mathematics

MDPI

Article

An Offline Weighted-Bagging Data-Driven Evolutionary Algorithm with Data Generation Based on Clustering

Zongliang Guo, Sikai Lin, Runze Suo and Xinming Zhang *

School of Science, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
* Correspondence: xinmingxueshu@hit.edu.cn

Abstract: In recent years, a variety of data-driven evolutionary algorithms (DDEAs) have been proposed to solve time-consuming and computationally intensive optimization problems. DDEAs are usually divided into offline DDEAs and online DDEAs, with offline DDEAs being the most widely studied and proven to display excellent performance. However, most offline DDEAs suffer from three disadvantages. First, they require many surrogates to build a relatively accurate model, which is a process that is redundant and time-consuming. Second, when the available fitness evaluations are insufficient, their performance tends to be not entirely satisfactory. Finally, to cope with the second problem, many algorithms use data generation methods, which significantly increases the algorithm runtime. To overcome these problems, we propose a brand-new DDEA with radial basis function networks as its surrogates. First, we invented a fast data generation algorithm based on clustering to enlarge the dataset and reduce fitting errors. Then, we trained radial basis function networks and carried out adaptive design for their parameters. We then aggregated radial basis function networks using a unique model management framework and demonstrated its accuracy and stability. Finally, fitness evaluations were obtained and used for optimization. Through numerical experiments and comparisons with other algorithms, this algorithm has been proven to be an excellent DDEA that suits data optimization problems.

Keywords: data-driven evolutionary algorithm (DDEA); surrogate models; radial basis function networks; bagging

MSC: 68T07; 68T20

Citation: Guo, Z.; Lin, S.; Suo, R.; Zhang, X. An Offline Weighted-Bagging Data-Driven Evolutionary Algorithm with data generation Based on Clustering. *Mathematics* **2023**, *11*, 431. https://doi.org/10.3390/math11020431

Academic Editors: Yu Xue, Chunlin He and Ferrante Neri

Received: 29 November 2022
Revised: 6 January 2023
Accepted: 10 January 2023
Published: 13 January 2023

1. Introduction

In the past few decades, the evolutionary algorithm (EA) has become a popular approach for solving data optimization problems. However, traditional evolutionary algorithms need clear objective functions and constraints and rely heavily on fitness evaluations (FEs) to generate and select new populations [1,2]. Regarding practical problems, on the one hand, it is often difficult to obtain objective functions and constraints; on the other hand, FEs may be too expensive to access. To solve these problems, data-driven evolutionary algorithms (DDEAs) have been proposed. DDEAs aim to use datasets composed of historical data to construct a surrogate model which approximates the objective functions and constraints and then uses traditional EAs to optimize them in order to obtain an approximate solution to the original problem while simultaneously reducing the computational complexity [3]. DDEAs' performance has been proven to be excellent in issues such as airfoil design optimization problems [4], traffic signal timing optimization problems [5], trauma system design optimization problems [6], and so on. In optimization problems such as these, only historical data can be used, which are often hard to obtain due to high simulation costs or long time costs. Therefore, the DDEAs aiming to solve those optimization problems are called offline DDEAs, as they can only utilize the existing dataset, and no

more new data can be added to the dataset [4,7]. This article aims to build a type of DDEA suitable for solving offline data problems.

An essential measurement for evaluating DDEAs is approximation errors of the surrogate model, which directly determines the optimization accuracy to some degree [4]. Since an excellent surrogate model can greatly reduce the approximation error, the selection of a suitable surrogate model becomes an important part of constructing DDEAs [8]. In addition, how to make full use of the existing data to train the model is also an essential issue. Both aspects mentioned above have inspired us to search for answers using machine learning methods.

Most machine learning models and neural networks can be used as surrogates, including radial basis functions [8], polynomial response surface methods [9], Kriging methods [10,11], support vector machines [12], artificial neural networks [13], and so forth. Of these, the radial basis function(RBF) has advantages comprising fast convergence speed and strong robustness, and the approximation accuracy of neural networks (NN) is better than that of most other machine learning models. Based on the advantages of RBF and NN, in this paper, we adopt the radial basis function network as the basic surrogate model and combine a new proposed model management strategy named weighted-bagging (W-bagging) to propose a W-bagging data-driven evolutionary algorithm with data generation based on clustering (WDDEA-DBC). The proposed algorithm is driven by the following three motives.

