Complex & Intelligent Systems (2024) 10:1639–1660
https://doi.org/10.1007/s40747-023-01199-w

CASE STUDY

A two-stage multiobjective evolutionary ensemble learning for silicon prediction in blast furnace

Qiang Li[1] · Jingchuan Zhang[2] · Wenhao Wang[3] · Xianpeng Wang[1]

Received: 8 December 2022 / Accepted: 15 July 2023 / Published online: 19 August 2023

Abstract
Silicon content of molten iron is an important indicator in the blast furnace ironmaking process. Accurate prediction of silicon content is very important for monitoring the operating condition of the blast furnace and the quality of the molten iron. However, accurate and effective online prediction of silicon content is a challenging task due to the complex and high-dimensional nonlinear relationship between silicon content and process variables. Therefore, a two-stage multiobjective evolutionary ensemble learning algorithm is proposed to achieve a high-accuracy and low-complexity prediction model using support vector machine (SVR) as the base learner. In the first stage, a non-dominated sorting differential evolution algorithm with dynamic resource allocation (DRA-NSDE) is proposed to generate a set of non-dominated solutions (SVRs) with the objectives of accuracy and complexity. In the second stage, a stacking method based on clustering and differential evolution (CDE-Stacking) is proposed to select base learners with better diversity and construct the ensemble model. The effectiveness of the proposed DRA-NSDE and CDE-Stacking strategies is verified through a series of numerical experiments. The experimental results show that the proposed algorithm outperforms the rival methods on both the UCI benchmark data set and the actual blast furnace data set. In addition, the analysis of model complexity shows that the proposed model can achieve higher prediction accuracy with relatively low model complexity, which indicates that the algorithm is very suitable for the online prediction of silicon content in actual blast furnace ironmaking process.

Keywords Support vector regression · Multiobjective evolutionary ensemble learning · Silicon content prediction

توقع محتوى السيليكون.التعلم الجماعي التطوري متعدد الأهداف.دعم الانحدار المتجه

Qiang Li and Jingchuan Zhang have contributed equally to this work.

This research was supported by the Major Program of National Natural Science Foundation of China (72192830, 72192831), the Fund for the National Natural Science Foundation of China (62073067), the 111 Project (B16009), and the Fundamental Research Funds for the Central Universities (N2128001).

✉ Xianpeng Wang
wangxianpeng@ise.neu.edu.cn

1 National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Shenyang 110819, China

2 Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang 110819, China

3 Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Liaoning Key Laboratory of Manufacturing System and Logistics Optimization, Shenyang 110819, China

Introduction

As the main method of modern ironmaking, blast furnace ironmaking is the most critical process in iron and steel industry. As shown in Fig. 1, a blast furnace can be divided into five zones from top to bottom: throat, shaft, belly, bosh, and hearth. In practical production, the thermal state of the blast furnace is a crucial indicator of its output, energy consumption, and service life. However, as a typical black-box system, it is difficult to directly measure the internal thermal state of the blast furnace due to the limitations of measurement and sensor technology [1, 2]. Many studies have demonstrated that the silicon content of molten iron can indirectly characterize the blast furnace temperature without loss of information, and its trend can reflect the quality of the molten iron and the operating condition of the blast furnace [3]. High silicon content indicates that the blast furnace is overheating, resulting in unnecessary energy loss [4]. On the contrary,

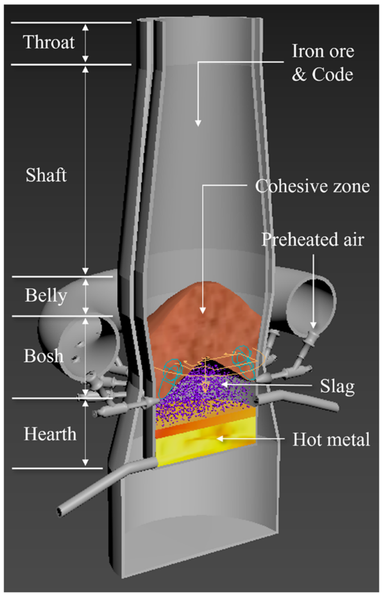


Fig. 1 Blast furnace ironmaking process

low silicon content indicates a low blast furnace temperature, which severely affects the iron output and often takes weeks to recover [5]. Therefore, it is important to establish a prediction model for the silicon content of molten iron to provide guidance for the actual production of the blast furnace ironmaking process.

Due to powerful data mining capability and less reliance on expert knowledge, data-driven algorithms for silicon content prediction have gradually become the focus of research [4, 6–8]. However, many existing works only applied a single model to predict the silicon content [9–11], which may not fully satisfy the accuracy requirement due to the complex and high-dimensional nonlinear relationship between silicon content and process variables. Therefore, in recent years, some sophisticated models have been proposed to address this problem. Liu et al. [12] proposed a novel silicon content prediction method called T-HyperGAT, which combined the hypergraph attention network (HyperGAT) and the gated recurrent unit (GRU) network to capture the high-order correlations and time-series characteristics from complex industrial data. Li et al. [13] proposed a context-aware enhanced GRU network with feature-temporal attention to predict silicon content, which allowed for enhanced local awareness and soft alignment of variables. Zhao et al. [14] proposed an ameliorated Moth-Flame optimization (AMFO) algorithm to optimize the parameters of fast learning network (FLN) to build the prediction model of silicon content. These algorithms aim to achieve better silicon content prediction performance by building more complex models, but with a corresponding increase in computational cost.

Support vector machine (SVR) is a promising machine learning algorithm that can deal with high-dimensional nonlinear prediction problems. However, the hyperparameters of SVR need to be adjusted carefully to achieve promising performance. Due to the powerful global search ability, evolutionary algorithms (EAs) have been widely used to search the optimal hyperparameters of SVR [15–18] and other industrial parameter optimization tasks [19–21] (e.g., Stojanovic et al. proposed the optimal tuning of cascade load force controllers for a parallel robot platform in [20]). However, most existing works [22–25] only considered the accuracy of SVR and ignored the complexity of the model, which might not be adequate for online prediction tasks in actual industrial processes.

Multiobjective evolutionary ensemble learning (MOEEL) algorithms have gradually become the research focus in various prediction problems [26–30]. MOEEL is suitable for dealing with real industrial data such as silicon content for two main reasons: (1) MOEEL aims to construct an ensemble model based on a series of high-quality base learners in order to better handle the complex and high-dimensional nonlinear correlations between silicon content and process variables; and (2) MOEEL usually takes the complexity and accuracy of base learners as two conflicting optimization objectives to achieve a balance between model accuracy and generalization, which is crucial for online prediction of silicon content in actual blast furnace ironmaking process. Wang et al. [31] proposed a multiobjective sparse nonlinear ensemble learning with evolutionary feature selection (MOSNE-EFS) to achieve the precise prediction of the strip hardness. Singh and Singh [32] developed a stacking-based evolutionary ensemble learning system called NSGA-II-Stacking to predict the onset of Type-2 diabetes mellitus (T2DM). Hu et al. [4] proposed a multiobjective evolutionary nonlinear ensemble learning model with evolutionary feature selection mechanism (MOENE-EFS) for silicon content prediction, which can efficiently select input features and construct the ensemble model in a nonlinear way. However, these works usually adopted traditional multiobjective evolutionary algorithm in training base learners and ignored the balance between local search and global search, which may suffer from falling into local optima with low diversity [31, 33–35].

To address the above issues, this paper proposes a two-stage multiobjective evolutionary ensemble learning algorithm for silicon content prediction of blast furnace, in which support vector regression (SVR) is adopted as the base learner. The proposed algorithm can obtain a series of non-dominated solutions with high accuracy and diversity, based on which an ensemble model with good prediction performance and low model complexity can be constructed. The main contributions in this paper are summarized as follows:

(1) Unlike traditional methods that only take the prediction accuracy of SVR as the optimization objective, the proposed method optimizes the accuracy and complexity of SVR at the same time, and a prediction model with high accuracy and low model complexity can be obtained.
(2) In the first stage, a non-dominated sorting differential evolution with dynamic resource allocation (DRA-NSDE) is proposed to achieve the balance between local and global search through dynamic resource allocation, which results in a series of base learners with high accuracy and diversity.
(3) In the second stage, a stacking ensemble method based on clustering and differential evolution (CDE-Stacking) is proposed to select and ensemble the base learners, in which the K-Means clustering is utilized to ensure the diversity of base learners, and the single-objective DE is used to optimize the ensemble weights of selected base learners.

The remainder of this paper is organized as follows: "Related works" section briefly introduces the related works. "The proposed algorithm" section elaborates the framework and details of the proposed two-stage MOEEL algorithm. "Experiments" section consists of the analysis of the proposed DRA-NSDE and CDE-Stacking strategies and the performance analysis of the proposed two-stage MOEEL algorithm on both the UCI benchmark data set and the actual blast furnace data set. Finally, "Conclusion" section concludes this paper and introduces our future work.

Related works

Support vector regression

Unlike traditional regression methods, SVR constructs a fault-tolerant band and makes as many points as possible fall into it. When $|f(x) - y| \leqslant \varepsilon$, i.e., the bias between the predicted value and the actual value is within the tolerance range, the prediction is considered correct.

By maximizing the interval and minimizing the empirical risk, SVR can obtain the optimal regression model. A maximizing interval problem can be transformed into a minimizing interval problem by taking the inverse and adding the empirical risk loss term into the objective function, as shown in Eq. (1).

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} l_\varepsilon(f(x_i) - y_i) \tag{1}$$

where C is the regularization parameter, and l_ε is the ε-insensitive loss function.

$$l_\varepsilon(z) = \begin{cases} 0, & |z| \leqslant \varepsilon \\ |z| - \varepsilon, & |z| > \varepsilon \end{cases}$$

However, setting ε by experience can not ensure that most of the data are located within the interval band, therefore, it is necessary to add slack variables for each sample so that some samples are allowed to be located outside the interval band. After introducing the slack variables $\xi, \hat{\xi}$, the optimization objective shown in Eq. (1) is changed to the following form [36]:

$$\begin{aligned} &\min_{\omega,b,\xi_i,\hat{\xi}_i} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) \\ &\text{s.t. } f(x_i) - y_i \leqslant \varepsilon + \xi_i \\ &y_i - f(x_i) \leqslant \varepsilon + \hat{\xi}_i \\ &\xi_i \geqslant 0, \hat{\xi}_i \geqslant 0, i = 1, 2, \ldots, m \end{aligned}$$

The following Lagrangian function [37] can be obtained by introducing the Lagrangian multipliers $u_i \geqslant 0, \hat{u}_i \geqslant 0, \alpha_i \geqslant 0, \hat{\alpha}_i \geqslant 0$ to remove the constraints:

$$\begin{aligned} &L(\omega, b, \alpha, \hat{\alpha}, \xi, \hat{\xi}, u, \hat{u}) \\ &= \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) \\ &\quad - \sum_{i=1}^{m} u_i\xi_i - \sum_{i=1}^{m}\hat{u}_i\hat{\xi}_i \\ &\quad + \sum_{i=1}^{m}\alpha_i(f(x_i) - y_i - \varepsilon - \xi_i) \\ &\quad + \sum_{i=1}^{m}\hat{\alpha}_i(y_i - f(x_i) - \varepsilon - \hat{\xi}_i) \end{aligned}$$

Substituting $f(x) = \omega^T x + b$ into the Lagrangian function and making the partial derivative of $\omega, b, \xi, \hat{\xi}$ equal to zero, the following equations can be obtained:

$$\begin{cases} \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)x_i = \omega \\ \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) = 0 \\ u_i + \alpha_i = C \\ \hat{u}_i + \hat{\alpha}_i = C \end{cases}$$

According to KKT Conditions [37]:

$$\begin{cases} \alpha_i(f(x_i) - y_i - \varepsilon - \xi_i) = 0 \\ \alpha_i(y_i f - (x_i) - \varepsilon - \xi_i) = 0 \\ \alpha_i\hat{\alpha_i} = 0 \\ \xi_i\hat{\xi_i} = 0 \\ (C - \alpha_i)\xi_i = 0 \\ (C - \hat{\alpha_i})\hat{\xi_i} = 0 \end{cases}$$

the pairwise problem of SVR can be obtained as follows:

$$\begin{aligned} \max_{\alpha,\hat{\alpha}} & \sum_{i=1}^{m} y_i(\hat{\alpha_i} - \alpha_i) - \varepsilon(\hat{\alpha_i} + \alpha_i) \\ & - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha_i} - \alpha_i)(\hat{\alpha_j} - \alpha_j)x_i^T x_j \\ \text{s.t.} & \sum_{i=1}^{m}(\hat{\alpha_i} - \alpha_i) = 0 \\ & 0 \leqslant \alpha_i, \hat{\alpha_i} \leqslant C \end{aligned}$$

By solving the above optimization problem, the regression function of SVR can be obtained as Eq. (2).

$$f(x) = \sum_{i=1}^{m}(\hat{\alpha_i} - \alpha_i)x_i^T x + b \tag{2}$$

When dealing with the nonlinear regression problem, SVR generally maps x to a high-dimensional space by $\phi(x)$, and consequently the nonlinear problem is transformed into an approximately linear regression problem. However, as the number of variables becomes larger, the problem dimensionality grows exponentially, which significantly increases the computational resource. This problem can be solved by introducing kernel function into SVR to avoid the tremendous inner product computation in the high-dimensional feature space. The commonly used kernel function are Gaussian kernel function, polynomial kernel function, Laplace kernel function and Sigmoid kernel function. After introducing kernel functions into SVR, Eq. (2) can be rewritten as:

$$\begin{aligned} f(x) &= \omega^T\phi(x) + b \\ &= \sum_{i=1}^{m}(\hat{\alpha_i} - \alpha_i)\phi(x_i)^T\phi(x) + b \\ &= \sum_{i=1}^{m}(\hat{\alpha_i} - \alpha_i)k(x, x_i) + b \end{aligned} \tag{3}$$

Multiobjective evolutionary ensemble learning

In ensemble learning, the construction and integration of base learners are two important processes. According to the "error-ambiguity decomposition" theory [38], the ensemble performance depends heavily on whether the prediction accuracy and diversity of the base learners are well balanced. Even if the base learners with high accuracy and strong diversity are obtained, the ensemble performance will not be satisfactory if an inappropriate ensemble strategy is chosen. Therefore, various ensemble strategies have been proposed, such as AdaBoost [39] and Random Forest [40], which have specific steps to build the ensemble model. In multiobjective evolutionary algorithms (MOEAs), multiple Pareto optimal solutions can be obtained by one evolution, and the corresponding non-dominated solution set can meet the accuracy and diversity requirements of ensemble learning. Therefore, multiobjective evolutionary ensemble learning (MOEEL) is very suitable for the silicon content prediction problem of blast furnace [4].

