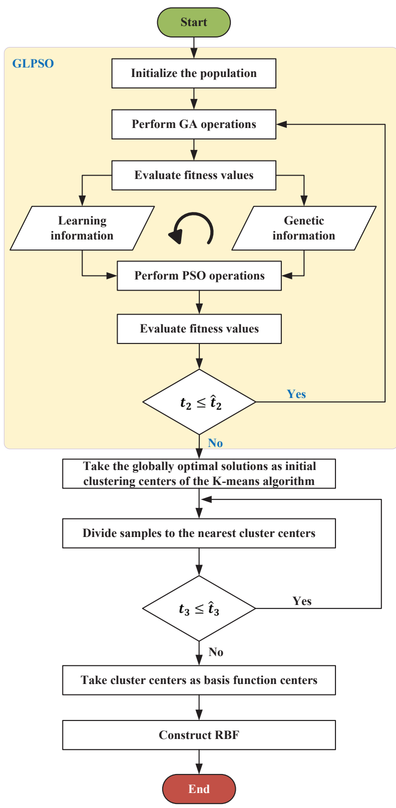


Fig. 1. Construction process of IRBF

Then, we integrate GLPSO and the K-means algorithm work in a cascade manner. Specifically, GLPSO first finds the initial clustering centers for the K-means algorithm, which divides the training dataset into m groups, and m equals the number of basis function centers. Then, RBF is built based on the chosen center points. The cooperation of GLPSO and the K-means algorithm chooses proper centers of RBF, which leads the model to have less training time and better prediction accuracy. The flowchart of IRBF is shown in Fig. 1, where $\hat{t}_2$ and $\hat{t}_3$ represent the maximum iterations of GLPSO and K-means, respectively. In addition, IRBF is trained only when enough data is collected to ensure the accuracy of the model [23]. Before the training of IRBF, all the positions and fitness values of individuals are collected. Then, IRBF is trained based on the collected data.

D. STO with Improved RBF and SAE

At the beginning of STORA, population P is initialized randomly in the decision space by Latin hypercube sampling (LHS) [24]. Then, several generations of evolution are carried out by STO to collect data samples for the training of SAE. Once the preset condition is reached, SAE is constructed based on the accumulated data samples. After the SAE training, the population is split into two sub-populations (P_1 and P_2) with the dynamic population allocation strategy to be 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 STO (possibly assisted by IRBF) in the original space. The diversity of the population helps STORA to jump out of local optima that are imperative to the optimization process.