First, considering that data generation always takes a significant proportion of runtime in DDEAs and that the data need to be clustered before the training of RBFNs [14], we came up with an idea to carry out data generation based on clustering (DBC). DBC works as follows. First, the dataset is clustered and DBC evaluates whether the cluster needs to generate data by using the compactness value (CP) of each cluster. Then, for those clusters that need to generate data, the DBC generates data in its small neighborhood for each point within them, making its CP value lower. Then the CP value of each cluster is evaluated again and iterated until convergence. As a result, DBC has two advantages.

1. Since this process is carried together with clustering and the CP values are calculated and new data are generated very quickly, it takes less running time.
2. Since the criterion of DBC is the CP value of clusters, it alleviates the uneven data distribution and thus ensures the good training of the surrogate models.

Second, although the performance of RBFN is good enough, any neural network needs enough data to be trained. In the case of offline optimization problems, we borrow the idea of bagging in ensemble learning and resample the data through bootstrap [15] to train multiple RBFNs and make the best possible use of the data as well as reduce the fitting error. To ensure that the trained network has strong enough heterogeneity, we adopt the strategy of the random learning rate, that is, the learning rate of each network is random and unknown within limits during training, and the model performance is evaluated by the mean square error (MSE) after training.

Finally, differently to the traditional bagging-solution regression method, which averages the results directly, we propose a weighted bagging framework that averages the MSE of each model. In this way, W-bagging plays a role in model selection, as it determines the weight of the model according to the value of MSE. If the performance is too poor, i.e., the MSE is too high, the model is given a smaller weight by W-bagging as a way to act as a model selection strategy. We also prove the effectiveness and stability of W-bagging through experiments and mathematical derivation.

The remainder of this paper is organized as follows: In Section 2, we briefly introduce DDEA and related work. In Section 3, the proposed algorithm is presented in detail. Section 4 presents the experimental setting, the experimental results, comparisons with representative algorithms, and discussions. Section 5 provides the conclusion of this paper.

2. Background and Related Work

2.1. Data-Driven Evolutionary Algorithms

As mentioned above, differently to traditional EAs, the focus of DDEAs is to obtain better fitting and optimization effects with discrete data. The way to achieve this is to build surrogate models and use the surrogates to replace real FEs, thereby removing the need to access real fitness. In terms of the algorithm framework, the model is generally divided into two parts: the model management strategy and the evolutionary optimization algorithm [3]. The model management strategy aims to select and aggregate models in a certain way so that the results obtained by the models are closer to the real fitness. In addition, DDEAs can be divided into two categories according to whether they can obtain new data during the optimization process: online DDEA and offline DDEA [4,16]. In online DDEA, users can still obtain new data for fitness assessment during the evolutionary optimization process in order to update the surrogate models. Therefore, the research on online DDEA focuses on how to use as few new data points as possible to build more accurate models [3,17]. Although relatively little research has been undertaken in regard to online DDEAs, due to their feasibility, ease of operation, and low dependence on data, online DDEAs are being used in a number of engineering fields. For example, to reduce the flood risk in urban areas, Xuan et al. proposed an online data-driven evolutionary algorithm-based optimal design of urban stormwater-drainage systems [18]. Since the focus of this paper is on offline DDEAs, the work on online DDEAs will not be repeated.

In offline DDEAs, users cannot obtain any new data and can only use historical data for optimization; thus, offline DDEA performance is mainly constrained by two problems, namely a lack of data and the uncertainty of model reliability [16]. As mentioned above, although there are differences between online and offline DDEAs, the essence of both is the reduction of the number of FEs and optimization by constructing surrogate models to simulate real fitness.

