Complex & Intelligent Systems (2023) 9:6875–6890
https://doi.org/10.1007/s40747-023-01113-4

ORIGINAL ARTICLE

A pairwise ranking estimation model for surrogate-assisted evolutionary algorithms

Tomohiro Harada[1]

Received: 31 October 2022 / Accepted: 16 May 2023 / Published online: 5 June 2023
© The Author(s) 2023

Abstract
Surrogate-assisted evolutionary algorithms (SAEAs) have attracted considerable attention for reducing the computation time required by an EA on computationally expensive optimization problems. In such algorithms, a surrogate model estimates the solution evaluation with a low computing cost and is used to obtain promising solutions to which the accurate evaluation with an expensive computation cost is then applied. This study proposes a novel pairwise ranking surrogate model called the Extreme Learning-machine-based DirectRanker (ELDR). ELDR integrates two machine learning models: extreme learning machine (ELM) and DirectRanker (DR). ELM is a single-layer neural network capable of fast learning, whereas DR uses pairwise learning to rank using a neural network developed mainly for information retrieval. To investigate the effectiveness of the proposed surrogate model, this study first examined the estimation accuracy of ELDR. Subsequently, ELDR was incorporated into a state-of-the-art SAEA and compared with existing SAEAs on well-known real-valued optimization benchmark problems. The experimental results revealed that ELDR has a high estimation accuracy even on high-dimensional problems with a small amount of training data. In addition, the SAEA using ELDR exhibited a high search performance compared with other existing SAEAs, especially on high-dimensional problems.

Keywords DirectRanker · Evolutionary algorithms · Extreme learning machine · Pairwise ranking · Surrogate model

Introduction

Evolutionary Algorithms (EAs) are population-based optimization methods that are applied to many real-world problems. However, because an expensive fitness evaluation is often required in real-world applications owing to simulations or complex numerical calculations, the computational cost of EAs is typically high. To reduce the computing cost of EAs, surrogate-assisted EA (SAEA) has been studied [1, 2] and applied to real-world applications such as aerospace engineering [3, 4], vehicle design [5], and manufacturing process optimization [6]. An SAEA utilizes a surrogate model that estimates fitness instead of a computationally expensive fitness function and finds promising solutions for actual fitness evaluation. Because surrogate estimation is computationally cheaper than actual fitness evaluation, the execution time of the SAEA is reduced compared to that of conventional EAs.

Three types of surrogate models are used in SAEAs [7]asfollows: (1) a regression model that directly estimates the fitness evaluation, (2) a classification model that estimates the relative acceptability compared to a reference value rather than a fitness evaluation, or (3) a ranking model that estimates the relative superiority of solutions compared to each other. This study proposes a novel ranking-based surrogate model that exploits the fact that general EAs can perform parent selection and survival selection based on the superiority of solutions. Specifically, this paper proposes the Extreme Learning-machine-based DirectRanker (ELDR), which combines an extreme learning machine (ELM) [8, 9]—a type of single-layer neural network (NN) with fast learning capability—with DirectRanker (DR) [10]—a pairwise ranking method based on a NN developed primarily for information retrieval.

To investigate the effectiveness of ELDR, its prediction accuracy was first analyzed by comparing it with other surrogate models used in previous research. Then, to confirm its capability on an SAEA, it was incorporated into a state-of-

✉ Tomohiro Harada
harada@tmu.ac.jp

[1] Faculty of System Design, Tokyo Metropolitan University, Hino, Tokyo, Japan

the-art SAEA—specifically, surrogate-assisted hybrid optimization (SAHO) [11]—and the consequent ELDR-SAHO was compared with recent SAEA methods, including SAHO.

The remainder of this paper is organized as follows: "Related Work" discusses related work on SAEAs. "Proposed Method" describes the proposed new pairwise ranking surrogate model, ELDR, in detail. "Example Application of ELDR to SAEA: ELDR Surrogate-Assisted Hybrid Optimization" presents the details of ELDR-assisted SAEA. In particular, this study employed SAHO—a state-of-the-art SAEA. "Preliminary Experiment: Accuracy of ELDR analyzes the parameter sensitivity of ELDR and compares its estimation accuracy with other conventional surrogate models used in existing SAEAs. Section "Numerical Experiments" outlines the numerical experiments conducted to investigate the effectiveness of ELDR-assisted SAHO and analyzes the results obtained. Finally, "Conclusion and Future Work" presents concluding remarks and outlines possible future work.

Related work

This section reviews surrogate models used in SAEAs in previous studies. This study focuses primarily on single-objective real-valued optimization problems but also refers to some studies on multi-objective optimization and discrete optimization, including genetic programming.

In previous studies, the mainstream approach has been SAEAs using regression models. In particular, most previous works [12–15] used the radial basis function (RBF) [16] as a surrogate model. Other SAEAs that use the Kriging model [17, 18], Gaussian process regression [19], and the nearest neighbor method [20] have also been proposed. Pavelski et al. proposed ELMOEA/D [21], a surrogate-assisted multi-objective evolutionary algorithm that uses ELM as a regression surrogate model.

Recent studies have proposed classification-based SAEAs. Pan et al. [22] proposed a classification-based SAEA (CSEA) that learns the dominance relationship between candidate solutions and reference solutions using an artificial neural network (ANN) [23]. Sonoda et al. [24] used a support vector machine (SVM) [25] to optimize a decomposition-based MOEA (MOEA/D) to classify whether an offspring solution is better than its parent solution for each aggregation function. Wei et al. [26] proposed a classifier-assisted level-based learning swarm optimizer (CA-LLSO) that uses a multiclass gradient boosting classifier (GBC) [27] to classify the swarm (population) into different levels for applying LLSO [28].

In contrast to the regression and classification models, few studies have reported ranking-based SAEAs. Ranking models are usually applied for pre-selection [7], for example, to estimate the population ranking in CMA-ES [29] in the works of Runarsson [30] and Loshchilov et al. [31]. Lu et al. [32] proposed Differential Evolution (DE) [33] with surrogate-assisted self-adaptation (DESSA), which uses ranking SVM (RankSVM) [34] to select the most promising trial vector. In recent years, Hao et al. [35] proposed a ranking model using a vector concatenating two solutions as input and demonstrated that the ranking model shows higher estimation accuracy than regression and classification models with a small number of training samples. Another work by Hao et al. [36] proposed a similar ranking model based on a neural network that estimates the dominance relationship for solving multi-objective optimization problems. However, there is limited research on high-performance SAEAs using rank models compared with SAEAs using regression and classification models.

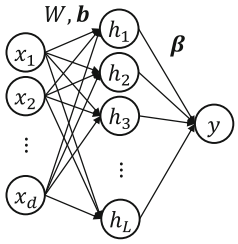


Fig. 1 A topology of ELM

Because the ranking model can estimate the dominance relation between solutions rather than the objective function value, it can be applied to a broader range of problem domains where regression and classification models are not applicable. One example is constrained optimization problems for the definition of parental or survival selection methods (e.g., feasibility rules [37] or the ϵ-constrained method [38]) without estimating all the constraint values. Another example is the subject human evaluation in interactive EAs [39], which generally uses relative rather than quantitative assessment. Therefore, developing a useful ranking surrogate model is more effective for applying SAEA to a different problem domain than regression and classification models.

Proposed method

This section first introduces ELM [8, 9] and DR [10], which are the components of ELDR. Then, the operation of ELDR, which integrates these two techniques, is explained.

Extreme learning machine

ELM is a single-layer NN. The topology of ELM is illustrated in Fig. 1. ELM is formed with a fully connected NN. Given a d-dimensional input $\boldsymbol{x}$, ELM with L hidden neurons calculates the output y as

$$\begin{aligned} y &= \sum_{i=1}^{L} \beta_i h(\boldsymbol{x}, \boldsymbol{w}_i, b_i) \\ &= \boldsymbol{h}(\boldsymbol{x}, W, \boldsymbol{b})^T \boldsymbol{\beta}, \end{aligned} \tag{1}$$

$$\boldsymbol{h}(\boldsymbol{x}, W, \boldsymbol{b}) = \left[h(\boldsymbol{x}, \boldsymbol{w}_1, b_1) \cdots h(\boldsymbol{x}, \boldsymbol{w}_L, b_L)\right]^T,$$
$$W = \{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_L\},$$
$$\boldsymbol{b} = \{b_1, \cdots, b_L\},$$
$$\boldsymbol{\beta} = \left[\beta_1 \cdots \beta_L\right]^T,$$

where h indicates an activation function and $\boldsymbol{w}_i$ and b_i indicate the weight vector and bias value from the input to the i-th hidden neuron ($1 \leq i \leq L$). The value of β_i represents the weight from the i-th hidden neuron to the output.

The most notable feature of ELM is that it randomly assigns hidden layer weights W and biases $\boldsymbol{b}$ and does not learn them, whereas the output weights $\boldsymbol{\beta}$ are the only parameters learned. For the training data of N input–output pairs $D = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \cdots, (\boldsymbol{x}_N, t_N)\}$, (1) can be expressed as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T},$$

where

$$\mathbf{H} = \left[\boldsymbol{h}(\boldsymbol{x}_1, W, \boldsymbol{b}) \cdots \boldsymbol{h}(\boldsymbol{x}_N, W, \boldsymbol{b})\right]^T$$
$$\mathbf{T} = \left[t_1 \cdots t_N\right]^T$$

For this purpose, the ELM output weights $\boldsymbol{\beta}$ are calculated by the following equation using pseudo-inverse matrix operations with the regularization term [40]:

$$\boldsymbol{\beta} = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T},$$

when the number of training samples was not large. Conversely, for the case where the number of training samples is large ($N \gg L$), the following alternative formula is used:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}. \tag{2}$$

The output weights of ELM can be quickly calculated using the pseudo-inverse matrix operation without any iterative procedures. The user parameters are only the activation function h, regularization coefficient C, and number of hidden neurons L, making parameter tuning easy. Furthermore, the activation function is non-differentiable, which provides flexibility. In ELM, the following activation functions are generally employed:

Sigmoid (SIG):

$$h(\boldsymbol{x}, \boldsymbol{w}, b) = \frac{1}{1 + \exp(-(\boldsymbol{w}\boldsymbol{x} + b))}$$

Gaussian (GAU):

$$h(\boldsymbol{x}, \boldsymbol{w}, b) = \exp(-b\|\boldsymbol{x} - \boldsymbol{w}\|^2)$$

Multiquadric (MQ):

$$h(\boldsymbol{x}, \boldsymbol{w}, b) = (\|\boldsymbol{x} - \boldsymbol{w}\|^2 + b^2)^{\frac{1}{2}}$$

The advantages of ELM against conventional machine learning models such as NNs and SVMs have been reported in some previous works [41, 42] as follows:

- The training cost is low because ELM does not require the iterative procedures of backpropagation.
- ELM tends to have better generalization performance because it not only minimizes the mean squared error but also looks for the much simpler model.
- Because ELM does not use gradient-based learning, non-differentiable activation functions can be used.