Algorithm 1 Framework of MOEEL

```
Require: Data set; parameters of MOEA; base learner
Ensure: Ensemble model
1: Design solution encoding scheme for base learners and initialize the
   population.
2: Define the evaluation metrics to evaluate the base learners in terms
   of model accuracy and complexity, and use them as the objective
   function of the solution.
3: Using appropriate evolutionary operators and environment selec-
   tion strategies to generate offspring solutions and update the current
   population. Repeat this step until the stop condition is met.
4: Select base learners from the last population to construct the ensem-
   ble model with an appropriate selection strategy.
5: Construct the ensemble model with the selected base learners based
   on the appropriate ensemble strategy.
```

The framework of MOEEL is shown in Algorithm 1, which includes two stages. In the first stage (lines 1–3), the base learners are encoded as solutions in the population, and then a suitable multiobjective evolutionary algorithm is used to search for base learners with high accuracy and strong diversity. To achieve this purpose, we need to design the encoding scheme, define objective functions for base learners, and choose appropriate genetic operators as well as the environment selection method. The second stage (lines 4–5) is to select appropriate base learners and then integrate them in some way to construct the final ensemble model.

The proposed algorithm

Algorithm overview

Algorithm 2 outlines the overall framework of the proposed two-stage MOEEL algorithm, where the contributions are highlighted in italic. In the first stage (lines 1–6), the initial population is generated and evaluated according to the proposed solution encoding scheme and two objective functions. Then the non-dominated solution set consisting of base learners with high accuracy and strong diversity is obtained based on the proposed DRA-NSDE algorithm. In the second stage (line 7), the obtained base learners are integrated through the proposed CDE-Stacking method to construct the ensemble model with high accuracy and generalization ability for silicon content prediction.

Algorithm 2 Framework of the Proposed Algorithm

```
Require: Training and validation data; algorithm parameters in two
    stages; base learner SVR
Ensure: Ensemble model $EM$
    // Stage 1: Training of base learners with DRA-NSDE
1: $P$ ← Generate the initial population according to the solution encod-
    ing scheme described in “Encoding and decoding” section.
2: for each solution $x$ in $P$ do
3:     Decode $x$ to a SVR model.
4:     Calculate the evaluation metrics of the SVR model and take them
    as the objective values.
5: end for
6: $NDSet$ ← Evolve the population based on *the proposed DRA-
    NSDE algorithm* and get the non-dominate solution set.
    // Stage 2: Ensemble of selected base learners
7: $EM$ ← Select appropriate base learners and construct the final
    ensemble model based on *the CDE-Stacking method*.
```

Encoding and decoding

As shown in Fig. 2, a solution in the population is encoded as a vector which has two parts: the feature selection part and the hyperparameter part. The feature selection part is a binary vector whose length equals to the total number of features. In the binary vector, 1 indicates that the feature is selected to construct the base learner while 0 indicates that the feature is not selected. The hyperparameter part is a real number vector which contains three genes, representing three hyperparameters in SVR, i.e., regularization parameter C, width coefficient γ in Gaussian kernel function and insensitive loss coefficient ε. Please note that the upper and lower bounds of the hyperparameter part are determined by pre-experiments, i.e., multiple experiments on a single SVR model to roughly determine the range of values of C, γ and ε.

The solution needs to be decoded to get its objective function values. First, the features used to build the base learner are determined from the feature selection section, and the corresponding training data can be extracted from the blast furnace dataset. Then three SVR hyperparameters are obtained from the hyperparameters section, and the SVR model can be constructed and trained. After the training is completed, the SVR model will be evaluated on the validation set and the evaluation results are considered as the objective function values of this solution.

Noted that the upper and lower bounds for the hyperparameter part are determined by pre-experiments. Several experiments are conducted on a single SVR model before the formal experiments in which we can roughly determine the range of values for C, γ and ε. The upper and lower bounds of parameters obtained in pre-experiments are shown in Table 1.

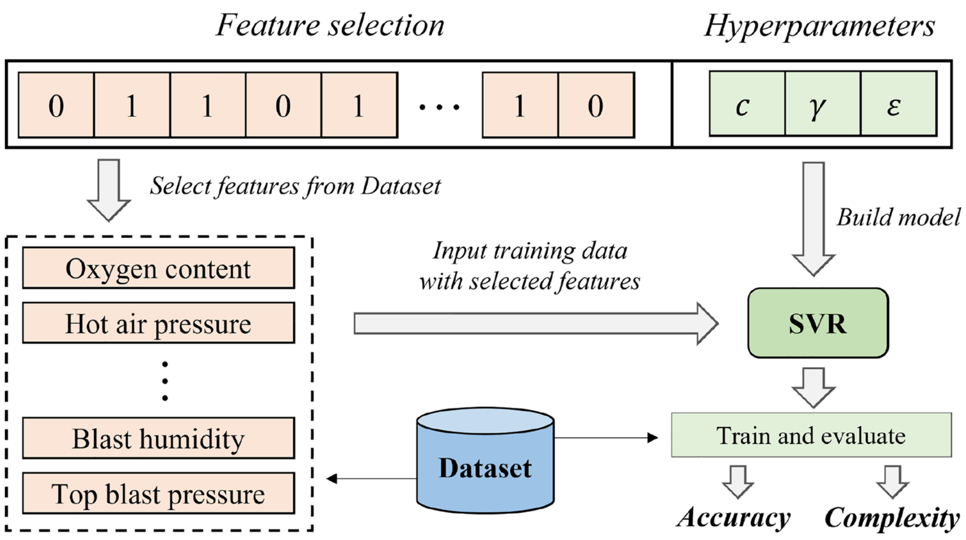


Fig. 2 Schematic of the solution encoding and decoding

Table 1 Upper bounds and lower bounds of parameters in the hyperparameter part

Parameter	Name	Lower bounds	Upper bounds
Regularization parameter	C	0.004	10.000
Width coefficient	γ	0.004	10.000
Insensitive loss coefficient	ε	0.0	0.4

Objective functions

In the silicon content prediction problem of blast furnace, the prediction accuracy and the model complexity are two conflicting optimization objectives to guide the evolutionary search direction of the algorithm.

Accuracy

The first objective is to maximize the prediction accuracy of the model. In this paper, root mean square error loss (RMSE) is used as the accuracy metric, which is defined as Eq. (4).

$$\text{RMSE} = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(\hat{y}_i - y_i)^2} \tag{4}$$

where L is the number of samples, $\hat{y}_i$ is the predicted value for the ith sample, and y_i is the true value of the ith sample. It should be noted that a smaller RMSE indicates a higher accuracy.

Complexity

The second objective is to minimize the complexity of the model. In this paper, the number of support vectors in the SVR model is used as the complexity metric, which is defined as Eq. (5).

$$\text{CMPLX} = N_{support} \tag{5}$$

where CMPLX stands for model complexity and $N_{support}$ is the number of support vectors in the SVR model.

Mutation and crossover

The mutation operator used in the proposed algorithm is DE/rand/1/bin [41]. In DE/rand/1/bin, for each target vector x_i, its associated mutant vector v_i can be generated by Eq. (6).

$$v_i = x_{r_1} + F(x_{r_2} - x_{r_3}) \tag{6}$$

where $x_{r_1}, x_{r_2}, x_{r_3}$ are three randomly selected solutions from the population and r_1, r_2, r_3 are mutually exclusive integers randomly generated with the range $[1, N]$ (N is the population size). The scaling factor F is a positive control parameter for scaling the difference vector.

Since the solution in the proposed algorithm is encoded as a vector that contains both binary part and real part, some necessary modifications have to be made in the original DE mutation operator. The mutation of the hyperparameter part (real encoded) between two solutions is the same as Eq. (6), while the mutation of the feature selection part (binary encoded) is performed as follows [4]:

Step 1: Apply an Exclusive-OR operation on the feature selection part of x_{r_2} and x_{r_3} to obtain a binary vector x_{temp}.

Step 2: Generate a random number R between $[0, 1)$ for each gene in the feature selection part of x_{temp} if its value is 1. Compare this random number with the difference weight F. If $R < F$, the binary number of this gene is changed from 1 to 0, otherwise no operation is performed.

Step 3: Perform OR operation on the feature selection part of the obtained x_{temp} and x_{r_1} to obtain the mutant vector v_i.

According to DE/rand/1/bin, the trial vector u_i can be obtained by the crossover operation on the mutant vector v_i and the target vector x_i, as shown in Eq. (7).

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } r_j < CR \text{ or } j = i \\ x_{i,j}, & otherwise \end{cases} \tag{7}$$

where $u_{i,j}$ denotes the jth gene on the ith solution in the population, r_j is a random number between $[0, 1)$, and CR is the crossover probability. Note that there is no selection operation after the mutation and crossover operation compared with the original DE algorithm. Instead, the trial vector u_i is also regarded as an offspring solution and added to the offspring population directly.

DRA-NSDE

A variety of crossover operators have been designed for the differential evolution (DE) algorithm, each with a different search behavior. For example, the DE/rand/1/bin operator focuses on maintaining good diversity and therefore con-

Algorithm 3 DRA-NSDE

```
Require: Training and validation data; maximum generation $G$; population size $N$; the number of clusters $K$
Ensure: Non-dominated solution set $NDSet$
1: Set current generation $g = 0$.
2: while $g < G$ do
3:     $\mathbb{P} \leftarrow$ Cluster the population $P$ into $K$ clusters with K-Means algorithm and obtain the sub-population set.
4:     $\rho_L, \rho_G \leftarrow$ Get local search and global search ratio.
5:     $Q \leftarrow$ Initialize the offspring population with $\phi$.
       // Local search
6:     for each $P_k (k := 1 \ to \ K)$ in $\mathbb{P}$ do
7:         for $i \leftarrow 1 \ to \ \rho_L * |P_k|$ do
8:             $x_{r_1}, x_{r_2}, x_{r_3} \leftarrow$ Randomly select three solutions from the sub-population $P_k$.
9:             $v_i \leftarrow$ Perform Mutation operator on $x_{r_1}, x_{r_2}, x_{r_3}$.
10:            $u_i \leftarrow$ Perform Crossover operator on the current solution $x_i$ and $v_i$.
11:            $Q \leftarrow Q \cup u_i$;
12:        end for
13:    end for
       // Global search
14:    for $i \leftarrow 1 \ to \ \rho_G * N$ do
15:        $x_{r_1}, x_{r_2}, x_{r_3} \leftarrow$ Randomly select three solutions from the current population $P$;
16:        $v_i \leftarrow$ Perform Mutation operator on $x_{r_1}, x_{r_2}, x_{r_3}$;
17:        $u_i \leftarrow$ Perform Crossover operator on the current solution $x_i$ and $v_i$;
18:        $Q \leftarrow Q \cup u_i$;
19:    end for
20:    $J \leftarrow$ Evaluate the offspring population $Q$ and combine it with the current population $P$ to obtain the joint population;
21:    $P \leftarrow$ Perform environment selection on $J$ and obtain the new population into the next generation;
22:    $\rho_L, \rho_G \leftarrow$ Update local search ratio and global search ratio based on their success rate;
23:    $g \leftarrow g + 1$;
24: end while
25: $NDSet \leftarrow$ Obtain the non-dominated solution set from the final population after the evolution process is done.
```

verges relatively slowly, while the DE/best/1/bin operator converges faster but tends to lose good diversity. To achieve a good balance between exploration and exploitation, the dynamic allocation strategy of computational resources is proposed in the DRA-NSDE. The algorithm details are presented in Algorithm 3.