This subsection briefly reviews the research status of offline DDEAs, the differences between them, and the model proposed in this paper. In offline scenarios, accuracy is severely affected by the inability to evaluate new data. Wang et al. proposed a DDEA using an integrated surrogate (DDEA-SE) [4]. Such methods use existing data to construct a series of surrogates and select these models through certain model management strategies, which are used to approximate FEs at different evolutionary stages, thereby reducing prediction errors. This type of method is quite advanced, and many current DDEA models use this idea [4–7], including the proposed WDDEA-DBC. On the other hand, the noise in the data is also a major factor that affects the accuracy of DDEAs [19]. Therefore, many data pre-processing methods are used in DDEAs to solve this kind of problem. For example, the local regression method proposed by Chugh et al. in the multi-objective blast furnace problem is used to reduce dataset noise [20]. For massive data scenarios, unsupervised learning techniques such as clustering and dimension reduction can be used to reduce data redundancy [21]. For example, in the problem of trauma system design, Wang et al. proposed a schema that employs a clustering approach to build the surrogate model, which ultimately saves approximately 90% of the total runtime [6]. In addition, insufficient data can be improved by generating new data. For example, in the multi-objective magnesium melting furnace optimization problem, Guo et al. used low-order polynomials to generate synthetic data and predict their practicality [22]. Li et al. proposed a DDEA combined with a boosting technique and introduced a new data generation method called localized data generation (LDG), which improves the problem of a lack of data while speeding up the algorithm and significantly reduces the runtime compared to similar algorithms [5].

Similar to the above algorithms, the proposed WDDEA generated based on clustering data is also an offline DDEA. It is worth mentioning that W-bagging can be used both as a model management strategy and a model selection method due to its feature of using MSE as model weights. In addition, referring to the idea suggested by Guo and Li et al., a new data generation method based on clustering (DBC) is proposed, so that the model still has a strong fitting ability in places with few data points.

2.2. RBFN and Bagging

In this paper, we use RBFN as a surrogate model, combined with bagging as our model management strategy. The following describes the related backgrounds and works of RBFN and bagging.

The radial basis function network (RBFN) is a single hidden layer, function approximation-based RBFN proposed in the late 1980s based on RBFs [23,24]. With the maturity of the research, RBFN has received significant attention from researchers in various fields due to its simple structure, strong nonlinear approximation capability, and good generalization ability, and it is widely used in many research fields such as pattern classification, function approximation, and data mining [25–27]. In RBFN, the number of hidden layer nodes is an important factor. Within a certain range, a smaller number of nodes can reduce the number of operations and obtain a better generalization ability. Therefore, determining the centroids of the RBFN is an important issue. Moody-Darken proposed a clustering-based K-means method [28] by clustering the independent variables of training samples and using the number of clusters as the hidden unit. The number of clusters is used as the number of hidden units. The center of the clusters is used as the RBF center, and the width parameter is generated using the clustering variance. However, this method has obvious drawbacks: (1) the clustering is sensitive to the initial value, which can lead to poor results if the parameters are not set properly; (2) the selection of the number of clusters affects the clustering results: if they are too small, they will not fully train the sample, and if they are too large, this will lead to some classes being too sparse; (3) the dependent variable of the training sample is not utilized. In addition, there are other methods used to determine the centroids of RBFNs, such as the system proposed by Gomm et al., who determined the centroids of RBFNs by recursive orthogonal least squares regression, and Chen et al. also used the same centroid selection method in their related work [29–31]. The drawbacks of these methods are that the width parameters need to be selected subjectively and the orthogonalization methods tend to lose sample information, which affects the network performance. As an improvement, Walczak and Massart introduced the partial least squares regression method into RBFN [32,33], which solves the problem of information loss but still requires pre-given width coefficients and is computationally intensive. To overcome these shortcomings and simplify the training process, this paper compares the MSE and running time variation of the surrogate model for different numbers of clusters (number of centroids) by numerical experiments. The most suitable number of clusters is selected according to the results, the clusters correspond to the centroids one by one, and the width factor of each centroid is determined by its corresponding cluster. The numerical results are presented in Section 5.