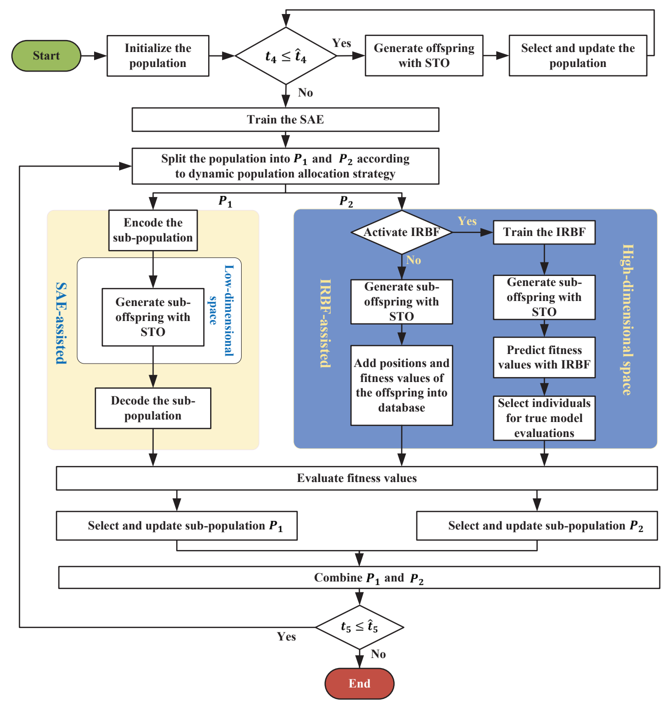


Fig. 2. Framework of STORA

In the SAE-assisted STO, P_1 is first encoded by the trained SAE into a lower-dimensional space. Then, STO is adopted to generate offspring. In that case, individuals have higher possibility to find promising offspring in the relatively low-dimensional space to speed up the optimization process. Due to the dimensional mismatch, FEs cannot be completed in the low-dimensional space. After the decoding phase of the SAE, the population is in the original and high-dimensional space, its individuals can be directly evaluated by the fitness function. Finally, new P_1 is updated for the next generation. Furthermore, in the IRBF-assisted STO, P_2 evolves with STO 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 the activation condition is met, IRBF is trained based on the collected data samples, and it is adopted to prescreen individuals in the rest of the optimization process. To be specific, after STO generates the offspring, the positions of the offspring are the input of IRBF that outputs the predicted fitness values of those individuals. Furthermore, to ensure the search accuracy, some individuals still need to be selected for the 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 the termination condition is met. The flowchart of STORA is shown in Fig. 2 and its pseudo codes are shown in Algorithm 1. Here, **SAEtraining(·)** and **IRBFtraining(·)** denote the training process of SAE and IRBF, respectively. In addition, **encode(·)** and **decode(·)** denote the encoding and decoding phases of SAE, respectively. Finally, **STO(·)** means the process of generating offspring.

E. Dynamic Population Allocation Strategy

The dynamic population allocation strategy includes two parts. The first one determines the number of individuals in each sub-population and the second one determines the selected individuals assigned to each sub-population. At the beginning of the evolution, STORA aims to locate promising areas quickly. The sub-population P_1 is assisted by SAE that compresses the original decision space to the reduced one, which is benefit to explore the promising region. As a result, more individuals are assigned to P_1 at the beginning. On the other hand, as promising areas are gradually explored, further compression to lower dimensions may lose the important area information and affect the optimization accuracy. Accordingly, the sub-population P_2 evolves at the original space (possibly helped by the IRBF) have more assigned individuals. Moreover, the individuals with worse fitness values are assigned to P_1 because they are difficult to evolve towards promising areas due to the high-dimensional search space. However, they may have higher possibility to produce better offspring in the compressed space. The two sub-populations are combined to a whole population again after each iteration. The dynamic adjustment is given as:

$$
\begin{aligned}
&P_1 = P \cdot \left(\frac{\hat{t}_5 - t_5}{\hat{t}_5}\right)^3 \\
&P_2 = P - P_1
\end{aligned}
\tag{7}
$$

where $\hat{t}_5$ is the number of maximum iterations for STORA and t_5 is the current iteration count. $(\frac{\hat{t}_5-t_5}{\hat{t}_5})^3$ controls the 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 the later stage to further exploit more promising areas.

Algorithm 1 STORA

Input: Maximum iterations of STO 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), and M
Output: x_{best} and f_{best}

Initialize P, $B_1=\varnothing$, and $B_2=\varnothing$
while $t_4 \leq \hat{t}_4$ **do**
  $P' = \mathbf{STO}(P)$
  Evaluate the fitness value of each individual in P'
  $B_1 = B_1 \cup P'$
  Select P' as P for the next generation
  $t_4 = t_4 + 1$
end while
$S = \mathbf{SAEtraining}(B_1)$
while $t_5 \leq \hat{t}_5$ **do**
  Split P into P_1 and P_2 according to the dynamic population allocation strategy
  $\hat{P}_1 = \mathbf{encode}(S, P_1)$
  $\hat{P}_1{}' = \mathbf{STO}(\hat{P}_1)$
  $P_1{}' = \mathbf{decode}(S, \hat{P}_1{}')$
  Select $P_1{}'$ as P_1 for the next generation
  if the activation condition of IRBF is not met **then**
    $P_2' = \mathbf{STO}(P_2)$
    Evaluate fitness value of each individual in P_2'
    $B_2 = B_2 \cup (f(P_2'), P_2')$
    Select $P_2{}'$ as P_2 for the next generation
    $t_5 = t_5 + 1$
  else
    $\gamma = \mathbf{IRBFtraining}(B_2)$
    $P_2' = \mathbf{STO}(P_2)$
    $f(P_2') = \mathbf{IRBFpredict}(\gamma, P_2')$
    Sort individuals in P_2' by their fitness values in an ascending order
    Select top M individuals for true model evaluations
    Select $P_2{}'$ as P_2 for the next generation
    $t_5 = t_5 + 1$
  end if
  $P = P_1 \cup P_2$
  Update f_{best} and x_{best}
end while
Return f_{best} and x_{best}