Since these features are helpful as a surrogate model for SAEAs, this study focuses on ELM.

DirectRanker

DR is a NN-based pairwise ranking method that implements a quasi-order $\succcurlyeq$ in future space F such that the ranking is unique. In particular, the quasi-order $\succcurlyeq$ satisfies the following three conditions for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in F$:

1. Reflexivity: $\boldsymbol{x} \succcurlyeq \boldsymbol{x}$
2. Antisymmetry: $\boldsymbol{x} \not\succcurlyeq \boldsymbol{y} \Rightarrow \boldsymbol{y} \succcurlyeq \boldsymbol{x}$
3. Transitivity: $(\boldsymbol{x} \succcurlyeq \boldsymbol{y} \wedge \boldsymbol{y} \succcurlyeq \boldsymbol{z}) \Rightarrow \boldsymbol{x} \succcurlyeq \boldsymbol{z}$

This quasi-order can be defined by the ranking function $r : F \times F \rightarrow \mathbb{R}$ as follows:

$$x \succcurlyeq y \Leftrightarrow r(\boldsymbol{x}, \boldsymbol{y}) \geq 0.$$

The ranking function $r(\boldsymbol{x}, \boldsymbol{y})$ in DR is constructed using two subnetworks nn and the output layer. Subnetworks nn have the same structure and shared weights. On the other hand, the output layer calculates the output from the difference in the outputs of the two subnetworks, the output weights $\boldsymbol{w}$, and the activation function τ as follows:

$$o = r(\boldsymbol{x}, \boldsymbol{y}) = \tau\left(\boldsymbol{w}^T(nn(\boldsymbol{x}) - nn(\boldsymbol{y}))\right), \tag{3}$$

where the activation function $\tau : \mathbb{R} \rightarrow \mathbb{R}$ satisfies the following conditions: $\tau(-x) = -\tau(x)$, $\mathrm{sgn}(\tau(x)) = \mathrm{sgn}(x)$; the original work [10] used the hyperbolic tangent (tanh). The function expressed in (3) satisfies the three conditions that

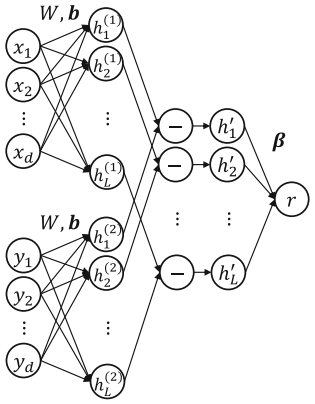


Fig. 2 A topology of ELDR

the quasi-order must satisfy: Please refer to the details in the original work [10].

Extreme learning machine-based DirectRanker (ELDR)

Assuming that DR is applied to SAEA as a ranking-based surrogate model, the problem of high computational cost for the training of DR may arise, because it optimizes the weights of NN through iterative procedures using backpropagation. To address this issue, this study proposes a novel pairwise ranking method, ELM-based DR (ELDR), which constructs the subnetwork of DR with ELM, enabling quick training by the pseudo-inverse matrix operation without the iterative procedures of backpropagation.

Figure 2 illustrates the topology of ELDR. ELDR takes two d-dimensional inputs, x and y, and calculates the output r from the subtraction of two hidden layers. ELDR employs the random-weighted single-layer network in (1) for the DR subnetwork and defines the ranking function r_{ELDR} as follows:

$$r_{ELDR}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{h}(\boldsymbol{x}, W, \boldsymbol{b}) - \boldsymbol{h}(\boldsymbol{y}, W, \boldsymbol{b}))^T \boldsymbol{\beta},$$

where W and $\boldsymbol{b}$ are the weight parameters randomly assigned to the same ELM and $\boldsymbol{\beta}$ is the only parameter to be trained. Because ELDR uses two subnetworks with the same structure, weights, and activation functions as DR, the ranking function r_{ELDR} also satisfies the three conditions of the quasi-order.

Algorithm 1 shows the training procedure for ELDR. First, all feature vectors $\boldsymbol{x}_i$ are normalized in the range $[-1, 1]$ using the minimum and maximum values in the dataset D. Then, the paired dataset D_{pair} is constructed with all combinations of the two solutions in dataset D as

$$D_{pair} = \{(\boldsymbol{x}_1^{(1)}, \boldsymbol{x}_1^{(2)}, t_1), \cdots, (\boldsymbol{x}_{N_p}^{(1)}, \boldsymbol{x}_{N_p}^{(2)}, t_{N_p})\}$$

Algorithm 1 ELDR Training Procedure

Input: Dataset $D = \{(\boldsymbol{x}_1, f_1), \cdots, (\boldsymbol{x}_N, f_N)\}(\boldsymbol{x} \in \mathbb{R}^d, f \in \mathbb{R})$, activation function h, regularization coefficient C, number of hidden neurons L
Output: Trained ELDR model

```
D' =Sort original dataset D according to the objective function value f_i
D_pair = ∅                                          ▷ Paired dataset generation
for i = {1, ··· N − 1} do
    for j = {i + 1, ··· N} do
        t = sgn(f_j − f_i)                          ▷ Minimization
        D_pair = D_pair ∪ {(x_i, x_j, t)}
    end for
end for
Randomly assign w_i and b_i (i = 1, ··· L)
Compute β by (5)
return W = {w_1, ··· , w_L}, b = {b_1, ··· , b_L}, β
```

$$t_j = \text{sgn}\left(f(\boldsymbol{x}_j^{(2)}) - f(\boldsymbol{x}_j^{(1)})\right)$$
$$j = 1, \cdots, N_p,$$

where sgn represents the sign function that returns $1, -1, 0$ according to the sign of the input value. Here, it is assumed that the minimization of f_i and t_j is $+1$ if $\boldsymbol{x}_j^{(1)}$ is better than $\boldsymbol{x}_j^{(2)}$, -1 if worse, and 0 if equal. For the constructed training dataset D_{pair}, ELDR is formulated as

$$(\mathbf{H^{(1)}} - \mathbf{H^{(2)}})\boldsymbol{\beta} = \mathbf{H'}\boldsymbol{\beta} = \mathbf{T} \tag{4}$$
$$\mathbf{H'} = \mathbf{H^{(1)}} - \mathbf{H^{(2)}}$$
$$\mathbf{T} = \left[t_1 \cdots t_{N_p}\right]^T,$$

where $\mathbf{H}^{(k)}$ is calculated from the k-th feature vector $\boldsymbol{x}_i^{(k)}$ $(k = 1, 2)$ in the i-th input pair as follows:

$$\mathbf{H}^{(k)} = \left[\boldsymbol{h}(\boldsymbol{x}_1^{(k)}, W, \boldsymbol{b}) \cdots \boldsymbol{h}(\boldsymbol{x}_{N_p}^{(k)}, W, \boldsymbol{b})\right]^T.$$

In this study, the weights W and bias $\boldsymbol{b}$ are randomly sampled from a uniform distribution in the ranges $[-1, 1]$ and $[0, 1]$, respectively [43]. From (4), the output weights $\boldsymbol{\beta}$ can be calculated as follows:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H'}^T\mathbf{H'}\right)^{-1}\mathbf{H'}^T\mathbf{T} \tag{5}$$

with a pseudo-inverse matrix calculation, as in Eq. (2).

When the prediction gives two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the output of ELDR is calculated as

$$\hat{t} = r_{ELDR}(\boldsymbol{x}_1, \boldsymbol{x}_2)$$
$$= \text{sgn}\left((\boldsymbol{h}(\boldsymbol{x}_1, W, \boldsymbol{b}) - \boldsymbol{h}(\boldsymbol{x}_2, W, \boldsymbol{b}))^T\boldsymbol{\beta}\right)$$

by using the weights W and biases $\boldsymbol{b}$ assigned in the training phase and the output weights $\boldsymbol{\beta}$ obtained by Eq. (5). If $\hat{t} =$

$+1$, ELDR predicts that $\boldsymbol{x}_1$ is better than $\boldsymbol{x}_2$; otherwise, it predicts that $\boldsymbol{x}_2$ is better.

Computational complexity

This subsection discusses the computational complexity of training ELDR. Let the size of the dataset D be N and the size of the paired dataset D_{pair} be $N_p = N(N-1)/2$. The number of hidden neurons is L. Thence, the size of the matrix H of the hidden layer is $N_p \times L$, and the size of the matrix T representing the training label is $N_p \times L$.

First, the time complexity of the hidden layer calculation is $O(NLd)$ for any activation function. Next, for the calculation of the output weights β, assuming that $N_p \gg L$ in general, and computing (2), the time complexity can be calculated as follows: The time complexity of $\mathbf{H}^T\mathbf{H}$ becomes $O(N_pL^2)$, and that of $\left(\mathbf{I}/C + \mathbf{H}^\mathbf{T}\mathbf{H}\right)^{-1}$ becomes $O(N_pL^2 + L^3)$, whereas, the time complexity of $\mathbf{H}^T$ is $O(N_pL)$. Therefore, the time complexity of computing $\boldsymbol{\beta} = \left(\mathbf{I}/C + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$ becomes $O(N_pL^2+L^3+N_pL+L^2)$. The most computationally expensive part is N_pL^2 and L^3, but because $N_p \gg L$, the time complexity of ELDR finally becomes $O(N_pL^2) = O(N^2L^2)$. Thus, the time complexity of ELDR increases with the product of the square of the dataset size N and the square of the number of hidden neurons L.

Example application of ELDR to SAEA: ELDR surrogate-assisted hybrid optimization

To investigate the applicability of ELDR to SAEAs, this study incorporated ELDR into the state-of-the-art SAEA method SAHO [11] to produce ELDR-SAHO. The original SAHO utilizes RBF as its surrogate model and finds a promising solution by simultaneously using Differential Evolution (DE) [33] and Teaching-learning-based Optimization (TLBO) [44, 45]. ELDR-SAHO estimates the dominance relationship between solutions using ELDR instead of predicting the fitness using RBF. This section briefly explains the search algorithms, DE and TLBO, used in SAHO and describes in detail the algorithm employed in ELDR-SAHO.