Bagging is a method widely used in the field of ensemble learning to improve the accuracy of learning algorithms. It was proposed by Breiman in 1996 [34] by constructing a sequence of predictors and combining them into a stronger predictor in an averaging manner. In each training round, a certain amount of data is randomly resampled from the initial training set by bootstrap sampling as the training set for the round in question. Due to the nature of bootstrap sampling, the initial training samples can appear multiple times or not in a particular training set. Several rounds of training and a sequence of prediction functions can be obtained, and the final prediction functions are obtained using voting for classification problems and simple averaging for regression problems. Compared with another integrated method boosting [35], the advantage of bagging is stability. For unstable algorithms, bagging can significantly improve the prediction accuracy, while it is not very effective for stable learning algorithms. Since the neural network is a type of unstable learning algorithm, bagging can be used for algorithms with neural network-based learners and can save considerable time overheads by utilizing parallel training. A variant of bagging, random forest [36], is used in many machine learning problems and consistently displays good performance. An improved bagging algorithm for imbalanced datasets [37] has also been proposed in recent years in order to improve the prediction accuracy for a small amount of data. In this paper, we propose another weight-based variant of bagging, W-bagging, to obtain the final prediction function by weighted averaging instead of simple

averaging. In the following sections, we demonstrate that this method has higher accuracy and better stability than the original bagging method through many numerical tests and mathematical derivations.

In addition, there are many boosting-based methods in the field of ensemble learning, such as AdaBoost [38], Xgboost [39–41], and GBDT [42], which can also be applied to the problem in this paper, and model management strategies based on these methods could be considered in the future.

3. Proposed Algorithm

3.1. Data Generation Based on Clustering

The main idea of DBC is to synthesize data in positions where data points are sparse to improve the quality of the surrogate model. Drawing on the idea of LDG in [5], the original data can be presented as data-fitness pairs to form a training dataset:

$$TD = \{\boldsymbol{x}_i, F(\boldsymbol{x}_i) | I = 1, 2, 3, ...\}, \tag{1}$$

where $F(\boldsymbol{x}_i)$ is the fitness of $\boldsymbol{x}_i$. The DBC's mission is to produce synthetic data based on original data in TD. The first step of DBC is to determine which regions need data generation. Several clusters are obtained by K-means clustering and sorted by cluster compactness CP, which is calculated as follows

$$CP_j = \sum_{i=1}^{n_j} \frac{||\boldsymbol{x_i} - \boldsymbol{c_j}||}{n_j}, j = 1, 2, \cdots, m, \tag{2}$$

where n_j is the number of points in cluster j and c_j is the centroid of cluster j. Then, the data generation is carried out on m clusters X_m with the highest CP, as a higher CP indicates a sparser data distribution. The input of the generated data points based on $\boldsymbol{x}_i \in X_m$ is $\boldsymbol{x}_i + \boldsymbol{\delta}_i$, where $\boldsymbol{\delta}_i$ is a vector following the d-dimensional normal distribution with mean 0 and covariance L_n, $L_n = l \cdot E_n (l > 0)$, E_n is the N-order identity matrix here, and the output is $F(\boldsymbol{x}_i)$. The resultant dataset K is denoted by

$$K = \{\boldsymbol{x}_{new}, F(\boldsymbol{x}_i) | \boldsymbol{x}_{new} = \boldsymbol{x}_i + \boldsymbol{\delta}_i\}, \tag{3}$$

where

$$\boldsymbol{\delta}_i \sim N_d(0, L_n), \boldsymbol{x}_i \in X_m, \tag{4}$$

$$l = \sqrt{\frac{\sum_{j=1}^{D} (U_j - L_j)^2}{D}} \cdot 10^{-6}, \tag{5}$$

where U_j and L_j represent the upper and lower bounds of the jth dimension, respectively. The role of l is to control the newly generated data within a small neighborhood of the original data, ensuring the quality of the generated data.

The original and synthetic datasets were combined to obtain the final augmented dataset ATD, where

$$ATD = TD \bigcup K. \tag{6}$$

The reason we decided to cluster the data first, rather than directly generate the data, is because calculating where and how much data are generated often incurs considerable time costs, and our goal is to create a fast and efficient data generation method.

Similar to the DBC workflow shown in Figure 1, Algorithm 1 simply gives the pseudocode of the DBC. The input of the DBC is the original dataset TD, the number of clusters is m, and the output is the synthetic dataset K. There are four main steps to the algorithm. The first step is to divide the data into m clusters through K-means clustering [43], obtain the cluster center and intra-cluster compactness CP and sort them according to CP

in descending order, and extract the samples of the first $40\%/d \cdot n$ clusters as objects for subsequent steps.