The schematic of local search and global search in Algorithm 3 is shown in Fig. 3. First, the K-Means clustering algorithm is performed on the current population P in the objective space, and the population P will be divided into K sub-populations, i.e., $\mathbb{P} = \{P_1, P_2, \ldots, P_K\}$. Then we get the local search ratio ρ_L and the global search ratio ρ_G in the current evolution process, where ρ_L is used to calculate the number of offspring solutions N_L^k generated only from sub-population $P_k (k = 1, \ldots, K)$, and ρ_G is used to calculate the number of offspring solutions N_G generated from the whole population P (see Eqs. (8) and (9)). In the following local search process, for each sub-population P_k in set $\mathbb{P}$, we first select three different solutions to perform the mutation and crossover operation described in "Mutation and crossover" section to generate the offspring solution u_i, and then add it to the offspring population Q. This process will be repeated until N_L^k offspring solutions are generated and added to Q. After the local search is completed on all sub-populations, the global search begins subsequently. Different from the local search, the three random solutions are selected from the whole population P. The subsequent mutation and crossover operations are consistent with the local search. This process will also be repeated until N_G offspring solutions are generated and added to Q. After the offspring population Q is obtained, Q will be combined with the current population P to get the joint population. Based on the non-dominated ranking and crowding distance [42], the environment selection operation is performed on the joint population to obtain a new population for the next generation.

$$N_L^k = \rho_L * |P_k| \tag{8}$$

$$N_G = \rho_G * |P| \tag{9}$$

To achieve the dynamic resource allocation, the algorithm needs to maintain two variables, i.e., the local search ratio ρ_L and the global search ratio ρ_G during the evolution process. At the beginning of the algorithm, these two variables are set to be equal, i.e., $\rho_L = \rho_G = 0.5$, so that each of the two searching strategies generates half of the solutions in the offspring population Q. After the environment selection, a new population that will enter the next generation can be obtained, and we can count the following four numbers: ns_L, nf_L, ns_G, nf_G. Among all the offspring solutions generated from the local search, the number of solutions that successfully enter the next generation is counted as ns_L, while the number of solutions that failed to enter the next generation is counted as nf_L. Similarly, among all the offspring solutions generated from the global search, the number of solutions that successfully enter the next generation is counted as ns_G and the number of solutions that failed to enter the next generation is counted as nf_G. These four numbers are accumulated after LP (learning period) generations, and then the two ratios ρ_L and ρ_G can be updated according to Eqs. (10) and (11) [43].

$$\rho_L = \frac{ns_L}{ns_L + nf_L} \tag{10}$$

$$\rho_G = \frac{ns_G}{ns_G + nf_G} \tag{11}$$

The above equations denote the success rate of the two searching strategies in a given learning period. After the initial LP generations, the local search ratio ρ_L and the global search ratio ρ_G will be dynamically updated in each generation, so that the computational resources can be allocated

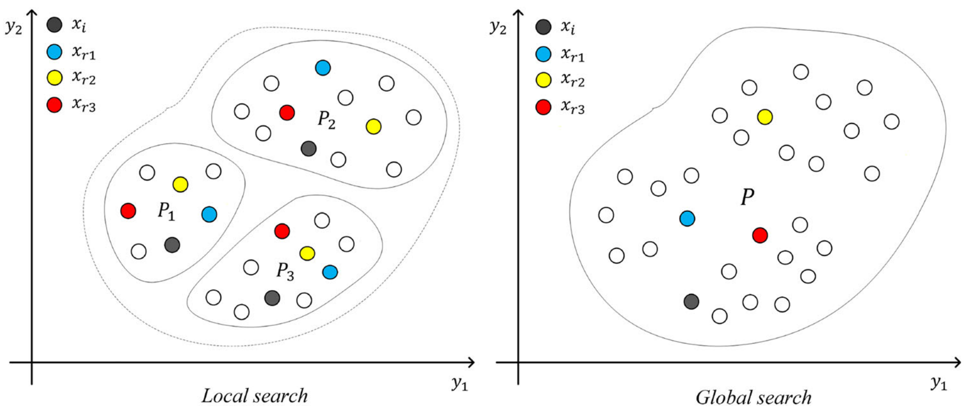


Fig. 3 Local search and global search ($K = 3$)

dynamically at different stages of the evolution. In this way, the algorithm can avoid falling into local optimal point and have a good ability to maintain better diversity. Note that after each update of the two searching strategy ratios, their success number ns and fail number nf recorded in the earliest generation need to be cleared from the archive to ensure that the update of the two ratios relies only on the most recent LP generations, thus eliminating the potential negative impact at the early stage of the evolution.

After the evolution process is completed, a fast non-dominated sorting [42] is performed on the population in the last generation. Then all solutions on the first non-dominated layer are extracted to obtain the final non-dominated solution set NDSet.

In the proposed DRA-NSDE algorithm, we divided the population into different local areas based on K-Means clustering, and then dynamically select the local search strategy or global search strategy based on their success rate to achieve dynamic resource allocation. This solves the problem of difficult selection of target vector in traditional NSDE algorithm, improves the diversity of the algorithm, and enables the algorithm to jump out of the local optimal region.

Algorithm 4 CDE-Stacking

```
Require: Non-dominate solution set $NDSet$; the number of clusters $K_e$; the number of solutions selected from each cluster $n$; maximum generation $G_e$; population size $N_e$
Ensure: Ensemble model $EM$
1: $\mathbb{S}$ ← Cluster $NDSet$ into $K_e$ classes with K-Means algorithm and obtain $K_e$ solution sets;
2: $\mathbb{L}$ ← Select $n$ solutions from each solution set in $\mathbb{S}$ and obtain $n * K_e$ solutions in total;
3: Set current generation $g_e = 0$;
4: while $g_e < G_e$ do
5:     $W_{r_1}, W_{r_2}, W_{r_3}$ ← Randomly select three solutions from the current population $P_e$;
6:     $v_i$ ← Perform Mutation operator on the current solution $W_i$ and $W_{r_1}, W_{r_2}, W_{r_3}$ according to DE/rand/1/bin;
7:     $u_i$ ← Perform Crossover operator on the current solution $W_i$ and $v_i$ according to DE/rand/1/bin;
8:     $W_i$ ← Evaluate the fitness of $u_i$ and replace $W_i$ if $u_i$ is better than $W_i$;
9:     $g_e \leftarrow g_e + 1$;
10: end while
11: $EM$ ← Get the global optimal solution $W^*$ and decode it into ensemble weights, then build the optimal ensemble model with base learners in $\mathbb{L}$ using stacking method.
```

CDE-stacking

After the non-dominated solution set is obtained in the first stage, we need to further ensemble these solutions to construct the ensemble model for silicon content prediction. In the ensemble learning, the key to achieve successful ensemble effectiveness lies in the accuracy and diversity of selected base learners that have potential contradictory relationship [4]. In the stacking-based ensemble learning, the ensemble weights also have significant influence on the ensemble effectiveness. The simple average method assigns the same ensemble weight to each base learner, which ignores the differences between base learners. Therefore, a clustering and differential evolution-based stacking ensemble (CDE-Stacking) method is proposed in this paper. Based on the K-Means clustering, the proposed CDE-Stacking first selects a series of “good and different” base learners, i.e., a series of SVRs with high accuracy and strong diversity from the non-dominated solution set obtained in the first stage. Then a single-objective differential evolution algorithm is used to optimize the ensemble weights of the selected base learners, based on which the selected base learners are integrated to construct the ensemble model for silicon content prediction. The details of the proposed CDE-Stacking algorithm are shown in Algorithm 4.

In Algorithm 4, there are two key hyperparameters, i.e., the number of clusters K_e and the number of non-dominated solutions n selected from each cluster. Figure 4 shows the schematic of the proposed CDE-Stacking when $K_e = 5, n = 4$. Once the base learners are randomly selected from each cluster, the output of the ensemble model can be obtained by computing the weight-sum of all base learners’ outputs. The weights used above are optimized by a single-objective

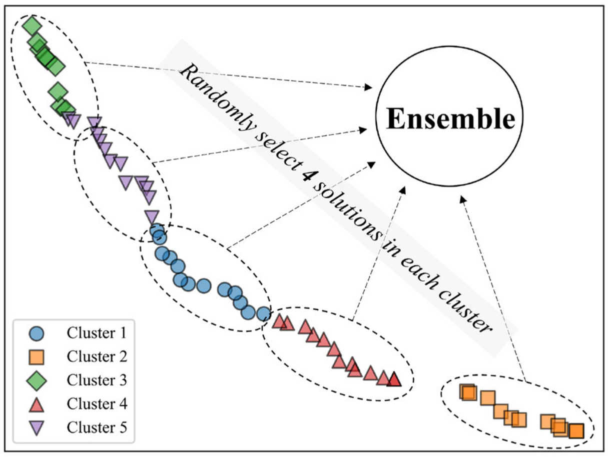


Fig. 4 Schematic of CDE-Stacking ($K_e = 5, n = 4$)

DE algorithm, where each solution W in the population $\{W_1, W_2, \ldots, W_{N_e}\}$ is a real encoded vector and each gene represents a base learner's ensemble weight. The DE algorithm aims to minimize the prediction error of the ensemble model, which is measured by the RMSE metric in Eq. (4). The evolution process is based on the standard DE/rand/1/bin. After the evolution, the global best solution W^* is obtained and decoded to integrate all the selected base learners in $\mathbb{L}$ to construct the final ensemble model for silicon content prediction. In Fig. 4, our proposed CDE-Stacking method selects solutions at different locations on the Pareto front by clustering the population into sub-populations, ensuring variability in the combinations of accuracy and complexity among the selected base learners. In general, the reduction of the complexity of the SVR base learner makes it easier to improve the generalization ability. Therefore, using the proposed CDE-Stacking method to construct the ensemble model can achieve a balance between accuracy and generalization ability.

Experiments

In this section, the proposed two-stage MOEEL algorithm with two main contributions, namely DRA-NSDE and CDE-Stacking, were tested through a series of experiments. The effectiveness of DRA-NSDE and CDE-Stacking was initially verified based on 14 multi-objective benchmark problems and a UCI benchmark dataset, respectively. Subsequently, the proposed two-stage MOEEL algorithm was evaluated based on both the UCI benchmark dataset and the actual blast furnace dataset, and compared with several other powerful machine learning algorithms. All the experiments were independently conducted multiple times to eliminate the effects of randomness. To determine the statistical significance, the Wilcoxon's rank sum test with a significance level of 0.05 was used to compare our proposed algorithm with the other rivals. The symbols "(+)" and "(-)" indicate whether our proposed algorithm is significantly better or worse than a rival, while "(=)" indicates that there is no statistical significant difference between them.

Validation of DRA-NSDE

To verify the effectiveness of the proposed DRA-NSDE algorithm in the first stage, 14 multiobjective benchmark test problems were used, including ZDT1-ZDT4, ZDT6 [44], UF1, UF2 [45] and DTLZ1-DTLZ7 [46]. The diversity and convergence of the approximate PFs obtained by the multiobjective evolutionary algorithms are two important factors to measure the algorithm performance [21]. Due to the ability to both measure the convergence and the diversity of the algorithm, the inverted generational distance (IGD) [47] and the hypervolumne (HV) [48] are used as performance evaluation indicators. The IGD metric is defined as follows:

$$IGD = \frac{\sum_{j=1}^{|P^*|} d_{min}(x_j^*, P)}{|P^*|} \tag{12}$$

$$d_{min}(x_j^*, P) = \min\left\{\sqrt{\sum_{k=1}^{m}(f_k(x_i) - f_k(x_j^*))^2},\ i = 1, 2, \ldots, |P|\right\} \tag{13}$$

where P is the non-dominated solution set obtained by the algorithm and P^* is the real Pareto optimal set. $|P|$ and $|P^*|$ represent the number of solutions in the solution set, respectively. x_i is the ith solution in $|P|$, and x_j^* denotes the jth solution in $|P^*|$. m is the number of objectives, and $f_k(x)$ represents the kth objective value of solution x. Given a reference point $r^* = (r_1^*, r_2^*, \ldots, r_m^*)$, the HV metric is defined as

$$HV(S) = VOL\left(\bigcup_{x \in P}[f_1(x), r_1^*] \times \cdots \times [f_m(x), r_m^*]\right) \tag{14}$$

where $VOL(\cdot)$ is the Lebesgue measure. A lager HV metric indicates a better algorithm performance.

Sensitivity test of parameter K

First, the sensitivity of the core parameter, i.e., parameter K of the K-Means clustering, is analyzed. Four different values of K, i.e., $\{2, 3, 4, 5\}$, are tested on the 14 benchmark problems. The population size is set to 100, and the maximum generation is set to 300 on the ZDT problems and 500 on the UF and DTLZ problems. For each problem, 30 independent experiments are conducted and the statistical results (mean and standard deviation) of IGD are recorded. This experimental setting is also adopted in the following experiments. Note that the standard deviation results are shown in round brackets, and the algorithm with the best performance in italic value. The experimental results are shown in Table 2.

According to Table 2, it is clear that the proposed DRA-NSDE algorithm with $K = 3$ outperforms the other settings for most test problems. The mean value of IGD in 30 independent experiments is visualized in Fig. 5. Intuitively, DRA-NSDE with $K = 3$ achieves lower IGD values for most problems. It should be noted that although DRA-NSDE with $K = 3$ does not achieve the best IGD values in some test problems such as ZDT1 and ZDT6, the difference between this setting and the best-performing setting is not significant. A more in-depth analysis suggests that using a too large cluster number (i.e., $K = 4$ or $K = 5$) will lead to insufficient non-dominated solutions assigned to each sub-population. As a result, the quality of the offspring solutions generated by the local search strategy may deteriorate, which will slow down the convergence speed within the sub-population and ultimately affect the overall performance of the algorithm. Conversely, if the cluster number is too small (i.e., $K = 2$), the clustering-based dynamic resource allocation mechanism cannot be fully utilized. Therefore, based on these results, we use $K = 3$ as the default hyperparameter setting for the following experiments.