DE

SAHO uses the DE/1/best strategy because of its high convergence capability. The DE/1/best strategy generates a mutant vector $\boldsymbol{v}_i$ for a solution $\boldsymbol{x}_i$ as follows:

$$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F \cdot (\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}), \tag{6}$$

where $\boldsymbol{x}_{best}$ represents the best-fitted individual in the current population, $r1$ and $r2$ ($r1 \neq r2 \neq i$) denote random indices, and F is the scaling parameter. Using a mutant vector $\boldsymbol{v}_i$, a trial vector $\boldsymbol{u}_i$ for a solution $\boldsymbol{x}_i$ is calculated as

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } \texttt{rand}(0,1) \leq Cr \vee j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}, \tag{7}$$

where $\texttt{rand}(0, 1)$ produces a uniform random value in the range $[0, 1]$, Cr is the crossover rate, and j_{rand} denotes a random index in $[1, D]$. If the fitness value of the trial vector $\boldsymbol{u}_i$ is better than that of the current solution $\boldsymbol{x}_i$, then its position is updated.

TLBO

TLBO is a population-based search algorithm that simulates teacher instruction and mutual learning in education and consists of a teacher phase and a learner phase.

In the teacher phase, the best solution in the current population is selected as a teacher individual $\boldsymbol{x}_{teacher}$, and the positions of other solutions (students) are updated using the mean of the population $\boldsymbol{x}_{mean}$ as follows:

$$\boldsymbol{x}_{new,i} = \boldsymbol{x}_{old,i} + r_i\left(\boldsymbol{x}_{teacher} - T_F\boldsymbol{x}_{mean}\right), \tag{8}$$

where r_i is a uniform random value in $[0, 1]$, and T_F is a variable calculated as $T_F = 1 + \texttt{round}\left(\texttt{rand}(0, 1)\right)$ ($\texttt{round}$ rounds off the input value); that is, T_F takes 1 or 2 with equal probability. If the fitness value of an updated position $\boldsymbol{x}_{new,i}$ is better than that of the original position $\boldsymbol{x}_{old,i}$, then its position is updated.

By contrast, the learner phase updates each solution in the population by interacting with a random solution. For a solution $\boldsymbol{x}_i$, a random solution $\boldsymbol{x}_j$ ($i \neq j$) is selected, and the following equation produces a new position:

$$\boldsymbol{x}_{new,i} = \begin{cases} \boldsymbol{x}_{old,i} + r_i\left(\boldsymbol{x}_{old,i} - \boldsymbol{x}_{old,j}\right) \\ \quad \text{if } f(\boldsymbol{x}_{old,i}) < f(\boldsymbol{x}_{old,j}) \\ \boldsymbol{x}_{old,i} + r_i\left(\boldsymbol{x}_{old,j} - \boldsymbol{x}_{old,i}\right) \\ \quad \text{otherwise} \end{cases}, \tag{9}$$

where r_i denotes a uniform random value in $[0, 1]$. Similar to the teaching phase, the position of a solution $\boldsymbol{x}_{old,i}$ is updated if the fitness value of the updated position $\boldsymbol{x}_{new,i}$ is better. TLBO alternatively repeats the teacher and learner phases, until the termination condition is satisfied.

The ELDR-SAHO Algorithm

The procedure used by the ELDR-SAHO algorithm, in which the proposed ELDR is incorporated into SAHO, is

Algorithm 2 ELDR-assisted SAHO (based on Algorithm 1 in [11])

Input: Population size ps, maximum number of fitness evaluations $MaxFE$, search generations K
Output: Optimal solution

1: Generate initial ps samples using LHS, evaluate them using the real fitness function, and store them to $DB = \{(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \cdots, (\boldsymbol{x}_{ps}, f(\boldsymbol{x}_{ps}))\}$
2: $NFE = ps$, $RunDE = True$
3: **while** $NFE < MaxFE$ **do**
4: Initialize population with ps top-ranking best samples in D, $gen = 1$
5: **while** $gen \leq K$ **do**
6: Select n nearest neighbors for each solution in the population as the training dataset $D \subseteq DB$
7: Build ELDR surrogate model r_{ELDR} using Algorithm 1
8: **if** $RunDE$ **then** ▷ Run DE
9: Generate the trial population using (6) and (7)
10: Replace parent individual $\boldsymbol{x}_i$ with trial vector $\boldsymbol{u}_i$ if $r_{ELDR}(\boldsymbol{u}_i, \boldsymbol{x}_i) = +1$
11: $gen = gen + 1$
12: **else** ▷ Run TLBO
13: Perform teacher phase using (8) and replace individuals if $r_{ELDR}(\boldsymbol{x}_{new,i}, \boldsymbol{x}_{old,i}) = +1$
14: $gen = gen + 1$
15: Perform learner phase using (9) and replace individuals if $r_{ELDR}(\boldsymbol{x}_{new,i}, \boldsymbol{x}_{old,i}) = +1$
16: $gen = gen + 2$
17: **end if**
18: **end while**
19: Use Algorithm 3 to select the individual $\boldsymbol{x}_{eval}$ for the real fitness evaluation, $NFE = NFE + 1$
20: Update the database $DB = DB \cup \{(\boldsymbol{x}_{eval}, f(\boldsymbol{x}_{eval}))\}$
21: **if** $f(\boldsymbol{x}_{eval})$ is worse than the former best **then**
22: $RunDE = \neg RunDE$ ▷ Reverse $RunDE$
23: **end if**
24: **end while**
25: **return** Optimal solution

shown in Algorithm 2. ELDR-SAHO first produces ps well-distributed initial samples using Latin hypercube sampling (LHS) [46], and evaluates them using an actual (computationally expensive) fitness evaluation. All evaluated initial samples are stored in the dataset DB. For each iteration, the initial population is generated from the top ps solutions in the dataset DB, and K generations are evolved using DE or TLBO. The ELDR surrogate model is trained using a subset D extracted from DB consisting of n nearest neighbors in DB for each solution in the current population. Note that the design variable is normalized within the minimum and maximum variables in the extracted dataset D. A variable $RunDE$ determines the algorithm used for each search. DE is performed if $RunDE = True$, whereas TLBO is performed if $RunDE = False$;

After the K-generations search, a promising solution for the actual fitness evaluation is selected according to Algorithm 3. In Algorithm 3, the best solution in the population at K generations is compared with the mean of the top r subpopulation ($r \in [1, ps]$), and a solution that the ELDR surrogate predicts better is selected as a promising solution. The selected promising solution is actually evaluated using the computationally expensive fitness function. If the promising solution is better than the best solution thus far, the current search algorithm (DE or TLBO) is continuously used. Otherwise, the search algorithm is switched.

Algorithm 3 Selecting the individual for real fitness evaluation (based on Algorithm 3 in [11])

1: Sort individuals in the current population according to the ELDR surrogate model
2: Randomly generate an integer number r in the range $[1, ps]$
3: Compute the mean of the top subpopulation with the indexes $[1, r]$ as $\boldsymbol{x}_{mean} = \frac{1}{r}\sum^{r} \boldsymbol{x}_i$
4: Select the best individual in the current population as $\boldsymbol{x}_{best}$
5: **if** $r_{ELDR}(\boldsymbol{x}_{best}, \boldsymbol{x}_{mean}) = +1$ **then**
6: **return** $\boldsymbol{x}_{best}$
7: **else**
8: **return** $\boldsymbol{x}_{mean}$
9: **end if**

The essential procedure is the same as that of the original SAHO with the RBF surrogate model but differs in Steps 7, 10, 13, and 15 in Algorithm 2 and Steps 1 and 5 in Algorithm 3. In these steps, the ELDR surrogate model is used to predict the dominance relationship.

Preliminary experiment: accuracy of ELDR

This section analyzes the estimation accuracy of the proposed ELDR surrogate model. The first experiment explored the parameter sensitivity of ELDR regarding the activation function h, number of hidden neurons L, and regularization coefficient C. Then, "Comparison with other surrogate models" compared the estimation accuracy of ELDR with other surrogate models, RBF and RankSVM, which are generally used in previous SAEA research. Finally, the computational time of these surrogate models were compared in "Computation time".

Experimental sttings

This study used the eight single-objective continuous optimization benchmarks shown in Table 1. This choice can be justified because these functions have been widely used to investigate the performance of SAEAs in recent research and have different fitness landscape characteristics. The dimensions of the design variables were set as $d = 20$ and 100 to compare surrogate model performance for small and large problems. It is known that RBF has high estimation accuracy for problems up to 20–30 dimensions, while 100 dimensions are considered a large-scale problem in the SAEA domain. The training datasets consisting of $N_{train} = 2d, 5d$ (d is the dimension) randomly sampled data units were used for train-

Table 1 Benchmark problems

Problem	Domain	Global opt.	Characteristics
Ellipsoid	$[-5.12, 5.12]^d$	0	Unimodal and separable
Rosenbrock	$[-2.048, 2.048]^d$	0	Multimodal with narrow valley and non-separable
Ackley	$[-32.768, 32.768]^d$	0	Multimodal and separable
Griewank	$[-600, 600]^d$	0	Multimodal and non-separable
Rastrigin	$[-5.12, 5.12]^d$	0	Multimodal and separable
Shifted rotated rastrigin's function (CEC 2005 F_{10} [47])	$[-5, 5]^d$	−330	Multimodal, shifted, rotated, non-separable
Rotated hybrid composition function (CEC 2005 F_{16} [47])	$[-5, 5]^d$	120	Multimodal, rotated, non-separable
Rotated hybrid composition function with a narrow basin for the global optimum (CEC 2005 F_{19}) [47])	$[-5, 5]^d$	10	Multimodal, non-separable, a narrow basin for the global optimum

ing ELDR, RBF, and RandSVM. The trained models were tested on $10d$ random test samples generated independently from the training dataset. The training data size was chosen to verify estimation accuracy in SAEA when data are limited in the early stages of the search and when a certain number of data samples are obtained in later stages.

The prediction accuracy was assessed using Kendall's rank correlation coefficient for 20 independent pairs of training and test data, and the average rank correlation was compared. Kendall's rank correlation coefficient (hereafter, Kendall's τ) returns $\tau = +1$ if the predicted and actual ranks are entirely the same, whereas it returns $\tau = -1$ if they are entirely different. A higher rank correlation indicates a better rank prediction. For the ranking method (ELDR and RankSVM), the test data are sorted by the prediction, and their rank is compared with the actual data. For the RBF model, the test data are sorted based on the predicted fitness values, and their rank is also compared with the actual rank (not the comparison of the actual fitness values). The experiments were executed on a computer with an Intel Xeon W-2295 3.00 GHz CPU with 64 GB RAM using MATLAB R2019a.