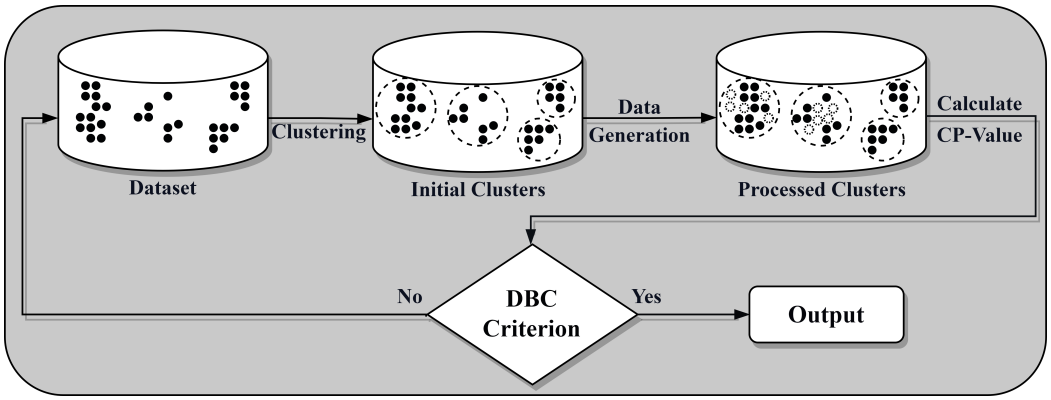


Figure 1. The procedure of DBC.

The second step is to generate a random variable that obeys the normal distribution of the mean 0 and the covariance of L_n at each point of the chosen clusters, where the value of l is given by Equation (5), and the random variable is given by the generated data points in the manner of Equation (4), which is classified into the set K. The third step is to cluster all the datasets to obtain m data centers and CPs as parameters for subsequent model building. Notably, if the number of clusters m is too large when clustering in the first and third steps, it is possible that a cluster has only one point and CP does not exist. To avoid this, the method of missing value filling is used to calculate the average CP for all clusters with at least two points, and the average CP is used as the CP of the cluster with only one point. In addition, the reason for data generation for the top $40\%/d$ of clusters is that we want to balance the quality of the data generated and the running time of the algorithm. The analysis in Section 4.6 and the results show that 40% is a more feasible number, and the reason for dividing by d is that as the dimension increases, the time taken to generate the data and calculate the CP value is greater, so the number of clusters being manipulated needs to be reduced accordingly.

Algorithm 1 Data generation based on clustering (DBC).

Input: TD—the original training dataset
n—the number of clusters
Output: ATD—the synthetic dataset
Begin
$CPs, IDs, Labels = Kmeans(TD, n)$
Sort ID according to their CP with ascending order
Set S as the first 40% samples of the sorted ID
Set ATD as an empty set
For each id in S **Do**
For each x_j in TD **Do**
If $Label(x_j)$ equals id **Do**
Generate x_{new} through x_j and Equation (2)
Set $y_{new} = y_j$
$ATD = ATD \bigcup (x_{new}, y_{new})$
End For
End For
End

3.2. Radial Basis Function Networks

As an important part of DDEAs, the surrogate model needs strong approximation ability and convergence [44]. Although most ANNs can fit the data well, they generally

have the problem of low-speed convergence. In DDEAs research, many schemes use a radial basis function neural network (RBFN) as a surrogate model [3], which has three layers: input layer, hidden layer, and output layer, and the number of units is n, q and m, respectively. The input layer receives n-dimensional vectors, and the hidden layer uses the radial basis function to activate the input and then transmits it to the output layer [45]. The output layer is the network output after linear processing, which aims to realize the mapping from n-dimensional input to m-dimensional output, as shown in Equation (7).

$$y_i = \sum_{j=1}^{q} w_{ji}\phi(\|\boldsymbol{x} - \boldsymbol{c}_i\|), i = 1, 2, \cdots, m, \tag{7}$$

where $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ is the input vector, y_i is the output vector, w_{ji} is the connection weights of the hidden layer to the output layer, $\|\cdot\|$ is the Euclidean function, $\phi(\cdot)$ is the radial basis function, and $\boldsymbol{c}$ is the center vector. In this paper, since fitness is a number, $m = 1$.