III. Experimental Results and Discussion

A. Benchmark Functions and Comparative Experiments

We compare STORA with two metaheuristic algorithms (teaching-learning-based optimization (TLBO) [25] and grey wolf optimizer (GWO) [26]) and one recently proposed algorithm (SAEO [27]), which is suitable to solve high-dimensional problems. We choose six different benchmark functions including unimodal and multimodal functions. Details of benchmark functions are shown in Table I. For benchmark algorithms, a population size is set to 50 and other parameters are used as their optimized values. For each benchmark function, 20 independent runs are performed and we record average values and standard deviations of optimal solutions. 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. The parameters of GLPSO are set as suggested in [22]. m is set to 125, and M is set to five. IRBF is activated when 500 FEs are executed as recommended in [28] to balance the training time and the accuracy of the IRBF. 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.

TABLE I

BENCHMARK FUNCTIONS

Functions	D	Range
$F1(x)=\sum_{i=1}^{N}(\|x_i+0.5\|)^2$	100	[-100,100]
$F2(x)=\sum_{i=1}^{N}\|x_i\|+\prod_{i=1}^{N}\|x_i\|$	100	[-10,10]
$F3(x)=\max_i\{\|x_i\|, 1\leq i\leq N\}$	100	[-100,100]
$F4(x)=\sum_{i=1}^{N}[x_i^2-10\cos(2\pi x_i)+10]$	100	[-5.12,5.12]
$F5(x)=-20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right)-\exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right)+20+e$	100	[-32,32]
$F6(x)=418.9829D-\sum_{i=1}^{N}x_i sin\sqrt{\|x_i\|}$	100	[-500,500]

B. Experimental Results

Table II provides statistical results of benchmark functions after 1000 iterations and Fig. 3 shows corresponding convergence curves. It is shown in Fig. 3 that STORA converges in fewer iterations because STO can easily find better regions in a lower-dimensional space because of the SAE. Moreover, the time cost of STORA per iteration is nearly the same as the compared algorithms. As for unimodal problems, *e.g.*, F1 in Fig. 3(a), STORA has faster optimization speed and better search result after 1000 iterations because of the better exploration ability of STO and the usage of SAE. The results of F2 and F3 have the similar trend. For multimodal problems, *e.g.*, F4 in Fig. 3(d), TLBO and SAEO both find the global optima due to the great performance of TLBO. Among these three algorithms, STORA still has the steepest slope on its iterative curve. For F5 and F6, STORA rapidly converges to high-quality solutions within fewer iterations and it further exploits the search space to find better solutions. As shown in Table II, STORA achieves better average results for all benchmark functions. In addition, the standard deviation of STORA is particularly small, which indicates that STORA has stable performance and great robustness. Thus, STORA achieves the best search result over its peers.

C. Real-world Computation Offloading Problem

We apply STORA to solve a real-world computation task offloading problem in an edge-computing-enabled large-scale factory [29]–[31], [39]. This problem considers to migrate a part of the data processing of mobile applications from resource-constrained smart mobile devices (SMDs) to high-performing platforms in a network edge, which is known as computation offloading [32]–[34]. The optimized decision variables include the computational speed of each SMD, its data transmission power, and task offloading ratio. Moreover, the 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 the total energy consumed by all SMDs and edge servers while guaranteeing above-mentioned