Comparison of hyperparameters

This subsection reports the experiment conducted to evaluate the different hyperparameters of ELDR: activation function h, number of hidden neurons L, and regularization coefficient C. For the activation function h, the experiment used three functions: Sigmoid (SIG), Gaussian (GAU), and Multiquadric (MQ), as shown in "Extreme Learning Machine". The number of hidden neurons was set to $L = \{d+1, 2d+1, 3d+1, \cdots, 10d+1\}$ according to the dimension d of the design variable. The regularization coefficient $C = \{2^{-5}, 2^{-4}, \cdots, 2^5\}$.

Because the difficulty of estimation varies for each problem, the scale of the estimation accuracy (i.e., Kendall's τ) varies. To consider these differences and investigate the overall performance for all problems, this study uses the same index as in the literature [7, 48]. Specifically, this study investigates the influence of the regularization coefficient C and the number of hidden neurons L for each activation function. For ELDRs using each activation function, the problem instance is denoted as $F = \{f_k \mid k = 1, 2, \cdots, n\}$, the parameter setting (combination of C and L) is denoted as $S = \{s_i \mid i = 1, 2, \cdots, m\}$, and ELDR with the parameter setting s_i is denoted as A_{s_i}. The following equation calculates the performance $PM(s_i)$ for each parameter setting s_i:

$$PM(s_i) = \frac{1}{m-1} \sum_{j=1, j\neq i}^{m} P\left(A_{s_i} \succ A_{s_j}\right), \tag{10}$$

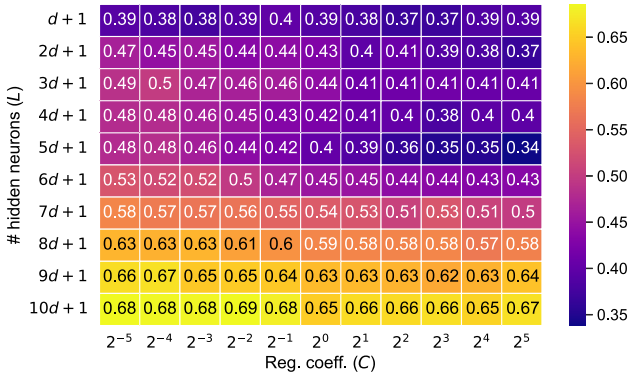
where $P(A_{s_i} \succ A_{s_j})$ represents the ratio that A_{s_i} outperforms A_{s_j} and is calculated as follows:

$$P\left(A_{s_i} \succ A_{s_j}\right) = \frac{1}{n} \sum_{k=1}^{n} P\left(q_{i,k} > q_{j,k} \mid f_k\right),$$

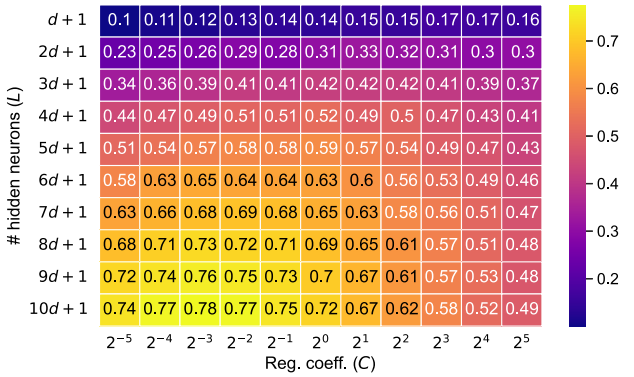
where $P\left(q_{i,k} > q_{j,k} \mid f_k\right)$ represents the ratio that the estimation accuracy of A_{s_i} is better than that of A_{s_j} for a problem instance f_k. $P\left(q_{i,k} > q_{j,k} \mid f_k\right)$ is calculated using the following equation:

$$P\left(q_{i,k} > q_{j,k} \mid f_k\right) = \frac{1}{N_i \times N_j} \sum_{t_i=1}^{N_i} \sum_{t_j=1}^{N_j} \mathbb{1}\left(\tau_{i,k,t_i} > \tau_{j,k,t_j}\right),$$

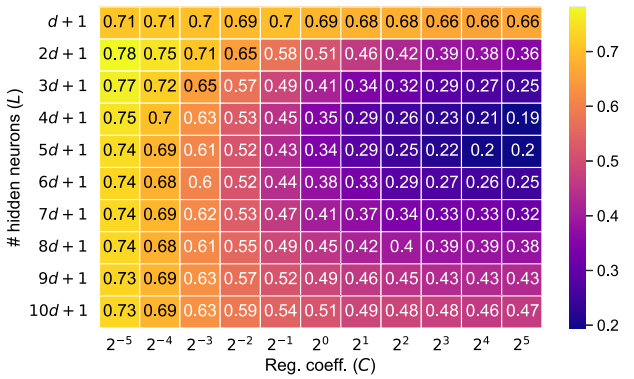
where $\tau_{i,k,t}$ represents Kendall's τ for the t-th dataset of a problem instance f_k obtained by A_{s_i}. N_i and N_j represent



(a) Sigmoid



(b) Gaussian



(c) Multiquadric

Fig. 3 Performance metric $PM(s_i)$ calculated using Eq. (10) for each activation function

the number of test data in a problem instance f_k for A_{s_i} and A_{s_j}, respectively (in this experiment $N_i = N_j = 20$). The function $\mathbb{1}(x)$ is the indicator function, which returns 1 if x is true and 0 otherwise.

Figure 3 shows the performance metric $PM(s_i)$ calculated using Eq. (10) for each activation function. The horizontal axis shows the regularization coefficient and the vertical axis shows the number of hidden neurons. The value in each cell indicates the $PM(s_i)$ value for the corresponding parameter setting.

The result in Fig. 3a indicates that when using the Sigmoid function, the number of hidden neurons significantly affects its estimation accuracy, and a larger number of hidden neurons results in better performance. Conversely, the effect of the regularization coefficient is small. The results show that the combination of $L = 10d + 1$ and $C = 2^{-2}$ achieves the highest estimation accuracy when using the Sigmoid function. However, the difference in the performance metric is small among the different parameter settings. This characteristic indicates that the optimal parameter setting varies depending on the characteristic of target problems.

Next, Fig. 3b shows that when using the Gaussian function, both the number of hidden neurons and the regularization coefficient significantly affect the estimation accuracy of ELDR. Specifically, a larger number of hidden neurons tends to result in high estimation accuracy, whereas the maximum performance is obtained around $C = 2^{-3}$ to 2^0 especially when the number of neurons is large for the regularization coefficient. Consequently, the combination of $L = 10d + 1$ and $C = 2^{-3}$ achieves the highest estimation accuracy when using the Gaussian function. The performance metric among the different parameter settings is large when using the Gaussian function. This indicates that the optimal parameter setting is largely independent of the problem, and the best parameter setting of $L = 10d + 1$ and $C = 2^{-3}$ can obtain stable performance when using the Gaussian activation function.

Finally, focusing on the Multiquadric function, Fig. 3c indicates that small normalization coefficients tend to give stable performance regardless of the number of hidden neurons. In particular, the normalization coefficient of $C = 2^{-5}$ provides the highest estimation accuracy with $L \geq 2d + 1$. Although the performance difference between different L is slight, the combination of $L = 2d + 1$ and $C = 2^{-5}$ achieves the highest estimation accuracy when using the Multiquadric function. The large difference in performance metrics between the parameter settings can be seen. In particular, the choice of $C = 2^{-5}$ can be optimal. This indicates that the best parameter setting of $L = 2d + 1$ and $C = 2^{-5}$ can be recommended regardless of problems for the Multiquadric function.

Comparison with other surrogate models

This subsection compares ELDR with RBF and RankSVM. ELDR employed the hyperparameters chosen from the results in the previous subsection. In particular, this comparison used ELDR with:

- Sigmoid activation with $L = 10d + 1$, $C = 2^{-2}$
- Gaussian activation with $L = 10d + 1$, $C = 2^{-3}$

Table 2 Comparison of Kendall's τ of ELDR, RBF, and RankSVM

Problem	d	N_{train}	ELDR			RBF	RankSVM
			SIG	GAU	MQ		
Ellipsoid	20	40	0.07	**0.60**	0.56	0.16	0.23
		100	0.19	0.63	**0.63**	0.55	0.48
	100	200	0.01	**0.61**	0.53	0.13	0.15
		500	0.03	**0.63**	0.61	0.51	0.34
Rosenbrock	20	40	0.16	**0.60**	0.58	0.25	0.29
		100	0.28	0.63	**0.64**	0.55	0.48
	100	200	0.14	0.57	**0.57**	0.26	0.26
		500	0.15	0.61	**0.65**	0.56	0.44
Ackley	20	40	0.03	**0.44**	0.39	0.10	0.15
		100	0.13	0.46	**0.46**	0.36	0.32
	100	200	0.01	**0.48**	0.37	0.09	0.11
		500	0.03	**0.50**	0.46	0.36	0.25
Griewank	20	40	0.08	0.84	**0.88**	0.22	0.34
		100	0.21	0.90	**0.92**	0.73	0.65
	100	200	0.01	0.84	**0.87**	0.19	0.23
		500	0.06	0.88	**0.92**	0.75	0.55
Rastrigin	20	40	0.03	**0.43**	0.38	0.10	0.15
		100	0.11	**0.44**	0.44	0.29	0.24
	100	200	0.00	**0.44**	0.33	0.07	0.09
		500	0.02	**0.47**	0.43	0.33	0.20
CEC 2005 F_{10}	20	40	0.51	0.60	**0.68**	0.60	0.60
		100	0.59	0.73	**0.77**	0.76	0.71
	100	200	0.54	0.44	**0.72**	0.64	0.58
		500	0.55	0.54	**0.79**	0.78	0.74
CEC 2005 F_{16}	20	40	0.38	0.39	0.40	0.41	**0.41**
		100	0.43	0.51	0.51	**0.56**	0.52
	100	200	0.57	0.35	**0.64**	0.62	0.58
		500	0.56	0.48	0.71	**0.72**	0.67
CEC 2005 F_{19}	20	40	0.09	**0.18**	0.15	0.13	0.15
		100	0.16	0.26	0.24	**0.38**	0.30
	100	200	0.19	0.24	**0.25**	0.22	0.20
		500	0.17	0.29	0.33	**0.36**	0.25
Average rank			4.88	2.25	**1.69**	2.88	3.31