Comparison with other DE strategies

In this section, we will analyze the performance of the DRA-NSDE algorithm with different mutation strategies, namely DE/rand/1/bin, DE/current-to-rand/bin, DE/best/1/bin, and DE/current-to-best/bin. The first two strategies focus on improving the algorithm diversity, while the last two strategies prioritize the algorithm convergence. The comparison results are presented in Tables 3 and 4.

According to Table 3, it is evident that DE/rand/1/bin outperforms the other three DE strategies for both IGD and HV metrics for most test problems. Specifically, when compared to DE/best/1/bin and DE/current-to-best/bin, DE/rand/1/bin achieves the best IGD and HV values for all ZDT and UF problems. The main reason can be analyzed as follows. The clustering and dynamic resource allocation can help to ensure good convergence of the NSDE algorithm. Since both DE/best/1/bin and DE/current-to-best/bin make full use of the best solution in the population, which further helps to improve the convergence, their interaction may lead to early convergence and lack of diversity. On the contrary, DE/rand/1/bin employs three randomly selected solutions to generate the mutant vector instead of the target vector, which can improve diversity. Therefore, the cooperation of the good convergence of the clustering and dynamic resource allocation and the good diversity of DE/rand/1/bin can achieve better balance of exploration and exploitation. However, DE/rand/1/bin is not the most effective strategy for DTLZ2, DTLZ4, and DTLZ5, indicating that no single strategy can perform optimally for all problems. Table 4 shows the statistical analysis results of DE/rand/1/bin and other DE strategies on the Wilcoxon's rank sum test of the IGD and HV metrics. The results indicate that DE/rand/1/bin significantly outperforms the other three DE strategies on most test problems. Therefore, we choose DE/rand/1/bin as the DE strategy for our proposed DRA-NSDE.

Table 6 Wilcoxon's rank sum test analysis of DRA-NSDE with rivals in terms of IGD and HV metrics on 14 benchmark test problems

DRA-NSDE *v.s.*	NSGA-II	NSDE	MO-ES	MO-CMA-ES
IGD				
+	13/14	11/14	14/14	13/14
=	0/14	1/14	0/14	0/14
−	1/14	2/14	0/14	1/14
HV				
+	12/14	12/14	14/14	13/14
=	0/14	1/14	0/14	0/14
−	2/14	1/14	0/14	1/14

Comparison with other MOEAs

In this section, we compare the performance of the proposed DRA-NSDE with four other powerful MOEAs: NSGA-II [42], NSDE [49], MO-ES [50] and MO-CMA-ES [51]. NSGA-II is a popular multiobjective optimization algorithm proposed by Kalyanmoy Deb in 2002. NSDE is a DE-based algorithm that uses non-dominated sorting to generate the next generation population. MO-ES is an evolutionary strategy designed for solving multiobjective optimization problems. MO-CMA-ES is the multiobjective version of CMA-ES - a highly effective stochastic optimization algorithm that adaptively estimates and updates the covariance matrix of the search distribution to efficiently find a global optimum in continuous and high-dimensional problem spaces. All algorithms were tested on 14 benchmark problems across 30 independent experiments, with the parameter settings remaining consistent with the previous section.

The comparison results are presented in Tables 5 and 6. According to the results, it can be found that the proposed DRA-NSDE outperforms the other four rival algorithms in terms of both IGD and HV metrics on 11 out of 14 test problems, which demonstrates the superiority of DRA-NSDE. It is worth noting that DRA-NSDE is unable to achieve the best results on DTLZ3 and DTLZ4, whereas traditional algorithms such as NSGA-II and NSDE perform better in the

Table 2 Sensitive test on parameter K in the proposed DRA-NSDE

Problem	DRA-NSDE (K=2)	DRA-NSDE (K=3)	DRA-NSDE (K=4)	DRA-NSDE (K=5)
ZDT1	*4.47e−03 (1.95e−04)*	4.52e−03 (1.82e−04)	4.51e−03 (1.66e−04)	4.51e−03 (2.05e−04)
ZDT2	4.59e−03 (1.85e−04)	*4.59e−03 (1.60e−04)*	4.61e−03 (1.98e−04)	4.59e−03 (1.65e−04)
ZDT3	5.11e−03 (1.51e−04)	*5.06e−03 (1.57e−04)*	5.12e−03 (1.18e−04)	5.12e−03 (1.52e−04)
ZDT4	7.55e−02 (1.35e−01)	6.36e−02 (1.19e−01)	*5.39e−02 (8.39e−02)*	1.17e−01 (1.36e−01)
ZDT6	4.58e−03 (2.22e−04)	4.57e−03 (2.53e−04)	*4.57e−03 (2.42e−04)*	4.65e−03 (2.30e−04)
DTLZ1	2.68e−02 (9.29e−04)	*2.67e−02 (8.39e−04)*	3.55e−02 (5.04e−02)	3.58e−02 (5.04e−02)
DTLZ2	6.74e−02 (2.39e−03)	*6.66e−02 (2.27e−03)*	6.72e−02 (2.22e−03)	6.69e−02 (2.23e−03)
DTLZ3	1.63e−01 (2.93e−01)	*9.97e−02 (1.75e−01)*	3.30e−01 (9.71e−01)	4.87e−01 (7.58e−01)
DTLZ4	6.72e−02 (2.14e−03)	6.73e−02 (1.96e−03)	6.69e−02 (2.71e−03)	*6.67e−02 (2.06e−03)*
DTLZ5	5.54e−03 (3.30e−04)	*5.51e−03 (2.72e−04)*	5.51e−03 (2.25e−04)	5.60e−03 (2.72e−04)
DTLZ6	6.11e−03 (3.44e−04)	6.03e−03 (2.54e−04)	*6.01e−03 (4.12e−04)*	6.06e−03 (4.06e−04)
DTLZ7	*7.30e−02 (4.48e−03)*	7.42e−02 (4.13e−03)	7.51e−02 (4.32e−03)	7.54e−02 (4.01e−03)
UF1	5.98e−02 (1.77e−02)	*4.60e−02 (1.49e−02)*	4.97e−02 (2.19e−02)	4.69e−02 (1.83e−02)
UF2	3.80e−02 (4.92e−03)	3.58e−02 (5.65e−03)	*3.32e−02 (7.20e−03)*	3.60e−02 (1.20e−02)

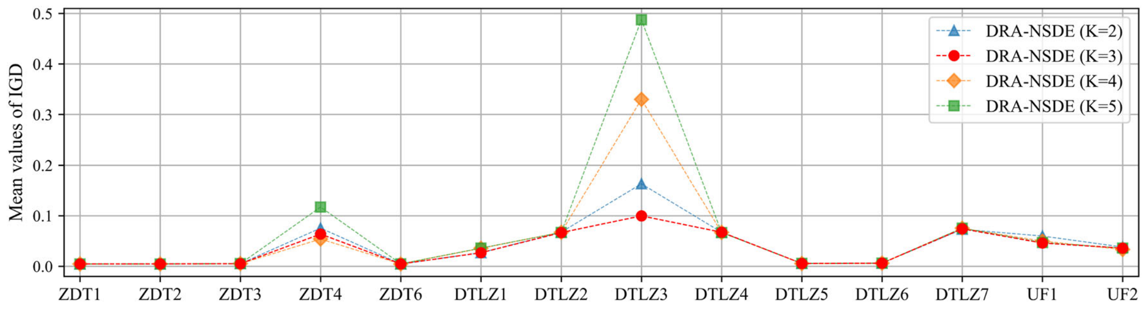


Fig. 5 Sensitive test on parameter K in the proposed DRA-NSDE

two problems due to their powerful global search abilities. In addition, MO-CMA-ES gives better result than our DRA-NSDE on ZDT6, which suggests that MO-CMA-ES may be more suitable for non-convex problems.

Figure 6 illustrates the Pareto fronts obtained by the five algorithms on three typical problems: ZDT4, UF1, and DTLZ2. The result indicates that the proposed DRA-NSDE algorithm not only obtains a Pareto front closer to the true Pareto front, but also maintains good diversity. In the case of ZDT4, NSDE, MO-ES, and MO-CMA-ES failed to converge to the true Pareto front, and NSGA-II lost the diversity during the evolution process. On the contrary, the Pareto front obtained by the proposed DRA-NSDE algorithm almost entirely covered the true Pareto front. The main reason is that the proposed dynamic resource allocation mechanism adaptively adjusts the ratio of local and global search in each generation based on their success rates, so that a good balance between exploration and exploitation can be achieved. To give an insight to the proposed dynamic resource allocation mechanism, we further illustrate the variation curves of local search ratio and global search ratio when solving the above three typical problems. As shown in Fig. 7, in the early stage of the evolution, the local search ratio is higher than the global search ratio because at this stage the population is relatively dispersed and thus the algorithm focuses more on the convergence speed (more local search can accelerate the convergence of the algorithm). As the evolution proceeds, the global search ratio gradually exceeds the local search ratio because the population may lose diversity during evolution, and consequently the algorithm turns to focuses more on maintaining good diversity by increasing the global search ratio. The dynamic allocation mechanism allows the algorithm to flexibly adopt different search strategies at different stages of the evolution, thus enabling the algorithm to obtain a balance between exploration and exploitation.

From the above experimental results and analysis, we can draw the conclusion that the proposed DRA-NSDE method can achieve superior or competitive results for most of test problems. This ensures the stability of DRA-NSDE in generating a range of high-quality solutions to participate in the next stage and construct an ensemble model which is expected to achieve promising prediction performance.

Table 3 Comparison of IGD and HV between different DE strategies

Metric	Problem	DE/rand/1/bin	DE/best/1/bin	DE/current-to-best/bin	DE/current-to-rand/bin
IGD	ZDT1	*4.53e−03 (1.62e−04)*	4.53e−03 (1.99e−04) (=)	6.16e−02 (2.47e−02) (+)	7.38e−02 (1.09e−02) (+)
	ZDT2	*4.62e−03 (1.30e−04)*	4.86e−01 (2.42e−01) (+)	5.72e+00 (2.08e+00) (+)	4.67e+00 (1.42e+00) (+)
	ZDT3	*5.15e−03 (1.23e−04)*	5.34e−03 (2.09e−04) (+)	5.79e−02 (1.69e−02) (+)	9.88e−02 (1.52e−02) (+)
	ZDT4	*8.59e−03 (2.21e−02)*	2.35e+00 (1.19e+00) (+)	3.99e+00 (1.01e+00) (+)	4.70e+00 (1.30e+00) (+)
	ZDT6	*4.73e−03 (2.47e−04)*	5.02e−03 (2.47e−04) (+)	5.81e−02 (1.04e−01) (+)	9.43e−02 (1.51e−01) (+)
	DTLZ1	*2.73e−02 (8.06e−04)*	5.45e−02 (8.49e−02) (+)	7.97e−02 (1.29e−01) (+)	6.53e−02 (6.27e−02) (+)
	DTLZ2	6.87e−02 (2.26e−03)	6.56e−02 (2.40e−03) (−)	*6.45e−02 (1.94e−03) (−)*	6.47e−02 (2.48e−03) (−)
	DTLZ3	*1.03e−01 (1.76e−01)*	4.63e+00 (7.51e+00) (+)	2.70e+00 (2.03e+00) (+)	1.76e+00 (1.40e+00) (+)
	DTLZ4	6.82e−02 (2.40e−03)	*6.76e−02 (3.75e−03) (−)*	2.07e−01 (2.46e−01) (+)	1.23e−01 (1.50e−01) (+)
	DTLZ5	5.66e−03 (3.30e−04)	6.23e−03 (5.39e−04) (+)	5.70e−03 (3.73e−04) (=)	*5.47e−03 (2.99e−04) (−)*
	DTLZ6	*5.97e−03 (3.30e−04)*	6.84e−03 (7.49e−04) (+)	2.56e−01 (4.81e−01) (+)	6.00e−03 (3.20e−04) (=)
	DTLZ7	*7.61e−02 (5.51e−03)*	8.24e−02 (5.31e−02) (+)	1.96e−01 (7.28e−02) (+)	2.67e−01 (1.13e−01) (+)
	UF1	*7.53e−02 (2.23e−02)*	9.36e−02 (1.29e−02) (+)	9.28e−02 (1.37e−02) (+)	9.06e−02 (1.91e−02) (+)
	UF2	*4.28e−02 (4.70e−03)*	6.21e−02 (4.82e−03) (+)	5.59e−02 (1.21e−02) (+)	5.38e−02 (1.43e−02) (+)
HV	ZDT1	*6.61e−01 (1.90e−04)*	6.61e−01 (2.39e−04) (+)	5.76e−01 (3.31e−02) (+)	5.58e−01 (1.47e−02) (+)
	ZDT2	*3.28e−01 (1.62e−04)*	6.54e−02 (1.28e−01) (+)	0.00e+00 (0.00e+00) (+)	0.00e+00 (0.00e+00) (+)
	ZDT3	*7.79e−01 (7.39e−05)*	7.78e−01 (5.92e−04) (+)	6.72e−01 (2.61e−02) (+)	6.08e−01 (2.55e−02) (+)
	ZDT4	*6.55e−01 (3.04e−02)*	1.96e−02 (7.14e−02) (+)	0.00e+00 (0.00e+00) (+)	0.00e+00 (0.00e+00) (+)
	ZDT6	*2.64e−01 (2.35e−04)*	2.64e−01 (2.62e−04) (+)	2.26e−01 (5.89e−02) (+)	2.03e−01 (8.93e−02) (+)
	DTLZ1	*9.51e−02 (4.45e−04)*	8.71e−02 (2.64e−02) (+)	8.03e−02 (2.50e−02) (+)	7.86e−02 (2.13e−02) (+)
	DTLZ2	3.68e−01 (6.98e−03)	3.82e−01 (5.14e−03) (−)	3.84e−01 (6.35e−03) (−)	*3.85e−01 (6.05e−03) (−)*
	DTLZ3	*3.48e−01 (6.71e−02)*	1.43e−01 (1.91e−01) (+)	2.13e−02 (7.20e−02) (+)	5.60e−02 (1.24e−01) (+)
	DTLZ4	3.69e−01 (5.10e−03)	*3.85e−01 (7.66e−03) (−)*	3.33e−01 (9.89e−02) (+)	3.57e−01 (5.89e−02) (+)
	DTLZ5	4.62e−02 (1.09e−04)	4.60e−02 (9.85e−05) (+)	4.63e−02 (8.80e−05) (−)	*4.64e−02 (8.58e−05) (−)*
	DTLZ6	*4.63e−02 (8.46e−05)*	4.62e−02 (1.00e−04) (+)	3.51e−02 (1.98e−02) (+)	4.63e−02 (9.46e−05) (+)
	DTLZ7	*7.30e−01 (7.86e−03)*	7.08e−01 (1.49e−02) (+)	4.19e−01 (5.36e−02) (+)	3.65e−01 (5.76e−02) (+)
	UF1	*5.60e−01 (2.83e−02)*	5.05e−01 (2.71e−02) (+)	5.10e−01 (2.27e−02) (+)	5.27e−01 (2.18e−02) (+)
	UF2	*6.15e−01 (4.21e−03)*	5.82e−01 (4.99e−03) (+)	5.97e−01 (5.97e−03) (+)	6.04e−01 (6.27e−03) (+)