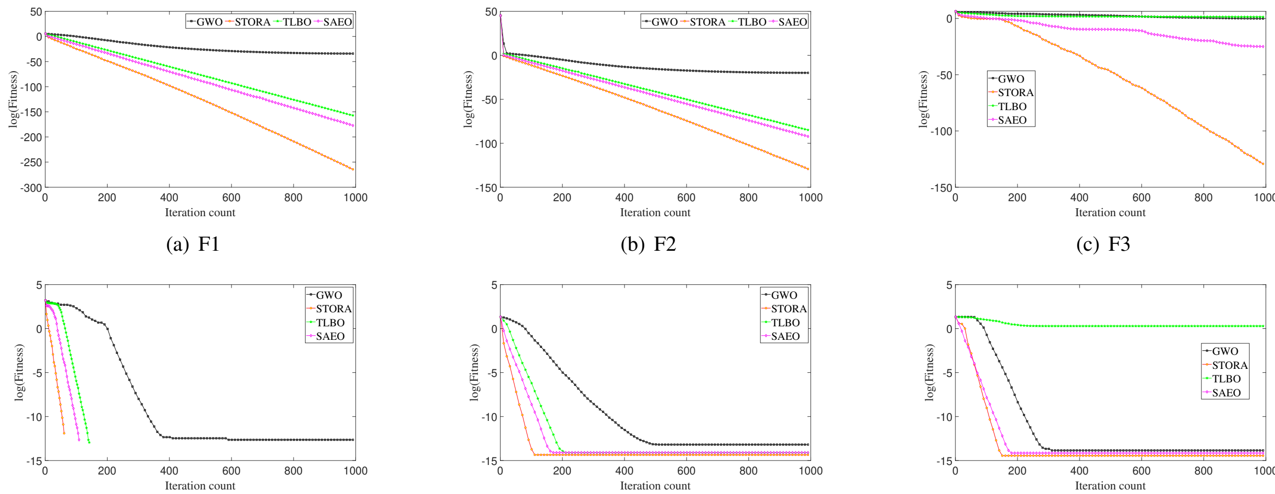
TABLE II

RESULTS OF BENCHMARK FUNCTIONS

Functions	Algorithms	Mean	Std
F1	**STORA**	$\mathbf{1.9062\times10^{-269}}$	3.2402×10^{-270}
	SAEO	4.2710×10^{-179}	6.7963×10^{-179}
	GWO	1.5261×10^{-34}	1.8280×10^{-34}
	TLBO	4.3572×10^{-166}	3.2102×10^{-166}
F2	**STORA**	$\mathbf{2.8283\times10^{-130}}$	3.7592×10^{-130}
	SAEO	1.3082×10^{-93}	5.0739×10^{-93}
	GWO	8.5767×10^{-21}	4.8600×10^{-21}
	TLBO	2.0665×10^{-86}	1.1691×10^{-86}
F3	**STORA**	$\mathbf{5.7237\times10^{-131}}$	1.2064×10^{-130}
	SAEO	2.6225×10^{-25}	1.1763×10^{-25}
	GWO	$0.2582\times10^{+00}$	$0.6141\times10^{+00}$
	TLBO	$9.6222\times10^{+00}$	$1.0436\times10^{+01}$
F4	**STORA**	$\mathbf{0.0000\times10^{+00}}$	$0.0000\times10^{+00}$
	SAEO	$\mathbf{0.0000\times10^{+00}}$	$0.0000\times10^{+00}$
	GWO	1.5438×10^{-13}	1.8907×10^{-13}
	TLBO	$\mathbf{0.0000\times10^{+00}}$	$0.0000\times10^{+00}$
F5	**STORA**	$\mathbf{3.4409\times10^{-15}}$	1.4584×10^{-15}
	SAEO	8.9936×10^{-15}	2.6100×10^{-15}
	GWO	3.4195×10^{-14}	3.5310×10^{-14}
	TLBO	9.6739×10^{-15}	2.1132×10^{-15}
F6	**STORA**	$\mathbf{3.5527\times10^{-15}}$	6.2970×10^{-16}
	SAEO	1.4211×10^{-14}	6.2232×10^{-15}
	GWO	7.6233×10^{-14}	$8.5868\times10^{+00}$
	TLBO	$2.1350\times10^{+01}$	3.6123×10^{-01}

constraints for prolonging the 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 [35], [36]. Each constraint is transformed into a non-negative penalty. For example, zero penalty means all the constraints are strictly met [37], [38]. The parameter setting is the same as [39].