The best and second-best results are highlighted in **boldface** and underlined, respectively

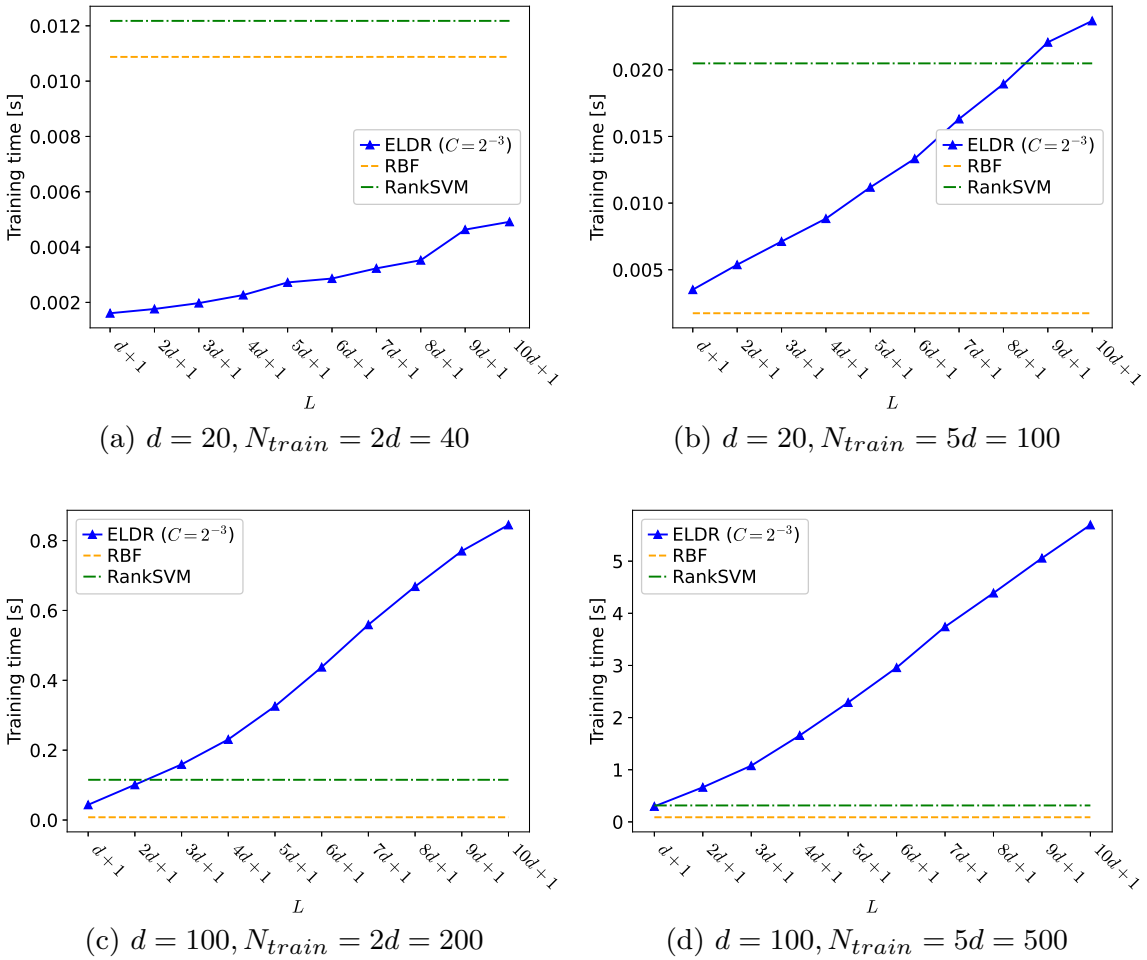
- Multiquadric activation with $L = 2d + 1$, $C = 2^{-5}$

RBF uses the cubic basis function ($\phi(x) = x^3$), which has been mostly used in previous works that employed the RBF surrogate. The parameters of RankSVM were chosen from the literature [7].

Table 2 shows the estimation accuracies for ELDR, RBF, and RankSVM. The bottom row summarizes the average rank of all problem instances and all dimensions.

First, it can be seen that ELDR using the Sigmoid function has a significantly lower estimation accuracy than the other methods in the first five benchmarks and is less accurate. However, its accuracy improves on the CEC 2005 benchmarks and is higher than that of the Gaussian activation function for some problems, e.g., 100-dimensional cases of F_{10} and F_{16}. The average rank is the lowest among the five surrogate models. This indicates that the Sigmoid function has low estimation accuracy and low usefulness even when the parameters are optimally configured, but it can be an option for complex composition functions.

Second, ELDRs with the Gaussian and Multiquadric functions outperform RBF and RankSVM used in conventional SAEAs and achieve the best or second-best performance on most problems. In particular, ELDR with the Multiquadric



(a) $d = 20, N_{train} = 2d = 40$ (b) $d = 20, N_{train} = 5d = 100$

(c) $d = 100, N_{train} = 2d = 200$ (d) $d = 100, N_{train} = 5d = 500$

Fig. 4 Computation time of three surrogate models

function has a consistently high estimation accuracy and the highest average rank among the five surrogate models. Additionally, ELDRs with the Gaussian and Multiquadric functions exhibit a more stable estimation accuracy than RBF and RankSVM, even when the training dataset is small ($N_{train} = 2d$). This feature benefits SAEAs because the amount of training data available for training surrogate models is generally limited.

Computation time

To evaluate the computation time of the proposed ELDR, the training times of the three methods are compared. Note that because the influence of differences in the regularization coefficients and activation functions of ELDR on the computation time is extremely small, only results using Gaussian as the activation function and $C = 2^{-3}$ as the regularization coefficient are compared in this paper. It has been confirmed that similar results can be obtained for other activation functions and regularization coefficients. In addition, because the training time does not depend on the characteristics of the problem, this paper addresses the average of all problems.

Figure 4 shows the average training time for ELDR, RBF, and RankSVM. The blue, orange, and green lines represent the results for ELDR, RBF, and RankSVM, respectively. In each figure, the horizontal axis represents the number of hidden neurons in ELDR, while the vertical axis represents the training time [s].

These figures show that the training time for ELDR increases with increasing the number of hidden neurons L in a nearly linear trend. This trend is smaller than the increase of L squared, derived from $O(N^2L^2)$ for the computational complexity of ELDR discussed in "Computational complexity". This is because this paper uses MATLAB to implement ELDR, and the computational complexity required to calculate H^TH, which is the most computationally expensive, is less than that required for a simple matrix product. This indicates that the computational complexity of ELDR is smaller than the theoretical one, and the computation time linearly increases with increasing L in practice.

Next, the computation times of RBF and RankSVM is compared. First, ELDR has the shortest training time when the number of dimensions and training data are small ($d = 20$, $N_{train} = 2d = 40$). When the number of dimensions and training data is large, the computation time of ELDR for small L ($L = d + 1$) is similar to or slightly longer than that of RBF and RankSVM. On the other hand, as the number of hidden neurons L increases, the training time of ELDR becomes longer than that of other surrogate models. In particular, in the case of $d = 100$, $N_{train} = 5d = 500$, which has the highest number of dimensions and training data, the computation time of ELDR for $L = 10d + 1$ is approximately 60 times longer.

These results indicate that ELDR can be trained with the same training time as existing surrogate models such as RBF and RankSVM for low-dimension, small-sample training. On the other hand, for high-dimension, many-sample training, ELDR requires significantly longer training time than existing models, especially when the number of hidden neurons L is increased. However, the training time for ELDR is acceptable compared to the expensive solution evaluations, which sometimes require several hours or days, targeted by SAEAs.

Numerical experiments

Numerical experiments were conducted using well-known benchmark problems to investigate the effectiveness of the proposed method. The experiments used the eight single-objective continuous optimization benchmarks listed in Table 1. The dimensions of the problems were set to $d = \{10, 20, 30, 50, 100\}$.

Experimental settings

In the experiments, ELDR-SAHO—explained in "Example application of ELDR to SAEA: ELDR surrogate-assisted aybrid optimization"—was used as the ELDR-assisted EA. It was compared with the conventional state-of-the-art SAEA methods GORS-SSLPSO [13], FSAPSO [15], SLPSO [49], CA-LLSO [26], and SAHO [11]. GORS-SSLPSO, FSAPSO, SLPSO, and SAHO use the regression model based on the RBF, while CA-LLSO uses the classification model based on gradient boosting classifier (GBC) [50]. The implementations of GORS-SSLPSO,[1] FSAPSO,[2] SLPSO,[3] and CA-LLSO [4] available on the Internet were used. The implementation of SAHO was provided by Pan et al. [11], and ELDR-SAHO was implemented based on it.

Algorithm 4 Model selection procedure in ELDR-SAHO

Input: Dataset $D = \{(\boldsymbol{x}_1, f_1), \cdots, (\boldsymbol{x}_N, f_N)\}$
Output: Trained ELDR model
1: Randomly select 80% of D as D_{train}, and 20% as D_{valid}
2: Train $ELDR_{GAU}$ with the Gaussian function, $L = 10d + 1$, $C = 2^{-3}$ by using D_{train}
3: Train $ELDR_{MQ}$ with the Multiquadric function, $L = 2d + 1$, $C = 2^{-5}$ by using D_{train}
4: Calculate the Kendall's τ_{GAU} of $ELDR_{GAU}$ and τ_{MQ} of $ELDR_{MQ}$ using D_{valid}
5: **if** $\tau_{GAU} > \tau_{MQ}$ **then**
6: **return** $ELDR_{GAU}$
7: **else**
8: **return** $ELDR_{GAU}$
9: **end if**

The experimental setup was established based on Pan et al. [11]. The population size was set to $ps = 5 \times d$ when $d = \{10, 20, 30\}$ and to $ps = 100 + \lfloor d/10 \rfloor$ when $d = \{50, 100\}$. The neighbor size n when training ELDR (Step 6 in Algorithm 2) was set to $n = d$. The parameter values for DE were $Cr = 0.9$ and $F = 0.5$. The number of generations K required for DE or TLBO on the surrogate model was set to $K = 30$. The maximum fitness evaluation was $MaxFE = 11 \times d$ when $d = \{10, 20, 30\}$ and $MaxFE = 1000$ when $d = \{50, 100\}$. For the other algorithms, the parameter values recommended in their respective studies were used.