Table 4 Wilcoxon's rank sum test analysis of different DE strategies in terms of IGD and HV metrics on 14 benchmark test problems

DE/rand/1/bin *v.s.*	DE/best/1/bin	DE/current-to-best/bin	DE/current-to-rand/bin
IGD			
+	11/14	12/14	11/14
=	1/14	1/14	1/14
−	2/14	1/14	2/14
HV			
+	12/14	12/14	12/14
=	0/14	0/14	0/14
−	2/14	2/14	2/14

Validation of CDE-stacking

In order to verify the effectiveness of the proposed CDE-Stacking method in selecting non-dominated solutions and constructing the ensemble model in the second stage, CDE-Stacking is compared with its DE-free version (denoted as C-Stacking) and clustering-free version (denoted as DE-Stacking) after the same evolutionary process in the first stage. Please note that all the three versions have the same non-dominated solution set obtained by the proposed DRA-NSDE algorithm, and thus the convincing and valid results can be obtained. In C-Stacking, instead of optimizing the ensemble weights by DE, the ensemble weight of each solution is set to be equal to $\frac{1}{n*K_e}$, where $n * K_e$ represents

Table 5 Comparison of IGD and HV between DRA-NSDE and rivals

Metric	Problem	NSGA-II	NSDE	MO-ES	MO-CMA-ES	DRA-NSDE
IGD	ZDT1	8.14e−03 (8.90e−04) (+)	5.16e−03 (2.26e−04) (+)	6.53e−02 (4.69e−03) (+)	5.77e−03 (3.49e−04) (+)	*4.51e−03 (1.27e−04)*
	ZDT2	3.04e−01 (1.12e−01) (+)	7.34e−03 (6.68e−04) (+)	2.58e+00 (2.93e−01) (+)	5.95e−03 (3.45e−04) (+)	*4.66e−03 (1.81e−04)*
	ZDT3	1.06e−02 (1.96e−02) (+)	5.80e−03 (2.66e−04) (+)	1.39e−01 (7.22e−02) (+)	7.07e−03 (4.76e−04) (+)	*5.14e−03 (1.98e−04)*
	ZDT4	1.26e−01 (8.00e−02) (+)	5.75e+00 (1.30e+00) (+)	2.21e+01 (4.05e+00) (+)	9.62e+00 (4.07e+00) (+)	*9.84e−02 (2.02e−01)*
	ZDT6	2.10e−02 (8.78e−03) (+)	4.77e−03 (2.15e−04) (+)	6.79e−02 (3.98e−02) (+)	*3.91e−03 (2.19e−04) (−)*	4.66e−03 (2.94e−04)
	DTLZ1	2.31e−01 (2.59e−01) (+)	2.86e−02 (1.74e−03) (+)	1.39e+01 (3.36e+00) (+)	1.01e+01 (6.29e+00) (+)	*2.74e−02 (1.19e−03)*
	DTLZ2	6.91e−02 (2.96e−03) (+)	6.86e−02 (2.63e−03) (=)	1.03e−01 (7.15e−03) (+)	1.19e−01 (6.22e−03) (+)	*6.84e−02 (2.08e−03)*
	DTLZ3	7.48e−02 (6.39e−03) (−)	*6.79e−02 (2.69e−03) (−)*	1.73e+02 (2.20e+01) (+)	7.36e+01 (3.25e+01) (+)	1.08e−01 (1.90e−01)
	DTLZ4	1.94e−01 (2.13e−01) (+)	*6.83e−02 (2.19e−03) (−)*	2.30e−01 (1.51e−01) (+)	1.86e−01 (8.30e−02) (+)	6.95e−02 (2.19e−03)
	DTLZ5	5.98e−03 (2.19e−04) (+)	5.90e−03 (3.48e−04) (+)	1.36e−02 (1.61e−03) (+)	1.35e−02 (2.65e−03) (+)	*5.65e−03 (2.94e−04)*
	DTLZ6	6.15e−03 (4.09e−04) (+)	6.16e−03 (3.78e−04) (+)	3.72e+00 (6.10e−01) (+)	4.95e−02 (1.04e−02) (+)	*6.08e−03 (4.02e−04)*
	DTLZ7	8.46e−02 (5.96e−03) (+)	7.76e−02 (4.39e−03) (+)	3.60e−01 (1.58e−01) (+)	1.23e−01 (1.84e−02) (+)	*7.51e−02 (3.44e−03)*
	UF1	1.30e−01 (3.02e−02) (+)	9.55e−02 (7.44e−03) (+)	1.20e−01 (1.84e−02) (+)	2.06e−01 (4.35e−02) (+)	*8.06e−02 (1.68e−02)*
	UF2	6.12e−02 (1.14e−02) (+)	7.49e−02 (5.19e−03) (+)	5.49e−02 (8.89e−03) (+)	8.52e−02 (1.26e−02) (+)	*4.66e−02 (1.33e−02)*
HV	ZDT1	6.53e−01 (1.53e−03) (+)	6.58e−01 (4.53e−04) (+)	5.69e−01 (6.30e−03) (+)	6.59e−01 (6.54e−04) (+)	*6.61e−01 (1.76e−04)*
	ZDT2	9.84e−02 (7.08e−02) (+)	3.21e−01 (9.68e−04) (+)	0.00e+00 (0.00e+00) (+)	3.26e−01 (5.22e−04) (+)	*3.27e−01 (2.63e−04)*
	ZDT3	7.66e−01 (1.41e−02) (+)	7.73e−01 (1.00e−03) (+)	5.61e−01 (6.29e−02) (+)	7.78e−01 (2.83e−04) (+)	*7.79e−01 (1.27e−04)*
	ZDT4	5.48e−01 (7.25e−02) (+)	0.00e+00 (0.00e+00) (+)	0.00e+00 (0.00e+00) (+)	0.00e+00 (0.00e+00) (+)	*5.55e−01 (2.05e−01)*
	ZDT6	2.41e−01 (1.06e−02) (+)	2.64e−01 (2.33e−04) (+)	2.09e−01 (2.47e−02) (+)	*2.65e−01 (2.10e−04) (−)*	2.64e−01 (2.97e−04)
	DTLZ1	5.24e−02 (4.25e−02) (+)	9.45e−02 (9.32e−04) (+)	0.00e+00 (0.00e+00) (+)	0.00e+00 (0.00e+00) (+)	*9.49e−02 (4.90e−04)*
	DTLZ2	*3.74e−01 (6.05e−03) (−)*	3.65e−01 (5.51e−03) (+)	3.05e−01 (9.97e−03) (+)	2.59e−01 (1.05e−02) (+)	3.70e−01 (5.81e−03)
	DTLZ3	3.55e−01 (1.53e−02) (−)	*3.67e−01 (5.64e−03) (−)*	0.00e+00 (0.00e+00) (+)	0.00e+00 (0.00e+00) (+)	3.46e−01 (7.14e−02)
	DTLZ4	3.34e−01 (7.68e−02) (+)	3.68e−01 (5.17e−03) (=)	3.07e−01 (5.11e−02) (+)	2.80e−01 (5.31e−02) (+)	*3.68e−01 (4.17e−03)*
	DTLZ5	4.59e−02 (1.13e−04) (+)	4.59e−02 (1.50e−04) (+)	4.32e−02 (2.76e−04) (+)	4.35e−02 (3.73e−04) (+)	*4.61e−02 (1.12e−04)*
	DTLZ6	4.63e−02 (7.56e−05) (+)	4.63e−02 (1.07e−04) (+)	0.00e+00 (0.00e+00) (+)	3.52e−02 (2.35e−03) (+)	*4.63e−02 (9.07e−05)*
	DTLZ7	6.54e−01 (1.71e−02) (+)	6.92e−01 (1.21e−02) (+)	3.33e−01 (4.25e−02) (+)	6.49e−01 (2.24e−02) (+)	*7.25e−01 (8.03e−03)*
	UF1	4.84e−01 (4.44e−02) (+)	5.09e−01 (1.38e−02) (+)	4.63e−01 (3.16e−02) (+)	3.21e−01 (6.94e−02) (+)	*5.47e−01 (2.30e−02)*
	UF2	5.91e−01 (6.51e−03) (+)	5.63e−01 (7.31e−03) (+)	5.91e−01 (1.32e−02) (+)	5.46e−01 (1.72e−02) (+)	*6.10e−01 (5.63e−03)*