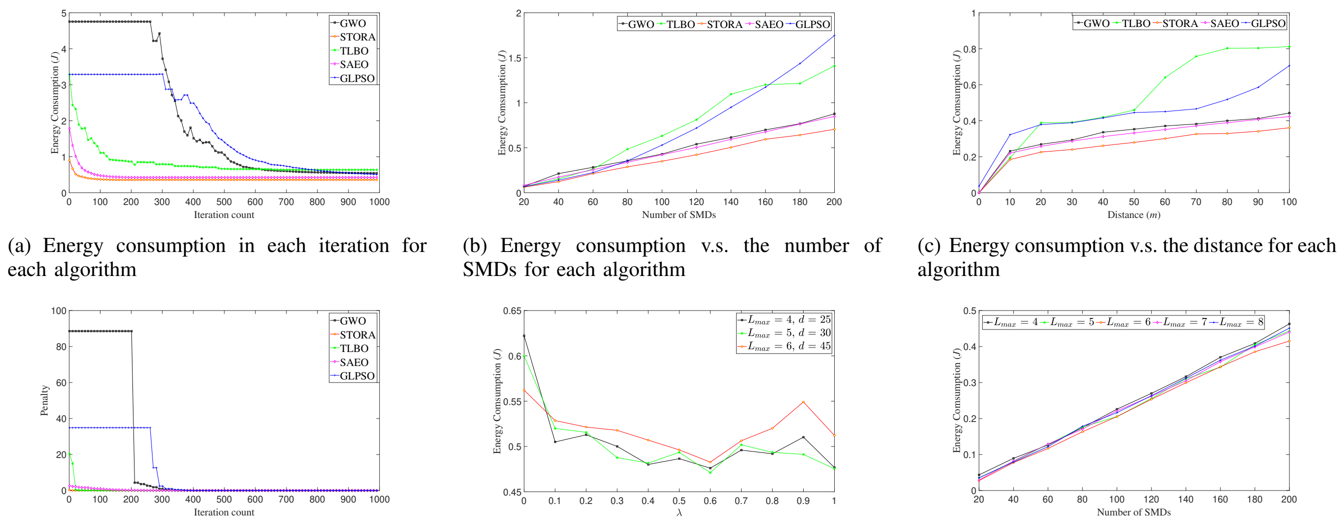
We compare STORA with GWO, TLBO, SAEO and GLPSO by applying them to solve the above problem. Fig. 4(a) shows that the total energy consumption of STORA is the least among all algorithms. In addition, STORA needs fewer than 150 iterations to converge to its final value, which is faster than its peers. Fig. 4(b) shows that the 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. 4(c) shows their final energy consumption with different distances between each SMD and its nearest edge server. It is shown that the final energy consumption of STORA is still the least among all algorithms as distances increase. Fig. 4(d) presents the comparison of penalty values of STORA and its peers. It is shown that the penalty of STORA is quite small at the beginning and it keeps the least during iterations. Furthermore, the final penalty value of STORA is zero, which proves that STORA produces a high-quality solution meeting all the constraints in this problem. Fig. 4(e) shows the final energy consumption of each SMD, which is a function of λ under several simulation settings. The minimum energy consumption can be obtained by adjusting λ and the final values of λ under different conditions are all 0.6. Moreover, Fig. 4(f) shows the final energy consumption of STORA



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

Fig. 3. Results of benchmark functions



(a) Energy consumption in each iteration for each algorithm

(b) Energy consumption v.s. the number of SMDs for each algorithm

(c) Energy consumption v.s. the distance for each algorithm

(d) Penalty in each iteration for each algorithm

(e) Energy consumption v.s. λ of STORA

(f) Energy consumption with different values of L_{max} and numbers of SMDs of STORA

Fig. 4. Results of the real-world problem

given different numbers of SMDs and values of maximum latency (L_{max}). It demonstrates that STORA always finds the final solution under different latency requirements.

IV. CONCLUSIONS

This work presents a Self-adaptive Teaching-learning-based Optimizer with an improved Radial basis function model and a sparse Autoencoder (STORA) for complex optimization problems. First, to trade off the exploration and exploitation abilities, a Self-adaptive Teaching-learning-based Optimizer (STO) is designed to adjust parameters in the 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 peers on six high-dimensional benchmark functions. Experimental results demonstrate that STORA yields the best search results with the least time among all compared algorithms. Then, we apply STORA to solve a real-world computational offloading problem in an edge computing environment, and results show that STORA yields higher-quality solutions meeting all constraints than its typical peers. Our next work should extend it to solve many-objective optimization high-dimensional problems with discrete and continuous parameters. In addition, other advanced surrogate models can be applied to better solve these problems, thus further improving performance of STORA.