From the preliminary experimental results in "Preliminary experiment: accuracy of ELDR", it was revealed that ELDR using either the Gaussian or Multiquadric function has high potential as a surrogate model. However, the optimal activation function varies depending on the problem. Based on this analysis, the model selection through validation is introduced into ELDR-SAHO. Algorithm 4 shows the model selection procedure of ELDR-SAHO. The model selection compares the Gaussian function with $L = 10d + 1$, $C = 2^{-3}$, and the Multiquadric function with $L = 2d + 1$, $C = 2^{-5}$ as candidates. Subsequently, the model with the highest Kendall's τ on the validation set between these two models is used as a surrogate in the search. The validation set for model selection comprises randomly selected 20% of the data from the training dataset. This procedure is used at line 7 in Algorithm 2.

Results

Table 3 shows the mean and standard deviation of the best fitness after the maximum fitness evaluations in 20 independent runs. For each benchmark, the best (smallest) value is highlighted in boldface, whereas the second-best is underlined.

[1] https://github.com/yuhaibo2017/GORS-SSLPSO_code (accessed April 12, 2022).

[2] https://github.com/fanli525/-FSAPSO (accessed April 12, 2022).

[3] https://github.com/yuhaibo2017/SHPSO_code (accessed January 10, 2023).

[4] https://github.com/CarrieWei/CA-LLSO_Code (accessed January 10, 2023).

Table 3 Experimental results of the performance comparison

Problem	d	GORS-SSLPSO		FSAPSO		SHPSO		CA-LLSO		SAHO		ELDR-SAHO	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
Ellipsoid	10	4.83E-02 (≈)	6.28E-02	1.01E-01 (+)	1.63E-01	1.17E+00 (+)	7.19E-01	3.82E+01 (+)	1.31E+01	2.34E-02 (≈)	3.70E-02	**2.11E-02**	2.18E-02
	20	2.42E-01 (+)	1.75E-01	4.98E-01 (+)	4.59E-01	1.48E+01 (+)	7.03E+00	6.52E+01 (+)	1.76E+01	6.95E-02 (+)	8.63E-02	**2.01E-02**	1.66E-02
	30	5.18E-01 (+)	4.24E-01	1.01E+00 (+)	8.70E-01	3.39E+01 (+)	9.70E+00	1.07E+02 (+)	2.83E+01	1.60E-01 (+)	1.11E-01	**4.77E-02**	2.79E-02
	50	6.41E+01 (+)	2.21E+02	2.39E-01 (+)	2.45E-01	5.54E+00 (+)	1.68E+00	9.68E+01 (+)	1.90E+01	1.93E-04 (+)	3.24E-04	**7.47E-06**	3.03E-05
	100	1.26E+02 (+)	3.11E+02	8.39E+00 (+)	3.34E+00	1.15E+02 (+)	3.56E+01	1.50E+03 (+)	1.51E+02	9.02E-02 (+)	5.38E-02	**2.23E-02**	3.80E-02
Rosenbrock	10	2.31E+01 (+)	1.60E+01	1.34E+01 (+)	8.05E+00	1.15E+02 (+)	8.48E+01	1.41E+02 (+)	6.31E+01	1.09E+01 (≈)	4.06E+00	**8.98E+00**	1.59E-01
	20	5.83E+01 (+)	1.66E+01	2.08E+01 (+)	1.87E+00	1.62E+02 (+)	5.62E+01	1.37E+02 (+)	2.24E+01	2.80E+01 (+)	1.48E+01	**1.89E+01**	1.05E-01
	30	8.92E+01 (+)	1.58E+01	3.99E+01 (+)	9.13E+00	1.92E+02 (+)	5.80E+01	1.71E+02 (+)	2.18E+01	3.74E+01 (+)	1.30E+01	**2.88E+01**	7.85E-02
	50	1.71E+02 (+)	2.06E+02	4.91E+01 (+)	1.45E+00	5.11E+01 (+)	1.70E+00	1.28E+02 (+)	1.78E+01	4.77E+01 (≈)	1.19E-01	**4.76E+01**	4.09E-01
	100	2.28E+02 (+)	4.05E+01	1.12E+02 (+)	1.38E+01	2.18E+02 (+)	6.40E+01	6.17E+02 (+)	7.22E+01	**9.80E+01** (−)	5.00E-02	9.82E+01	2.22E-01
Ackley	10	3.39E+00 (+)	1.11E+00	3.76E+00 (+)	1.63E+00	1.37E+01 (+)	3.60E+00	1.53E+01 (+)	1.27E+00	5.02E+00 (+)	4.27E+00	**9.97E-01**	8.08E-01
	20	5.31E+00 (+)	3.60E+00	5.20E+00 (+)	2.48E+00	1.08E+01 (+)	1.46E+00	1.17E+01 (+)	6.69E-01	2.43E+00 (+)	1.65E+00	**4.70E-01**	2.41E-01
	30	5.45E+00 (+)	2.15E+00	5.03E+00 (+)	1.06E+00	9.88E+00 (+)	6.71E-01	1.11E+01 (+)	6.82E-01	1.91E+00 (+)	8.36E-01	**4.15E-01**	1.83E-01
	50	7.58E+00 (+)	3.41E+00	4.50E+00 (+)	1.44E+00	2.57E+00 (+)	2.58E-01	7.93E+00 (+)	4.35E-01	4.11E-05 (≈)	1.36E-05	**3.91E-05**	6.40E-05
	100	1.35E+01 (+)	2.10E+00	6.06E+00 (+)	1.64E+00	5.68E+00 (+)	7.50E-01	1.20E+01 (+)	4.85E-01	1.62E-02 (+)	4.56E-03	**7.64E-03**	4.50E-03
Griewank	10	7.44E-01 (−)	1.91E-01	8.83E-01 (≈)	1.32E-01	1.21E+00 (+)	2.30E-01	2.41E+01 (+)	6.85E+00	**7.39E-01** (−)	1.78E-01	9.44E-01	7.06E-02
	20	4.09E-01 (−)	1.50E-01	5.23E-01 (−)	1.58E-01	1.15E+00 (+)	1.79E-01	2.16E+01 (+)	4.49E+00	**3.14E-01** (−)	1.72E-01	8.15E-01	2.14E-01
	30	3.23E-01 (−)	9.85E-02	3.26E-01 (−)	1.12E-01	1.15E+00 (+)	1.01E-01	2.58E+01 (+)	5.87E+00	**1.14E-01** (−)	1.11E-01	8.73E-01	1.55E-01
	50	4.59E-03 (+)	7.05E-03	5.23E-03 (+)	6.54E-03	9.45E-01 (+)	6.96E-02	1.57E+01 (+)	2.75E+00	1.03E-02 (+)	2.71E-02	**5.24E-04**	1.56E-03
	100	4.90E+00 (+)	2.08E+01	1.73E-01 (+)	3.54E-02	1.13E+00 (+)	4.69E-02	1.12E+02 (+)	1.16E+01	2.96E-01 (+)	1.31E-01	**2.16E-02**	1.80E-02
Rastrigin	10	3.13E+01 (≈)	1.54E+01	3.21E+01 (≈)	9.73E+00	9.68E+01 (+)	1.50E+01	8.01E+01 (+)	1.26E+01	3.27E+01 (≈)	1.51E+01	**2.90E+01**	7.67E+00
	20	4.63E+01 (−)	2.02E+01	4.69E+01 (−)	1.48E+01	1.97E+02 (+)	2.44E+01	1.53E+02 (+)	1.46E+01	**4.57E+01** (−)	2.31E+01	6.99E+01	2.53E+01
	30	7.04E+01 (≈)	2.82E+01	6.82E+01 (−)	1.92E+01	2.75E+02 (+)	3.53E+01	2.26E+02 (+)	1.63E+01	**3.65E+01** (−)	1.66E+01	9.44E+01	4.24E+01
	50	1.71E+02 (+)	5.61E+01	1.08E+02 (≈)	2.53E+01	3.31E+02 (+)	6.32E+01	3.70E+02 (+)	2.07E+01	1.01E+02 (≈)	3.54E+01	**9.09E+01**	4.47E+01
	100	3.42E+02 (+)	4.74E+01	1.96E+02 (+)	2.70E+01	8.92E+02 (+)	1.01E+02	8.65E+02 (+)	2.57E+01	3.36E+02 (+)	9.87E+01	**4.14E+01**	9.35E+01
CEC 2005 F_{10}	10	**– 2.96E+02** (−)	1.77E+01	– 2.79E+02 (≈)	2.01E+01	-2.24E+02 (+)	1.77E+01	– 2.02E+02 (+)	1.89E+01	– 2.89E+02 (≈)	2.22E+01	-2.79E+02	1.75E+01
	20	– 2.44E+02 (−)	2.80E+01	– 1.97E+02 (≈)	5.21E+01	– 1.28E+02 (+)	3.00E+01	– 7.35E+01 (+)	1.87E+01	**– 2.55E+02** (−)	2.36E+01	-2.01E+02	3.24E+01
	30	– 1.93E+02 (−)	3.05E+01	– 1.67E+02 (−)	4.44E+01	– 3.49E+01 (+)	2.95E+01	2.80E+01 (+)	3.26E+01	**– 2.23E+02** (−)	2.85E+01	– 1.06E+02	4.60E+01
	50	– 1.07E+01 (≈)	5.94E+01	– 4.76E+01 (−)	7.58E+01	1.14E+02 (+)	1.64E+01	2.80E+02 (+)	4.64E+01	**– 1.46E+02** (−)	3.95E+01	7.52E+00	3.04E+01
	100	4.05E+02 (−)	8.04E+01	**2.39E+02** (−)	7.16E+01	7.22E+02 (+)	4.33E+01	1.22E+03 (+)	8.52E+01	3.63E+02 (−)	1.36E+02	6.34E+02	1.02E+02