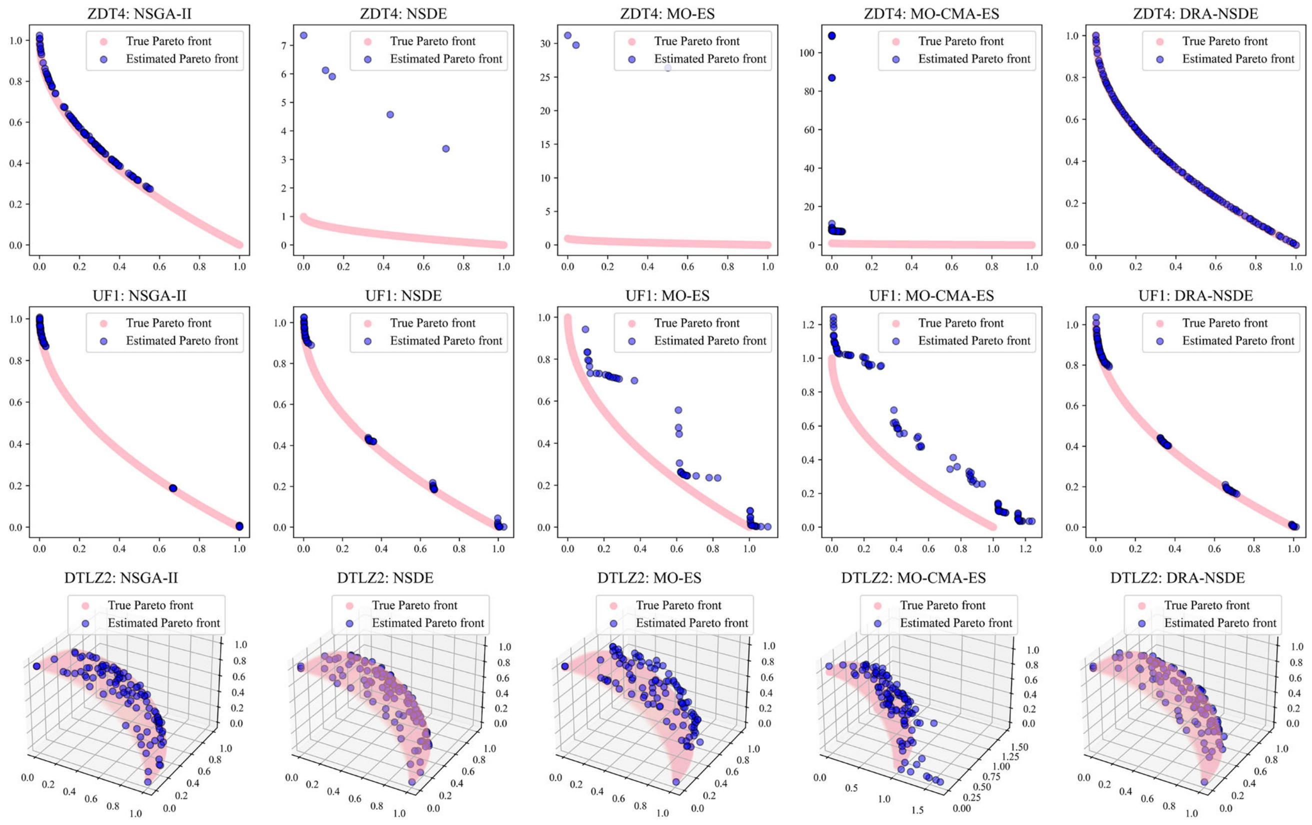


Fig. 6 Comparison of Pareto front obtained by different algorithms

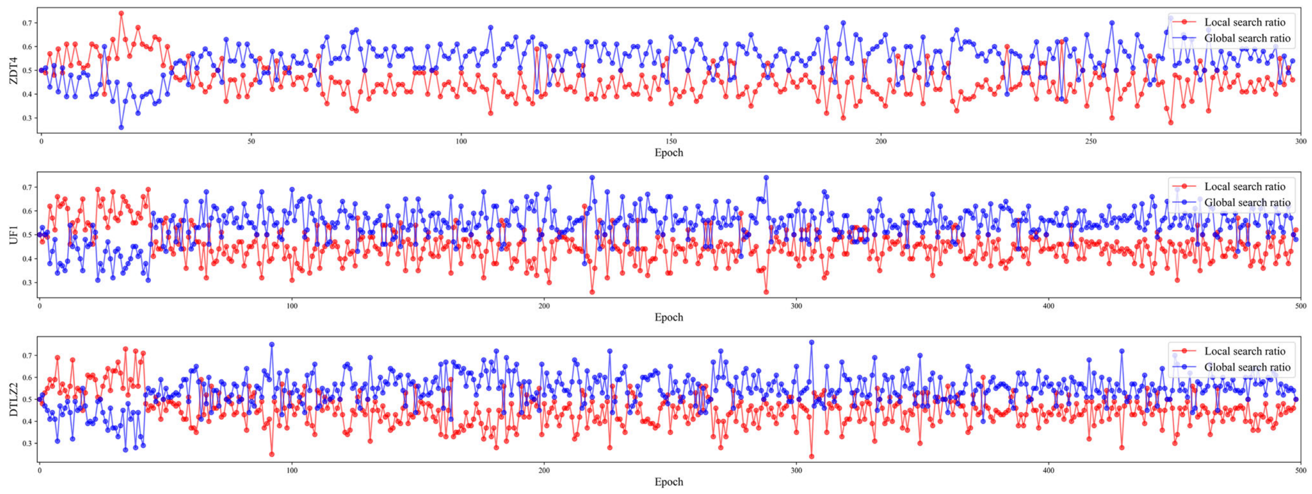


Fig. 7 The variation curves of local search ratio and global search ratio

the total number of selected solutions. In DE-Stacking, the solution selection process based on K-Means clustering is removed, and all solutions are selected in the non-dominated set whose ensemble weights are optimized by DE to construct the ensemble model.

The data set used in this subsection, i.e., SkillCraft1 Master Table Dataset [52], is obtained from the UCI Machine Learning Repository. This data set contains 3395 samples and each sample has 20 original attributes. After discarding the first attribute GameID and handling the missing values, a set of 3338 samples with 18 input features and 1 predictable feature (LeagueIndex) is obtained. Then, the obtained data set is normalized according to the following method:

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \tag{15}$$

Table 7 Comparison between different solution ensemble methods

	RMSE		R^2	
	Mean	Std	Mean	Std
C-Stacking	9.59e−01 (+)	6.87e−03	5.16e−01 (+)	6.94e−03
DE-Stacking	9.49e−01 (+)	6.35e−03	5.26e−01 (+)	6.35e−03
CDE-Stacking	*9.37e−01*	*2.61e−03*	*5.38e−01*	*2.57e−03*

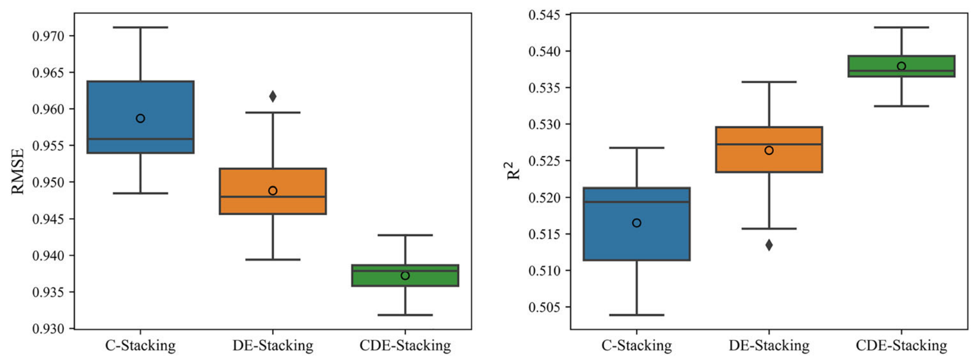


Fig. 8 Box plots of different solution ensemble methods

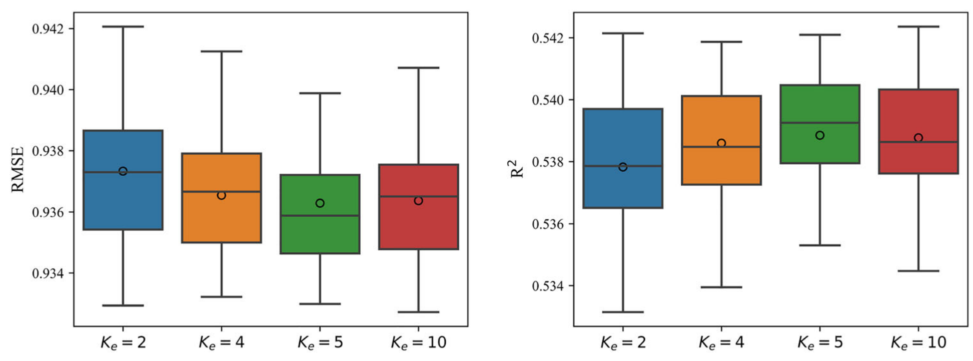


Fig. 9 Sensitive test on parameter K_e in the proposed CDE-Stacking

In the prediction problems, RMSE and R^2 are two commonly used metrics to evaluate the algorithm performance. RMSE is the root mean square error that can evaluate the model prediction accuracy as shown in Eq. (4), while R^2 is the model goodness of fit which is defined as follows:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{L}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{L}(y_i - \bar{y}_i)^2} \tag{16}$$

where $\hat{y}_i$ is the predicted value for the ith sample, y_i is the corresponding true value and $\bar{y}_i$ is the average value of all true value y_i.

In the following experiments, three versions of ensemble method, i.e., C-Stacking, DE-Stacking, and CDE-Stacking, are tested on the preprocessed SkillCraft1 data set. The population size in DE-Stacking and CDE-Stacking is set to 100, and their maximum number of generations is set to 500. The number of clusters K_e is set to 5, and the number of solutions selected from each cluster n is set to 4. All results are obtained through 15 independent experiments.

The computational results are shown in Table 7 and Fig. 8. It can be seen that CDE-Stacking achieved the best and the most stable results in both RMSE and R^2 among all the three methods, which means that CDE-Stacking is significantly better and more robust than C-Stacking and DE-Stacking. The clustering-based solution selection process can further improve the diversity of the ensemble model by randomly selecting solutions in sub-populations. The DE-based optimization of ensemble weights can utilize the solutions in different locations of the Pareto front, thus improving the accuracy of the ensemble model. Therefore, the above results verified the effectiveness of the proposed CDE-Stacking method.

We further analyze the sensitivity of the parameter K_e in the proposed CDE-Stacking method. The values of K_e are set as $K_e \in \{2, 4, 5, 10\}$. To ensure the total number of selected solutions is the same, i.e., 20, the parameter n (number of solutions selected from each cluster) needs to be set as $n \in \{10, 5, 4, 2\}$. The other parameters are the same as the above settings. Figure 9 gives the R^2 and RMSE results obtained by different values of K_e. It shows that the perfor-

Table 8 Parameter setting of the proposed algorithm

	Parameter	Name	Value
Stage 1	Population size	N	100
	Maximum generation	G	1000
	Number of clusters in DRA-NSDE	K	3
Stage 2	Population size	N_e	100
	Maximum generation	G_e	500
	Number of clusters in CDE-Stacking	K_e	5
	Number of solutions selected from each cluster	n	4

Table 9 Comparison between different prediction models in SkillCraft1 data set

Model	RMSE		R^2	
	Mean	Std	Mean	Std
MLP	9.48e−01 (+)	7.04e−03	5.28e−01 (+)	7.02e−03
SVR	9.66e−01 (+)	2.22e−16	5.09e−01 (+)	2.22e−16
Bagging	9.61e−01 (+)	1.22e−02	5.14e−01 (+)	1.23e−02
AdaBoost	9.60e−01 (+)	5.90e−03	5.15e−01 (+)	5.97e−03
RF	9.49e−01 (+)	8.93e−03	5.26e−01 (+)	8.97e−03
Bagging-SVR	9.67e−01 (+)	1.41e−03	5.08e−01 (+)	1.44e−03
AdaBoost-SVR	9.62e−01 (+)	1.21e−03	5.13e−01 (+)	1.22e−03
Stacking-SVR	9.67e−01 (+)	5.05e−03	5.08e−01 (+)	5.16e−03
Proposed	*9.36e−01*	2.03e−03	*5.39e−01*	2.00e−03

mance of the proposed algorithm first becomes better but then worse as the number of clusters increases. This indicates that when K_e increases, more sub-populations in different locations of the Pareto front are generated so that the diversity of the selected solution set is getting better. But when K_e becomes too large, i.e., 10, the solutions with high accuracy will be less in the selected solution set, which will in turn deteriorate the performance of the algorithm. Therefore, the setting of K_e should balance the diversity and the accuracy of the selected solution set. The sensitive results indicate that the proposed CDE-Stacking method works well when the number of clusters K_e is set to 5.

UCI benchmark test

To verify the effectiveness of the proposed two-stage MOEEL algorithm, a series of numerical experiments are conducted on the same UCI benchmark data set described in "Validation of CDE-stacking" section, i.e., the SkillCraft1 data set. For the purpose of model comparison, we introduce 8 rival prediction models including 2 single machine learning models, 3 tree-based non-linear ensemble learning models, and 3 SVR-based non-linear ensemble learning models.

(1) *MLP*: Multi-layer perceptron regressor;
(2) *SVR*: Support vector regression model;
(3) *Bagging*: Bagging regressor with decision tree as its base-learner;
(4) *AdaBoost*: AdaBoost regressor with decision tree as its base-learner;
(5) *RF*: Random forest regressor with decision tree as its base-learner;
(6) *Bagging-SVR*: Bagging regressor with SVR as its base-learner;
(7) *AdaBoost-SVR*: AdaBoost regressor with SVR as its base-learner;
(8) *Stacking-SVR*: Stacking of SVRs which uses a MLP model as its final regressor. Consistent with the proposed algorithm, the number of SVRs in Stacking-SVR is set to 20. The MLP model used to combine the SVRs' outputs is a three-layer neural network which has 20 hidden nodes in its hidden layer.
(9) *Proposed*: The proposed two-stage MOEEL algorithm with DRA-NSDE and CDE-Stacking.

The main parameters of the proposed algorithm are listed in Table 8. All the 9 models were tested on SkillCraft1 data set, and the comparison results obtained through 15 independent experiments are shown in Table 9 and Fig. 10. Based on these results, the following analysis can be drawn:

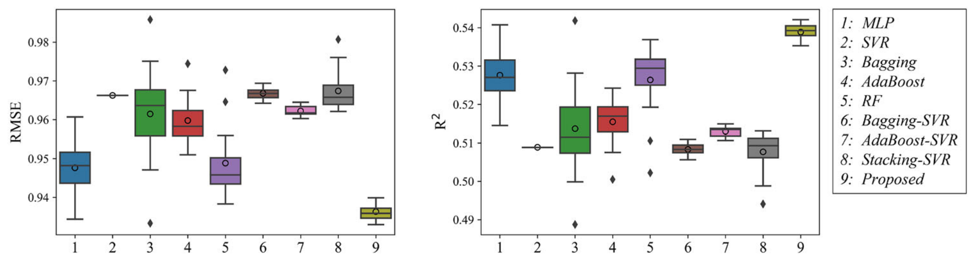


Fig. 10 Box plots of different prediction models in SkillCraft1 data set

(1) The proposed prediction model significantly outperforms the other 8 models in both RMSE and R^2, indicating that our model is more advantageous.
(2) *MLP* and three tree-based ensemble learning models (*Bagging*, *AdaBoost*, *RF*) achieve better performance than *SVR*, indicating that neural network and ensemble learning are more suitable than *SVR* in complex prediction problem because they have a strong non-linear relationship extraction ability.
(3) While the superiority has been verified in tree-based ensemble models, three SVR-based ensemble models (*Bagging-SVR*, *AdaBoost-SVR*, *Stacking-SVR*) fail to show a significant improvement compared with *SVR*. However, based on the proposed two-stage MOEEL algorithm, the SVR-based ensemble model (*Proposed*) achieves significantly better performance than all the three tree-based ensemble models. This indicates that the proper selection of base learners and the proper ensemble method are critical when constructing the ensemble model. Our proposed method can obtain a better non-dominated solution set with high accuracy and strong diversity through the dynamic resource allocation mechanism, and it can efficiently select high-diversity solutions and construct the ensemble model with appropriate ensemble weights.