References

[1] A. Razjigaev, A. K. Pandey, D. Howard, J. Roberts and L. Wu, "End-to-End Design of Bespoke, Dexterous Snake-Like Surgical Robots: A Case Study With the RAVEN II," *IEEE Transactions on Robotics*, pp. 1–14, Apr. 2022.

[2] A. Favaro, A. Segato, F. Muretti and E. D. Momi, "An Evolutionary-Optimized Surgical Path Planner for a Programmable Bevel-Tip Needle," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1039–1050, Aug. 2021.

[3] J. Bi, H. Yuan, K. Xu, H. Ma and M. Zhou, "Large-scale Network Traffic Prediction With LSTM and Temporal Convolutional Networks," *2022 International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, 2022, pp. 3865–3870.

[4] J. Zhai, J. Bi and H. Yuan, "Collaborative Computation Offloading for Cost Minimization in Hybrid Computing Systems," *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Prague, Czech Republic, 2022, pp. 1772–1777.

[5] H. Yuan, J. Bi, J. Zhang and M. Zhou, "Energy Consumption and Performance Optimized Task Scheduling in Distributed Data Centers," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 9, pp. 5506–5517, Sept. 2022.

[6] M. Qin, R. Wang, Z. Shi, L. Liu and L. Shi, "A Genetic Programming-Based Scheduling Approach for Hybrid Flow Shop With a Batch Processor and Waiting Time Constraint," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 94–105, Jan. 2021.

[7] L. Schramm and A. Boularias, "Learning-Guided Exploration for Efficient Sampling-Based Motion Planning in High Dimensions," *2022 International Conference on Robotics and Automation*, pp. 4429–4435, Jul. 2022.

[8] H. Dong, P. Wang, X. Yu and B. Song, "Surrogate-assisted Teaching-Learning-based Optimization for High-dimensional and Computationally Expensive Problems," *Applied Soft Computing Journal*, vol. 99, no. 5, pp. 1–21, Feb. 2021.

[9] M. Kalweit, G. Kalweit, M. Werling and J. Boedecker, "Deep Surrogate Q-Learning for Autonomous Driving," *2022 International Conference on Robotics and Automation*, pp. 1578–1584, Jul. 2022.

[10] R. Lu, Y. Cai, J. Zhu, F. Nie, H. Yang, "Dimension Reduction of Multimodal Data by Auto-weighted Local Discriminant Analysis," *Neurocomputing*, vol. 461, pp. 27–40, Oct. 2021.

[11] 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*, vol. 328, pp. 5–15, Feb. 2019.

[12] X. Yao, Q. Zhao, D. Gong and S. Zhu, "Solution of Large-scale Many-objective Optimization Problems Based on Dimension Reduction and Solving Knowledge Guided Evolutionary Algorithm," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, Sept. 2021.

[13] C. Chen, J. Leu and S. W. Prakosa, "Using Autoencoder to Facilitate Information Retention for Data Dimension Reduction," *2018 3rd International Conference on Intelligent Green Building and Smart Grid*, pp. 1–5, Jun. 2018.

[14] J. Bi, H. Yuan, J. Zhai, M. Zhou and H. V. Poor, "Self-adaptive Bat Algorithm With Genetic Operations," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[15] A. S. Nandan, S. Singh, R. Kumar and N. Kumar, "An Optimized Genetic Algorithm for Cluster Head Election Based on Movable Sinks and Adjustable Sensing Ranges in IoT-Based HWSNs," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5027–5039, Apr. 2022.

[16] A. R. Kalantarnezhad and J. Hamidzadeh, "MCRS-SAE: Multi-criteria Recommender System based on Sparse Autoencoder," *2022 12th International Conference on Computer and Knowledge Engineering*, pp. 117–122, Nov. 2022.

[17] Y. Tian, C. Lu, X. Zhang, K. Tan and Y. Jin, "Solving Large-Scale Multiobjective Optimization Problems With Sparse Optimal Solutions via Unsupervised Neural Networks," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3115–3128, Jun. 2021.