Table 3 continued

Problem	d	GORS-SSLPSO		FSAPSO		SHPSO		CA-LLSO		SAHO		ELDR-SAHO	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
CEC 2005 F_{16}	10	3.90E+02 (≈)	6.28E+01	4.34E+02 (≈)	1.16E+02	5.68E+02 (+)	7.65E+01	1.34E+03 (+)	6.47E+01	**3.78E+02** (−)	4.97E+01	4.71E+02	1.13E+02
	20	**4.69E+02** (≈)	8.89E+01	4.77E+02 (≈)	1.15E+02	5.65E+02 (≈)	5.56E+01	1.39E+03 (+)	7.70E+01	4.83E+02 (≈)	8.88E+01	5.02E+02	1.01E+02
	30	5.79E+02 (≈)	1.32E+02	5.61E+02 (≈)	1.76E+02	5.63E+02 (≈)	8.04E+01	1.44E+03 (+)	4.93E+01	**4.90E+02** (≈)	8.38E+01	5.56E+02	1.11E+02
	50	4.92E+02 (≈)	5.51E+01	4.56E+02 (≈)	7.87E+01	4.86E+02 (≈)	4.32E+01	1.44E+03 (+)	2.84E+01	**3.24E+02** (−)	9.70E+01	5.54E+02	1.75E+02
	100	5.33E+02 (≈)	3.29E+01	**3.85E+02** (−)	6.31E+01	5.08E+02 (≈)	1.25E+01	1.53E+03 (+)	2.30E+01	3.94E+02 (−)	6.10E+01	4.97E+02	5.36E+01
CEC 2005 F_{19}	10	1.18E+03 (≈)	8.47E+01	1.12E+03 (−)	8.35E+01	1.32E+03 (+)	9.67E+01	**8.19E+02** (−)	9.72E+01	1.17E+03 (≈)	7.89E+01	1.18E+03	7.17E+01
	20	1.11E+03 (≈)	1.02E+02	1.16E+03 (≈)	1.14E+02	1.13E+03 (−)	2.73E+01	**8.86E+02** (−)	1.88E+02	1.09E+03 (≈)	1.16E+02	1.16E+03	1.22E+02
	30	9.94E+02 (−)	3.28E+01	9.89E+02 (−)	4.97E+01	1.04E+03 (≈)	3.12E+01	**9.04E+02** (−)	1.80E+02	9.88E+02 (−)	6.40E+01	1.04E+03	5.62E+01
	50	1.09E+03 (+)	7.81E+01	1.11E+03 (+)	5.58E+01	1.00E+03 (+)	2.23E+01	9.92E+02 (+)	5.47E+01	9.65E+02 (+)	1.90E+01	**9.26E+02**	6.57E+01
	100	1.28E+03 (+)	4.13E+01	1.30E+03 (+)	5.17E+01	1.23E+03 (+)	2.21E+01	1.03E+03 (+)	1.67E+01	1.13E+03 (+)	5.19E+01	**9.10E+02**	3.08E-02
+/≈/−	$d \leq 30$	8/8/8		9/8/7		20/3/1		21/0/3		7/8/9		NA	
	$d \geq 50$	12/3/1		11/2/3		14/2/0		16/0/0		8/3/5		NA	
	Total	20/11/9		20/10/10		34/5/1		37/0/3		15/11/14		NA	

The best and second-best results are highlighted in **boldface** and underlined, respectively

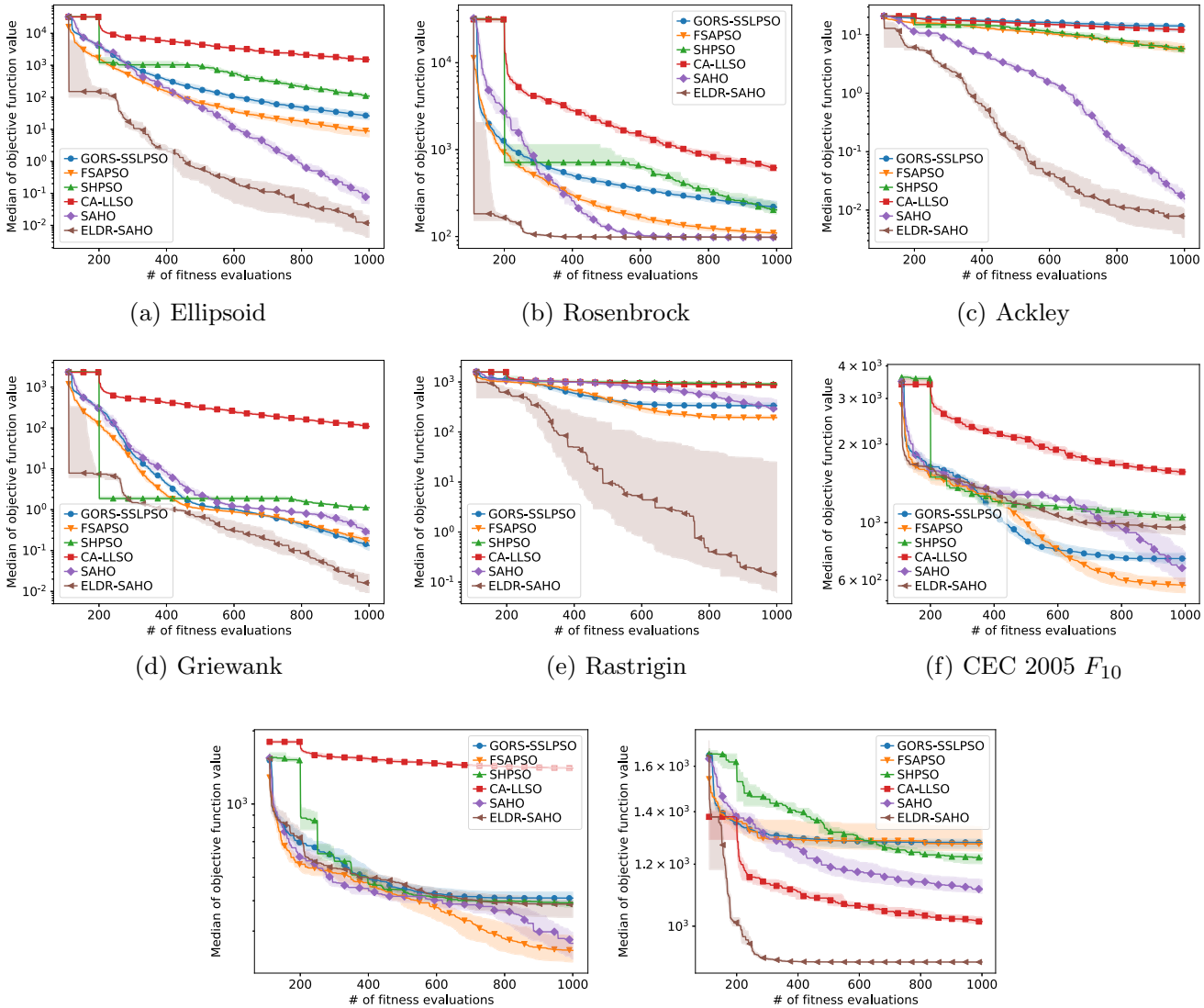
The Mann–Whitney U test with a significance level of 5% was performed, and the results were adjusted using the Bonferroni correction [51] to confirm the statistical difference between the proposed method and the other methods. Where the proposed method obtained significantly better results than the other methods, "+" marks are put, "−" marks denote significantly worse, and "≈" denotes no significant difference is found. Finally, the results are summarized at the bottom of the table.

The experimental results show that ELDR-SAHO achieved the best mean fitness in 21 out of 40 cases and was the second-best in two cases. Moreover, the proposed method is significantly better than the other algorithms on many benchmarks. However, on the F_{10} and F_{16} problems, the proposed method was inferior to SAHO, which uses RBF as a surrogate. Moreover, on the low-dimensional problem ($d \leq 30$) of the Griewank, Rastrigin, and F_{19}, ELDR-SAHO does not achieve good results compared with the other methods. On the other hand, ELDR-SAHO significantly outperformed the other algorithms for the high-dimensional benchmarks except for the Rosenbrock ($d = 100$), F_{10}, and F_{19} benchmarks.

Figure 5 shows the median objective function values transitions and the interquartile ranges [range between the third quartile (Q3) and the first quartile (Q1)] when $d = 100$. The horizontal axis indicates the number of actual fitness evaluations, whereas the vertical axis shows the objective function value on a logarithmic scale. Note that for the CEC 2005 benchmarks, the difference between the optimum and obtained value is plotted.

First, focusing on the F_{10} and F_{16} problems, where ELDR-SAHO was significantly inferior to the other methods on the 100-dimensional problem, it can be seen that the results are comparable to the other methods until the middle stage of the search (about 400 fitness evaluations). On the other hand, in the subsequent search, the proposed method stagnates, whereas the other methods decrease their objective function values. This suggests that the proposed method is stuck in a local solution from which it cannot escape on these benchmarks.

By contrast, the figures show ELDR-SAHO as having a rapid improvement in the objective function value from the early stage of optimization, and it continues the search without stagnation on the Ellipsoid, Rosenbrock, Ackley, Griewank, Rastrigin, and F_{19} problems. This can be attributed to the ability of ELDR to estimate the dominance relationship more accurately with a smaller amount of training data than RBF and GBC. In particular, although the final objective function values of ELDR-SAHO were worse than those of SAHO on the 100-dimensional Rosenbrock function (Fig. 5b), ELDR-SAHO converges to the final result in fewer evaluations than the other algorithms.



(a) Ellipsoid (b) Rosenbrock (c) Ackley

(d) Griewank (e) Rastrigin (f) CEC 2005 F_{10}

(g) CEC2005 F_{16} (h) CEC 2005 F_{19}

Fig. 5 Transitions of median objective function values and the interquartile ranges [ranges between the third quartile (Q3) and the first quartile (Q1)] when number of dimensions $d = 100$ (the vertical axis is logarithmic scale)

These results indicate that ELDR-SAHO has better search performance than conventional SAEA using regression and classification surrogate models for many problems (especially high-dimensional problems). However, the results suggest that ELDR-SAHO may fall into the local optimum on some problems. Note that although ELDR was applied to SAHO in this study, the application of ELDR is not necessarily suitable for SAHO because SAHO is designed as an SAEA using RBF. In particular, for the F_{10} and F_{16} problems, it may be possible to enhance the performance from ELDR by proposing an SAEA with a mechanism to prevent (or escape from) falling into local optimum. In addition, because ELDR can be searched only by the superiority of the solutions in all problems, it may be applicable for interactive EAs that assume each function as a human evaluation model. Therefore, ELDR has high potential as a new surrogate model for SAEA.

Conclusion and future work

This paper proposed a novel ranking-based surrogate model, called ELDR, for SAEA that combines ELM and DirectRanker. Further, it was incorporated into SAHO, a state-of-the-art SAEA. To investigate the estimation accuracy of ELDR, this study compared the estimation accuracy of ELDR with that of RBF and RankSVM, which are commonly used in SAEAs. The comparison results indicated

that ELDR with an appropriate hyperparameter setting exhibited the highest rank estimation accuracy compared to RBF and RankSVM, especially when the amount of training data was small. To investigate the compatibility of ELDR with SAEA, ELDR-SAHO was compared with existing SAEAs. The experimental results showed that ELDR-SAHO significantly outperforms existing SAEAs using regression and classification models, including RBF-based SAHO, on many problems, particularly on high-dimensional ($d \geq 50$) problems. Based on these results, it is concluded that ELDR is a promising surrogate model for SAEAs. However, because the combination of SAHO and ELDR may fall into the local optimum, future research should explore a different SAEA that can fully utilize the power of ELDR.