Blast furnace data test

To verify the effectiveness of the proposed two-stage MOEEL algorithm for silicon content prediction, the blast furnace data set is introduced in this subsection. The data come from one major iron and steel enterprise in China and is collected from May 1, 2021 to March 14, 2022. After handling the outliers and missing values, the silicon content prediction data set is obtained. It has 843 samples, each of which contains 40 input features (blast furnace process variable) and 1 output variable (silicon content).

In the actual blast furnace production process, the experts are more interested in the hit rate (HR) of the silicon content prediction, which is defined as follows:

$$\mathrm{HR} = \frac{1}{L}\left(\sum_{i=1}^{L} H_i\right) \times 100\%$$

$$H_i = \begin{cases} 1, & |\hat{y}_i - y_i| < 0.1 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

where $\hat{y}_i$ is the predict value of silicon content for the ith sample, y_i is the true value, and L is the total number of samples. Hit rate indicates the prediction accuracy of the silicon content. Together with RMSE (Eq. (4)) and R^2 (Eq. (16)), the three indicators are used to evaluate the silicon content prediction performance of different algorithms.

To verify the effectiveness of the proposed algorithm, a new rival model called *Proposed/RF* is introduced. *Proposed/RF* is a variation version of the proposed two-stage MOEEL algorithm in which the random forest regressor is taken as the base learner instead of SVR. Due to the characteristic of randomly selecting features in random forest algorithm, all features collected in blast furnace data set are considered in the *Proposed/RF* model training process, which means the feature selection part in Fig. 2 is removed and only the hyperparameter part is utilized to optimize the parameters of RF. Other settings of *Proposed/RF* are the same as *proposed*. Including the other models described in “UCI benchmark test” section, all the 10 models are tested on blast furnace data set through 15 independent experiments, and the comparison results are presented in Table 10 and Fig. 11.

Based on the results, it can be clearly seen that our proposed model outperforms the other rivals, because the proposed model achieves the best results of R^2 and RMSE and the performance difference is significant. Though *AdaBoost* obtains the highest HR, the difference in HR results between our model and *AdaBoost* is not significant. We can also see that *Proposed/RF* achieves better prediction performance than *RF*, which indicates that the proposed two-stage MOEEL framework is effective in finding the optimal parameters of the base learner in ensemble learning. Compared with *Proposed/RF*, the performance of *Proposed* can be further improved, which indicates that SVR is a better choice than random forest to construct the ensemble model as base learner under our proposed two-stage MOEEL framework. The key reason is that the feature selection part in our designed encoding scheme is more effective than the original feature selection mechanism in random forest, leading to better diversity in our proposed two-stage MOEEL model and thus improving the prediction accuracy. For a more spe-

Table 10 Comparison between different Silicon prediction models

Model	RMSE		R^2		HR	
	Mean	Std	Mean	Std	Mean	Std
MLP	6.40e−02 (+)	2.87e−03	4.92e−01 (+)	4.53e−02	75.58% (+)	1.95e−02
SVR	6.08e−02 (+)	1.39e−17	5.43e−01 (+)	0.00e+00	79.29% (+)	0.00e+00
Bagging	6.09e−02 (+)	1.60e−03	5.41e−01 (+)	2.42e−02	78.34% (+)	1.27e−02
AdaBoost	5.99e−02 (+)	1.05e−03	5.55e−01 (+)	1.55e−02	*80.08% (=)*	1.49e−02
Bagging-SVR	6.08e−02 (+)	2.44e−04	5.42e−01 (+)	3.67e−03	79.01% (+)	6.43e−03
AdaBoost-SVR	6.04e−02 (+)	5.35e−04	5.48e−01 (+)	8.02e−03	77.40% (+)	9.95e−03
Stacking-SVR	6.03e−02 (+)	2.62e−03	5.49e−01 (+)	4.00e−02	76.77% (+)	3.13e−02
RF	5.87e−02 (+)	5.16e−04	5.74e−01 (+)	7.51e−03	79.53% (+)	1.12e−02
Proposed/RF	5.84e−02 (+)	3.13e−04	5.78e−01 (+)	4.51e−03	79.37% (+)	7.75e−03
Proposed	*5.73e−02*	1.23e−03	*5.93e−01*	1.74e−02	80.00%	1.52e−02

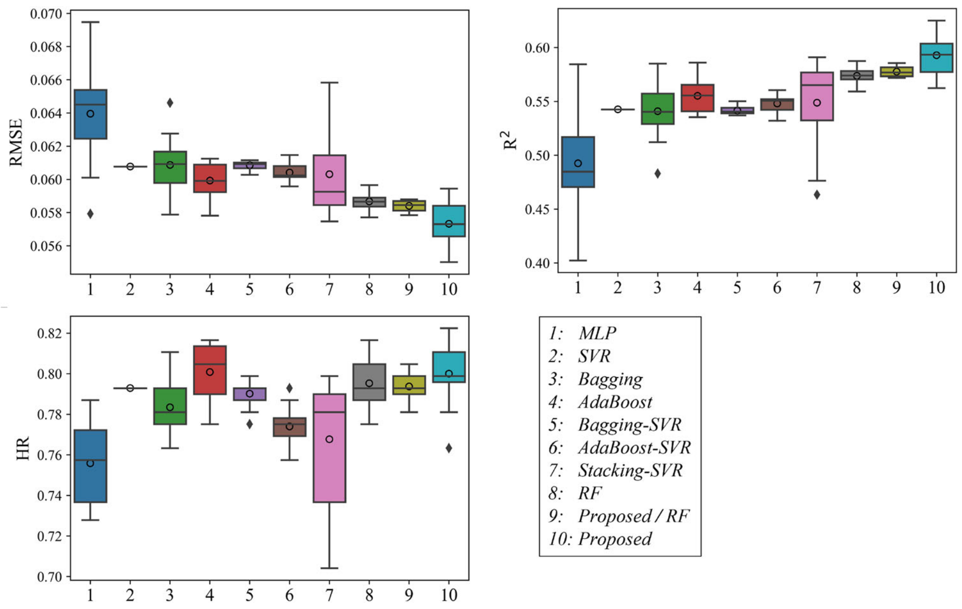


Fig. 11 Box plots of different Silicon prediction models

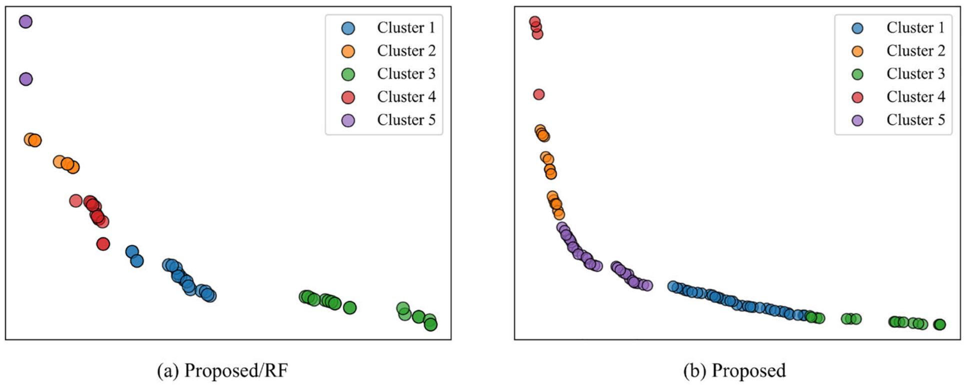
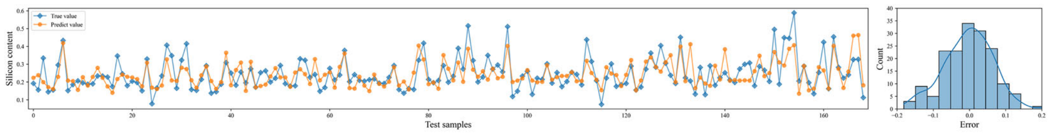
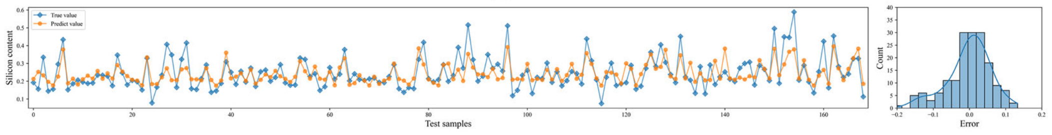


Fig. 12 Comparison of Pareto fronts between *Proposed/RF* and *Proposed*

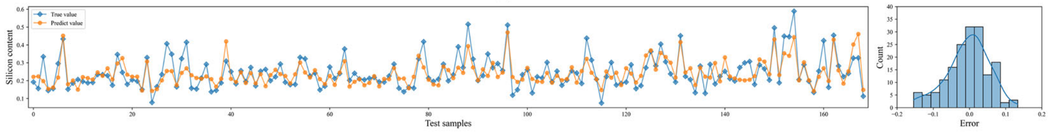
cific illustration, Fig. 12 gives a comparison of Pareto fronts between *Proposed/RF* and *Proposed* obtained from their best results in the 15 independent experiments. It can be clearly seen that the front diversity of *Proposed* is much better than that of *Proposed/RF*. Please note that the total number of non-dominated solutions is also more than that obtained by *Proposed/RF*, which is helpful in stage two when selecting solutions from different sub-populations. More specifically,



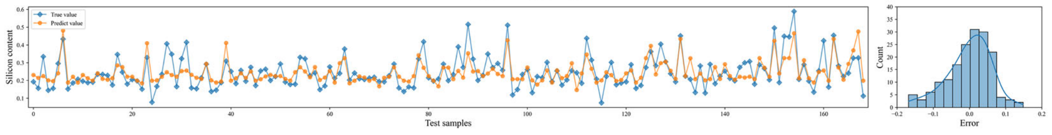
(a) *MLP*



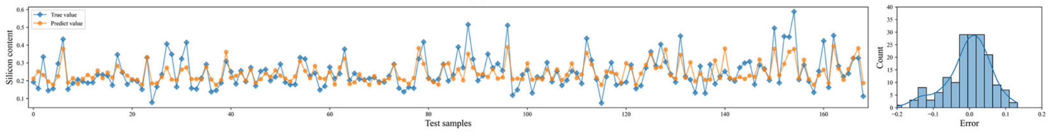
(b) *SVR*



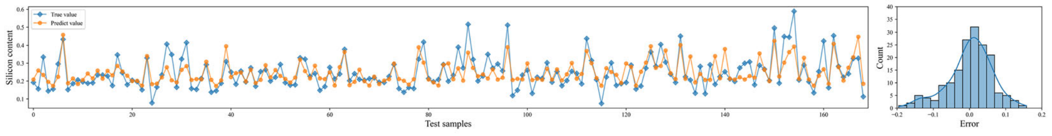
(c) *Bagging*



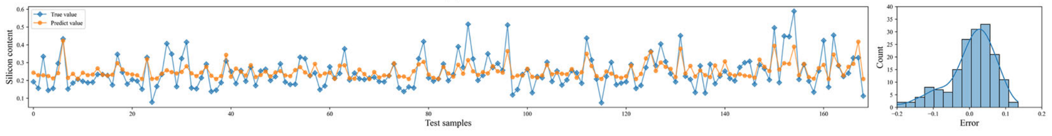
(d) *AdaBoost*



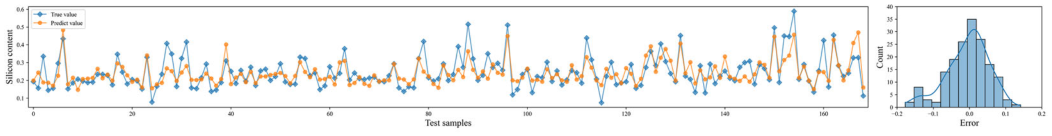
(e) *Bagging-SVR*



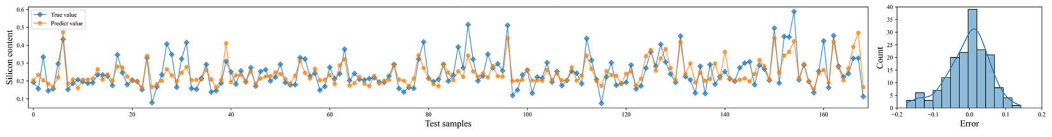
(f) *AdaBoost-SVR*



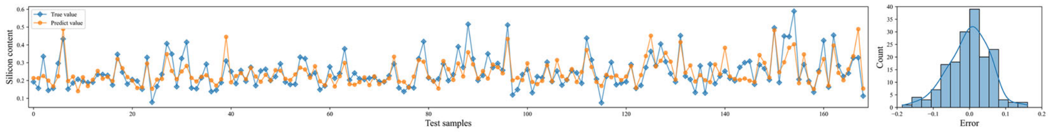
(g) *Stacking-SVR*



(h) *Random forest*



(i) *Proposed/RF*



(j) *Proposed*

Fig. 13 Prediction curves and error distribution plots of different silicon prediction models

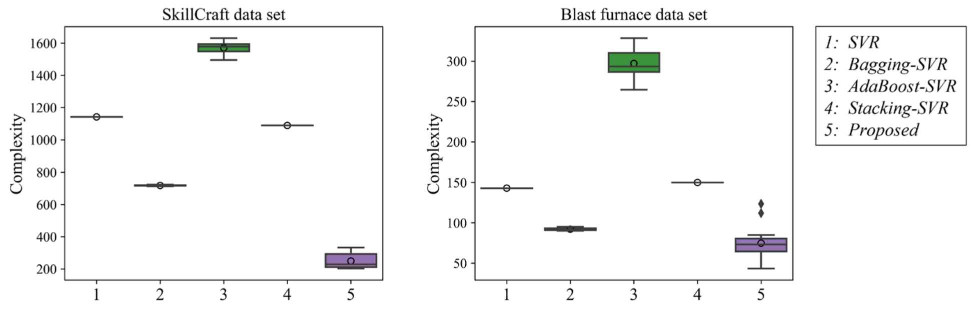


Fig. 14 Comparison of model complexity

if a sub-population contains enough numbers of different solutions, the solutions it generated will have a better diversity. At last, it is worth noting that the proposed method not only improves the prediction accuracy but also significantly saves the model training cost since the model complexity of SVR is much lower than that of random forest. Therefore, the above analysis verified the superiority of the proposed model in the silicon content prediction.