[18] A. Ghasemian, H. Hosseinmardi and A. Clauset, "Evaluating Overfit and Underfit in Models of Network Community Structure," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 9, pp. 1722–1735, Sept. 2020.

[19] M. Xu, G. Feng, Y. Ren and X. Zhang, "On Cloud Storage Optimization of Blockchain With a Clustering-Based Genetic Algorithm," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8547-8558, Sept. 2020.

[20] C. Baldassi, "Recombinator-k-Means: An Evolutionary Algorithm That Exploits k-Means++ for Recombination," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 991-1003, Oct. 2022.

[21] H. Xiong, J. Wu and J. Chen, "K-means Clustering Versus Validation Measures: A Data-Distribution Perspective," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 2, pp. 318–331, Apr. 2009.

[22] Y. Gong, J. Li, Y. Zhou, Y. Li, H. Chung, Y. Shi and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[23] Q. Lin, X. Wu, L. Ma, J. Li, M. Gong and C. A. C. Coello, "An Ensemble Surrogate-Based Framework for Expensive Multiobjective Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 631–645, Aug. 2022.

[24] P. S. Shin, S. H. Woo, Y. Zhang and C. S. Koh, "An Application of Latin Hypercube Sampling Strategy for Cogging Torque Reduction of Large-Scale Permanent Magnet Motor," *IEEE Transactions on Magnetics*, vol. 44, no. 11, pp. 4421–4424, Nov. 2008.

[25] R. V. Rao, V. J. Savsani, D. P. Vakharia, "Teaching–learning-based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems," *Computer-Aided Design*, vol. 43, no. 5, pp. 303–315, Mar. 2011.

[26] S. Mirjalili, S. M. Mirjalili, A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, no. 10, pp. 46–61, Mar. 2014.

[27] M. Cui, L. Li, M. Zhou and A. Abusorrah, "Surrogate-assisted Autoencoder-embedded Evolutionary Optimization Algorithm to Solve High-dimensional Expensive Problems," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, May 2021.

[28] H. Wang, Y. Jin and J. Doherty, "Committee-Based Active Learning for Surrogate-Assisted Particle Swarm Optimization of Expensive Problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2664–2677, Sept. 2017.

[29] S. Jošilo and G. Dán, "Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1507–1520, Dec. 2021.

[30] H. Yuan and M. Zhou, "Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.

[31] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim and H. Ning, "A Novel Framework for Mobile-Edge Computing by Optimizing Task Offloading," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 13065–13076, Aug. 2021.

[32] A. Yousafzai, I. Yaqoob, M. Imran, A. Gani and R. Md Noor, "Process Migration-Based Computational Offloading Framework for IoT-Supported Mobile Edge/Cloud Computing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4171–4182, May 2020.

[33] P. X. Nguyen, *et al.*, "Backscatter-Assisted Data Offloading in OFDMA-Based Wireless-Powered Mobile Edge Computing for IoT Networks," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9233–9243, Jun. 2021.

[34] J. Zhao, Q. Li, Y. Gong and K. Zhang, "Computation Offloading and Resource Allocation for Cloud Assisted Mobile Edge Computing in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[35] H. Yuan, J. Bi and M. Zhou, "Energy-Efficient and QoS-Optimized Adaptive Task Scheduling and Management in Clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1233–1244, Apr. 2022.

[36] H. Yuan, J. Bi and M. Zhou, "Multiqueue Scheduling of Heterogeneous Tasks With Bounded Response Time in Hybrid Green IaaS Clouds," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5404–5412, Oct. 2019.

[37] H. Yuan, J. Bi, M. Zhou, Q. Liu and A. C. Ammari, "Biobjective Task Scheduling for Distributed Green Data Centers," *IEEE Trans. on Automation Science and Engineering*, vol. 18, no. 2, pp. 731–742, Apr. 2021.

[38] H. Yuan, J. Bi and M. Zhou, "Geography-Aware Task Scheduling for Profit Maximization in Distributed Green Data Centers," *IEEE Trans. on Cloud Computing*, vol. 10, no. 3, pp. 1864–1874, Sept. 2022.

[39] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Trans. on Communications*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.