Because ELDR is a NN-based method, it has the advantage that, unlike RBF, it can be applied to both discrete and mixed variable optimization problems and real-valued optimization problems. In addition, a ranking-based surrogate model is useful for constrained and aesthetic optimizations. Therefore, future research should examine the application of ELDR to these problem domains.

Funding Japan Society for the Promotion of Science 21K17826.

Declarations

Conflict of interest The author declares no conflicts of interest associated with this manuscript.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Jin Y (2011) Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol Comput 1(2):61–70. https://doi.org/10.1016/j.swevo.2011.05.001
2. Jin Y, Wang H, Sun C (2021) Data-driven evolutionary optimization. Springer. https://doi.org/10.1007/978-3-030-74640-7
3. Fujio C, Ogawa H (2021) Physical insight into axisymmetric scramjet intake design via multi-objective design optimization using surrogate-assisted evolutionary algorithms. Aerosp Sci Technol 113:106676. https://doi.org/10.1016/j.ast.2021.106676
4. Rozek M, Ogawa H, Ueda S, Ikenaga T et al (2019) Multi-objective optimisation of nrho-llo orbit transfer via surrogate-assisted evolutionary algorithms. In: AIAC18: 18th Australian International Aerospace Congress (2019): HUMS-11th Defence Science and Technology (DST) International Conference on Health and Usage Monitoring (HUMS 2019): ISSFD-27th International Symposium on Space Flight Dynamics (ISSFD), p. 1001. Engineers Australia, Royal Aeronautical Society
5. Urquhart M, Ljungskog E, Sebben S (2020) Surrogate-based optimisation using adaptively scaled radial basis functions. Appl Soft Comput 88:106050. https://doi.org/10.1016/j.asoc.2019.106050
6. Chugh T, Chakraborti N, Sindhya K, Jin Y (2017) A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem. Mater Manuf Process 32(10):1172–1178. https://doi.org/10.1080/10426914.2016.1269923
7. Tong H, Huang C, Minku LL, Yao X (2021) Surrogate models in evolutionary single-objective optimization: a new taxonomy and experimental study. Inform Sci 562:414–437. https://doi.org/10.1016/j.ins.2021.03.002
8. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501. https://doi.org/10.1016/j.neucom.2005.12.126. (**Neural Networks**)
9. Wang J, Lu S, Wang S-H, Zhang Y-D (2021) A review on extreme learning machine. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-11007-7
10. Köppel M, Segner A, Wagener M, Pensel L, Karwath A, Kramer S (2020) Pairwise learning to rank by neural networks revisited: reconstruction, theoretical analysis and practical performance. In: Brefeld U, Fromont E, Hotho A, Knobbe A, Maathuis M, Robardet C (eds) Machine learning and knowledge discovery in databases. Springer, Cham, pp 237–252
11. Pan J-S, Liu N, Chu S-C, Lai T (2021) An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems. Inform Sci 561:304–325. https://doi.org/10.1016/j.ins.2020.11.056
12. Wang H, Jin Y, Doherty J (2017) Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. IEEE Trans Cybern 47(9):2664–2677. https://doi.org/10.1109/TCYB.2017.2710978
13. Yu H, Tan Y, Sun C, Zeng J (2019) A generation-based optimal restart strategy for surrogate-assisted social learning particle swarm optimization. Knowl Based Syst 163:14–25. https://doi.org/10.1016/j.knosys.2018.08.010
14. Cai X, Qiu H, Gao L, Jiang C, Shao X (2019) An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems. Knowl Based Syst 184:104901. https://doi.org/10.1016/j.knosys.2019.104901
15. Li F, Shen W, Cai X, Gao L, Gary Wang G (2020) A fast surrogate-assisted particle swarm optimization algorithm for computationally expensive problems. Appl Soft Comput 92:106303. https://doi.org/10.1016/j.asoc.2020.106303
16. Hussain MF, Barton RR, Joshi SB (2002) Metamodeling: radial basis functions, versus polynomials. Eur J Oper Res 138(1):142–154. https://doi.org/10.1016/S0377-2217(01)00076-5
17. Huang C, Radi B, Hami AE, Bai H (2018) CMA evolution strategy assisted by kriging model and approximate ranking. Appl Intell 48:4288–4304. https://doi.org/10.1007/s10489-018-1193-3
18. Zhan D, Xing H (2021) A fast kriging-assisted evolutionary algorithm based on incremental learning. IEEE Trans Evol Comput 25(5):941–955. https://doi.org/10.1109/TEVC.2021.3067015
19. Liu B, Zhang Q, Gielen GGE (2014) A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Trans Evol Comput 18(2):180–192. https://doi.org/10.1109/TEVC.2013.2248012
20. Hildebrandt T, Branke J (2015) On using surrogates with genetic programming. Evol Comput 23(3):343–367. https://doi.org/10.1162/EVCO_a_00133
21. Pavelski LM, Delgado MR, Almeida CP, Gonçalves RA, Venske SM (2016) Extreme learning surrogate models in multi-objective

optimization based on decomposition. Neurocomputing 180:55–67. https://doi.org/10.1016/j.neucom.2015.09.111. (**Progress in Intelligent Systems Design**)
22. Pan L, He C, Tian Y, Wang H, Zhang X, Jin Y (2019) A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. IEEE Trans Evol Comput 23(1):74–88. https://doi.org/10.1109/TEVC.2018.2802784
23. Svozil D, Kvasnicka V, Pospichal J (1997) Introduction to multilayer feed-forward neural networks. Chemometr Intell Lab Syst 39(1):43–62. https://doi.org/10.1016/S0169-7439(97)00061-0
24. Sonoda T, Nakata M (2020) MOEA/D-S3: MOEA/D using SVM-based surrogates adjusted to subproblems for many objective optimization. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. https://doi.org/10.1109/CEC48606.2020.9185549
25. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. COLT '92, pp. 144–152. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/130385.130401
26. Wei F-F, Chen W-N, Yang Q, Deng J, Luo X-N, Jin H, Zhang J (2021) A classifier-assisted level-based learning swarm optimizer for expensive optimization. IEEE Trans Evol Comput 25(2):219–233. https://doi.org/10.1109/TEVC.2020.3017865
27. Friedman JH (2002) Stochastic gradient boosting. Comput Stat Data Anal 38(4):367–378. https://doi.org/10.1016/S0167-9473(01)00065-2. (**Nonlinear Methods and Data Mining**)
28. Yang Q, Chen W-N, Deng JD, Li Y, Gu T, Zhang J (2018) A level-based learning swarm optimizer for large-scale optimization. IEEE Trans Evol Comput 22(4):578–594. https://doi.org/10.1109/TEVC.2017.2743016
29. Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317. https://doi.org/10.1109/ICEC.1996.542381
30. Runarsson TP (2006) Ordinal regression in evolutionary computation. In: Runarsson TP, Beyer H-G, Burke E, Merelo-Guervós JJ, Whitley LD, Yao X (eds) Parallel problem solving from nature-PPSN IX. Springer, Heidelberg, pp 1048–1057
31. Loshchilov I, Schoenauer M, Sebag M (2012) Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. GECCO '12, pp. 321–328. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2330163.2330210
32. Lu X, Tang K, Sendhoff B, Yao X (2014) A new self-adaptation scheme for differential evolution. Neurocomputing 146:2–16. https://doi.org/10.1016/j.neucom.2014.04.071
33. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
34. Joachims T (2002) Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '02, pp. 133–142. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/775047.775067
35. Hao H, Zhang J, Lu X, Zhou A (2020) Binary relation learning and classifying for preselection in evolutionary algorithms. IEEE Trans Evol Comput 24(6):1125–1139. https://doi.org/10.1109/TEVC.2020.2986348
36. Hao H, Zhou A, Qian H, Zhang H (2022) Expensive multiobjective optimization by relation learning and prediction. IEEE Trans Evol Comput 26(5):1157–1170. https://doi.org/10.1109/TEVC.2022.3152582
37. Mezura-Montes E, Coello Coello CA, Tun-Morales EI (2004) Simple feasibility rules and differential evolution for constrained optimization. In: Mexican International Conference on Artificial Intelligence, pp. 707–716. Springer
38. Takahama T, Sakai S, Iwane N (2005) Constrained optimization by the ϵ constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In: Zhang S, Jarvis R (eds) AI 2005: advances in artificial intelligence. Springer, Heidelberg, pp 389–400
39. Takagi H (2001) Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. Proc IEEE 89(9):1275–1296. https://doi.org/10.1109/5.949485
40. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B (Cybernetics) 42(2):513–529. https://doi.org/10.1109/TSMCB.2011.2168604
41. Huang G-B, Siew C-K (2004) Extreme learning machine: RBF network case. In: ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004., vol. 2, pp. 1029–10362. https://doi.org/10.1109/ICARCV.2004.1468985
42. Huang G-B (2015) What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle. Cognit Comput 7:263–278
43. Dudek G (2016) Extreme learning machine as a function approximator: Initialization of input weights and biases. In: Burduk R, Jackowski K, Kurzyński M, Woźniak M, Żołnierek A (eds.) Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, pp. 59–69. Springer, Cham
44. Rao RV, Savsani VJ, Vakharia DP (2012) Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inform Sci 183(1):1–15. https://doi.org/10.1016/j.ins.2011.08.006
45. Rao RV, Patel V (2013) An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. Sci Iran 20(3):710–720. https://doi.org/10.1016/j.scient.2012.12.005
46. McKay MD, Beckman RJ, Conover WJ (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2):239–245 (**Accessed 2022-04-12**)
47. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. KanGAL Rep 2005005(2005):2005
48. Peng F, Tang K, Chen G, Yao X (2010) Population-based algorithm portfolios for numerical optimization. IEEE Trans Evol Comput 14(5):782–800. https://doi.org/10.1109/TEVC.2010.2040183
49. Yu H, Tan Y, Zeng J, Sun C, Jin Y (2018) Surrogate-assisted hierarchical particle swarm optimization. Inform Sci 454–455:59–72. https://doi.org/10.1016/j.ins.2018.04.062
50. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232. https://doi.org/10.1214/aos/1013203451
51. Dunn OJ (1961) Multiple comparisons among means. J Am Stat Assoc 56(293):52–64. https://doi.org/10.1080/01621459.1961.10482090

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.