The prediction curves and its error distribution of each model on the test set are visualized in Fig. 13, from which we can see that the prediction value of the proposed model is closer to the true value. In addition, the error distribution curve of the proposed model is thinner and the errors are centered around zero, indicating that the proposed model has a better prediction accuracy.

Further discussion

In this subsection, we further perform a statistical analysis of the model complexity. In the proposed DRA-NSDE, the number of support vectors in SVR base learner is used to evaluate the complexity of a solution (Eq. (5)). Consequently, in the proposed ensemble model, the average support vector number of all SVR base learners is adopted as the evaluation of the model complexity. The same method is used to evaluate the complexity of other SVR-based ensemble models (*Bagging-SVR*, *AdaBoost-SVR*, *Stacking-SVR*). The statistical results in 15 independent experiments on both the UCI benchmark data set and the blast furnace data set are shown in Fig. 14.

From the results, we can see that the proposed model has the lowest complexity on both the UCI benchmark data set and the blast furnace data set. This indicates that the second objective (CMPLX) of solutions in the proposed DRA-NSDE is effective, as it enables our proposed model to improve the prediction accuracy while maintaining relatively low model complexity. This property of our proposed method can save a large amount of computational resources during model training process and accelerate the online prediction efficiency of blast furnace silicon content. In actual blast furnace ironmaking process, the silicon content is detected about every 40 min [53], while our model can produce online predictions in real time once it is trained offline. Therefore, our proposed method can fully satisfy the task of online prediction of silicon content in the actual blast furnace ironmaking process.

Conclusion

As an important indicator of operating conditions for blast furnace, accurate prediction of silicon content in the molten iron is very important to ensure stable production of blast furnace in the iron and steel industry. To achieve a prediction model with high accuracy and good generalization, this paper proposed a two-stage multiobjective evolutionary algorithm. The task of the first stage is to construct a set of near Pareto optimal base learners by means of the DRA-NSDE with dynamic resource allocation based on clustering, while the task of the second stage is to achieve the ensemble learning model from the candidate base learners using a single-objective differential evolution algorithm. There are three main characteristics of the proposed silicon content prediction method: 1)The proposed method chooses SVR as the base learner. Together with the designed solution encoding scheme, the input features of each base learner can be selected automatically so that the algorithm can not only extract meaningful combinations of features to improve the accuracy of the model, but also maintain better diversity of the ensemble model. 2) The proposed DRA-NSDE algorithm can dynamically allocate resources at different evolution stages based on the success rate of the two search strategies (local search and global search), which can solve the problem of difficult selection of target vectors in traditional NSDE. DRA-NSDE also improves the quality of the non-dominated solutions, which helps the algorithm to obtain faster convergence speed and better diversity. The diversity of the solutions to be integrated in stage two is further improved by the proposed CDE-Stacking method, which also optimizes the ensemble weights through a single-objective differential evolution algorithm. 3) The ensemble model has relatively lower model complexity, which is important in actual blast furnace ironmaking process. Experimental results on both the UCI benchmark data set and the actual blast furnace data set demonstrate the effectiveness of the improvement strategies used in the proposed two-stage MOEEL algorithm, and further comparison results show that the proposed algorithm can achieve prediction models with higher accuracy and lower complexity with comparison to some other powerful single machine learning methods and ensemble learning methods.

In the future work, it is worthwhile to further investigate how to select potential base learners from the clustered subpopulations in a more targeted manner rather than random selection. In addition, the application of the proposed algorithm to the operation optimization of the blast furnace ironmaking process is also one of our future research.

Declarations

Conflict of interest The authors declare that they do not have any commercial or associative interest that represents a conflict of interest in connection with the submitted work.

References

1. Zhou P, Li W, Wang H, Li M, Chai T (2019) Robust online sequential RVFLNs for data modeling of dynamic time-varying systems with application of an ironmaking blast furnace. IEEE Trans Cybern 50(11):4783–4795
2. Zhuang Z, Tao H, Chen Y, Stojanovic V, Paszke W (2022) An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints. IEEE Trans Syst Man Cybern Syst 53(6): 3461–3473
3. Zhou P, Guo D, Chai T (2018) Data-driven predictive control of molten iron quality in blast furnace ironmaking using multi-output LS-SVR based inverse system identification. Neurocomputing 308:101–110
4. Wang X, Hu T, Tang L (2021) A multiobjective evolutionary nonlinear ensemble learning with evolutionary feature selection for silicon prediction in blast furnace. IEEE Trans Neural Netw Learn Syst 33(5):2080–2093
5. Zhou H, Zhang H, Yang C (2019) Hybrid-model-based intelligent optimization of ironmaking process. IEEE Trans Ind Electron 67(3):2469–2479
6. Tang L, Meng Y (2021) Data analytics and optimization for smart industry. Front Eng Manag 8(2):157–171
7. Li Y, Zhang J, Zhang S, Xiao W (2022) Dual ensemble online modeling for dynamic estimation of hot metal silicon content in blast furnace system. ISA Trans 128:686–697
8. Luo S, Chen T (2020) Two derivative algorithms of gradient boosting decision tree for silicon content in blast furnace system prediction. IEEE Access 8:196112–196122
9. Zhang H, Zhang S, Yin Y, Chen X (2018) Prediction of the hot metal silicon content in blast furnace based on extreme learning machine. Int J Mach Learn Cybern 9:1697–1706
10. Fontes DOL, Vasconcelos LGS, Brito RP (2020) Blast furnace hot metal temperature and silicon content prediction using soft sensor based on fuzzy C-means and exogenous nonlinear autoregressive models. Comput Chem Eng 141:107028
11. Cardoso W, di Felice R, Baptista RC (2022) Artificial neural network for predicting silicon content in the hot metal produced in a blast furnace fueled by metallurgical coke. Mater Res 25 :e20210439
12. Liu C, Tan J, Li J, Li Y, Wang H (2022) Temporal hypergraph attention network for silicon content prediction in blast furnace. IEEE Trans Instrum Meas 71:1–13
13. Li J, Yang C, Li Y, Xie S (2021) A context-aware enhanced GRU network with feature-temporal attention for prediction of silicon content in hot metal. IEEE Trans Ind Inf 18(10):6631–6641
14. Zhao X, Fang Y, Liu L, Xu M, Zhang P (2020) Ameliorated moth-flame algorithm and its application for modeling of silicon content in liquid iron of blast furnace based fast learning network. Appl Soft Comput 94:106418
15. Hong W-C (2009) Hybrid evolutionary algorithms in a SVR-based electric load forecasting model. Int J Electr Power Energy Syst 31(7–8):409–417
16. Wang J, Li L, Niu D, Tan Z (2012) An annual load forecasting model based on support vector regression with differential evolution algorithm. Appl Energy 94:65–70
17. Hong W-C, Dong Y, Zheng F, Wei SY (2011) Hybrid evolutionary algorithms in a SVR traffic flow forecasting model. Appl Math Comput 217(15):6733–6747
18. Hong W-C, Dong Y, Chen L-Y, Lai C-Y (2010) Taiwanese 3G mobile phone demand forecasting by SVR with hybrid evolutionary algorithms. Expert Syst Appl 37(6):4452–4462
19. Nedic N, Stojanovic V, Djordjevic V (2015) Optimal control of hydraulically driven parallel robot platform based on firefly algorithm. Nonlinear Dyn 82:1457–1473
20. Stojanovic V, Nedic N (2016) A nature inspired parameter tuning approach to cascade control for hydraulically driven parallel robot platform. J Optim Theory Appl 168:332–347
21. Wang X, Dong Z, Tang L, Zhang Q (2023) Multiobjective multitask optimization-neighborhood as a bridge for knowledge transfer. IEEE Trans Evol Comput 27(1):155–169
22. Li M-W, Geng J, Hong W-C, Chen Z-Y (2017) A novel approach based on the Gauss-v SVR with a new hybrid evolutionary algorithm and input vector decision method for port throughput forecasting. Neural Comput Appl 28:621–640
23. Hamdi T, Ali JB, Di Costanzo V, Fnaiech F, Moreau E, Ginoux J-M (2018) Accurate prediction of continuous blood glucose based on support vector regression and differential evolution algorithm. Biocybern Biomed Eng 38(2):362–372
24. Nguyen H, Choi Y, Bui X-N, Nguyen-Thoi T (2019) Predicting blast-induced ground vibration in open-pit mines using vibration sensors and support vector regression-based optimization algorithms. Sensors 20(1):132
25. Wu J, Xie Y (2019) Hybrid support vector regression with parallel co-evolution algorithm based on GA and PSO for forecasting monthly rainfall. J Softw Eng Appl 12(12):524–539
26. Zhao J, Jiao L, Xia S, Fernandes VB, Yevseyeva I, Zhou Y, Emmerich MT (2018) Multiobjective sparse ensemble learning by means of evolutionary algorithms. Decis Support Syst 111:86–100
27. Ribeiro VHA, Reynoso-Meza G (2020) Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. Expert Syst Appl 147:113232

28. Chandra A, Yao X (2006) Ensemble learning using multi-objective evolutionary algorithms. J Math Model Algorithms 5(4):417–445
29. Ren Y, Zhang L, Suganthan PN (2016) Ensemble classification and regression-recent developments, applications and future directions. IEEE Comput Intell Mag 11(1):41–53
30. Minku LL, Yao X (2013) An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation. In: Proceedings of the 9th international conference on predictive models in software engineering. p 1–10
31. Wang X, Wang Y, Tang L (2021) Strip hardness prediction in continuous annealing using multiobjective sparse nonlinear ensemble learning with evolutionary feature selection. IEEE Trans Autom Sci Eng 19(3):2397–2411
32. Singh N, Singh P (2020) Stacking-based multi-objective evolutionary ensemble framework for prediction of diabetes mellitus. Biocybern Biomed Eng 40(1):1–22
33. Liu J, Chi Y, Liu Z, He S (2019) Ensemble multi-objective evolutionary algorithm for gene regulatory network reconstruction based on fuzzy cognitive maps. CAAI Trans Intell Technol 4(1):24–36
34. Song H, Qin AK, Salim FD (2018) Evolutionary multi-objective ensemble learning for multivariate electricity consumption prediction. In: 2018 International joint conference on neural networks (IJCNN). IEEE, p 1–8
35. Zhang C, Lim P, Qin AK, Tan KC (2016) Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Trans Neural Netw Learn Syst 28(10):2306–2318
36. Guo Y (2015) Topics in Bayesian adaptive clinical trial design using dynamic linear models and missing data imputation in logistic regression. PhD thesis, Baylor University
37. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
38. Krogh A, Vedelsby J (1994) Neural network ensembles, cross validation, and active learning. Adv Neural Inf Process Syst 7:231–238
39. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
40. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
41. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341-359
42. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
43. Huang VL, Qin AK, Suganthan PN, Tasgetiren MF (2007) Multiobjective optimization based on self-adaptive differential evolution algorithm. In: 2007 IEEE congress on evolutionary computation. IEEE, p 3601–3608
44. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
45. Zhang Q, Zhou A, Zhao S, Suganthan PN, Liu W, Tiwari S et al (2008) Multiobjective optimization test instances for the cec 2009 special session and competition. University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report 264, 1–30
46. Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary multiobjective optimization. Springer, p 105–145
47. Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7(2):117–132
48. Emmerich M, Beume N, Naujoks B (2005) An emo algorithm using the hypervolume measure as selection criterion. In: Evolutionary multi-criterion optimization: third international conference, EMO 2005, Guanajuato, Mexico, March 9–11, 2005. Proceedings 3. Springer, p 62–76
49. Angira R, Babu B (2005) Non-dominated sorting differential evolution (NSDE): an extension of differential evolution for multiobjective optimization. In: IICAI. p 1428–1443
50. Costa L, Oliveira P (2002) An evolution strategy for multiobjective optimization. In: Proceedings of the 2002 congress on evolutionary computation. CEC'02 (Cat. No. 02TH8600), vol 1. IEEE, p 97–102
51. Hansen N (2016) The CMA evolution strategy: a tutorial. arXiv:1604.00772
52. Thompson JJ, Blair MR, Chen L, Henrey AJ (2013) Video game telemetry as a critical tool in the study of complex skill learning. PLoS ONE 8(9):75129
53. Jiang Z-H, Dong M-L, Gui W-H, Yang C-H, Xie Y (2016) Two-dimensional prediction for silicon content of hot metal of blast furnace based on bootstrap. Acta Autom Sin 42(5):715–723