After defining the general structure of RBFN, as is shown in Figure 2, the next step is to determine the value of each parameter, such as the number of hidden elements q, the center point c_j of each element, and the width parameter σ. Fortunately, this part of the work has already been achieved by the DBC proposed by III-A. The number of clusters k selected by the DBC can be used as the number of hidden units q, the centers of the clusters can be used as the respective c_j, and the CP of the jth cluster can be used to determine the width parameter σ_j of the jth unit center. The formula is

$$\sigma_j = \frac{1}{2CP_j^2}, j = 1, 2, \cdots, q. \tag{8}$$

By determining the network structure, DBC can effectively introduce the prior information provided by the data to accelerate the convergence speed. In addition, there is a structural problem to be solved: that is, how to initialize the weights w_{ji} from the hidden layer units to the output layer. The widely used Kaiming initialization has been chosen for this article. Finally, for the training of the network, adaptive moment estimation (Adam) was selected for the parameter update in this paper to ensure that the size of the parameter update does not change with the scaling of the gradient size. The details of Kaiming initialization and Adam can be found in the literature [46,47] and are not repeated here.

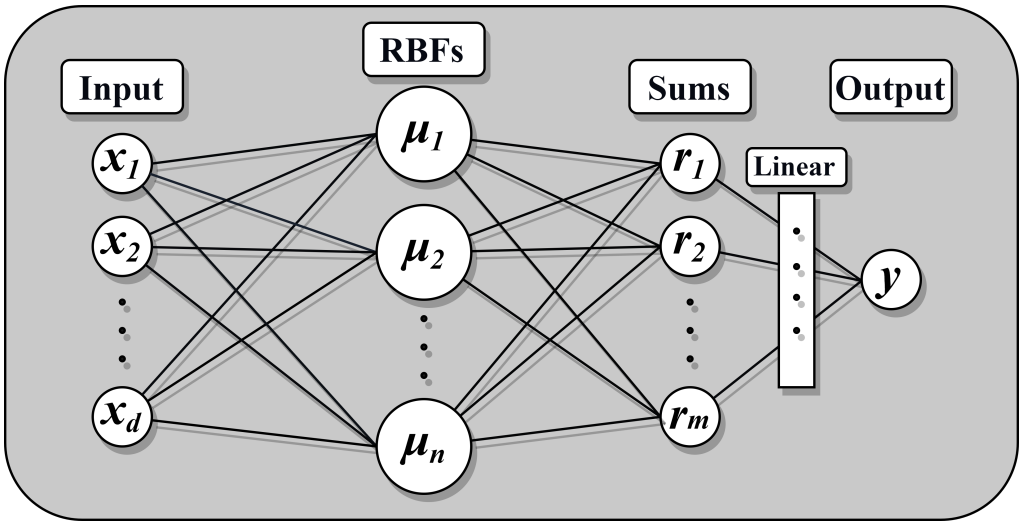


Figure 2. The structure of RBFN.

3.3. Weighted-Bagging

This article provides W-bagging to achieve a more fit-to-fit and stable final model. W-bagging adds fitting information of MSE on the basis of the bagging method of ensemble learning so that the fitting effect is better. This section first gives a brief introduction to bagging, then introduces W-bagging, and finally proves its effectiveness and stability through mathematical derivation. Bagging is a method of generating multiple predictors and using them to obtain an aggregated predictor, which is widely used in the field of machine learning [48,49]. This method can significantly improve accuracy and stability. For offline DDEAs that lack data, bagging can serve the purpose of generating a strong surrogate model using the available data as much as possible. In this work, the use of RBFN as surrogate models and having multiple weak RBFN aggregated as a stronger surrogate model by bagging is based on the following two considerations. First, as mentioned above, RBFN is a suitable model for solving function fitting problems and is widely used in the work of DDEAs. Secondly, although RBFN may not be as effective as it could be with the small amount of data and low number of iterations in this work, it can be used by bagging to aggregate into a strong surrogate model due to the one-to-one input–output property of its neural network. W-bagging differs from bagging in how the final result is obtained through the results of multiple predictors. Bagging is direct averaging, while W-bagging is weighted averaging, and the weight is positively correlated with the fitting ability of the predictor. The fitting ability is measured by MSE in this work. The workflow of W-bagging is shown in Figure 3, while its pseudocode is given in Algorithm 2.

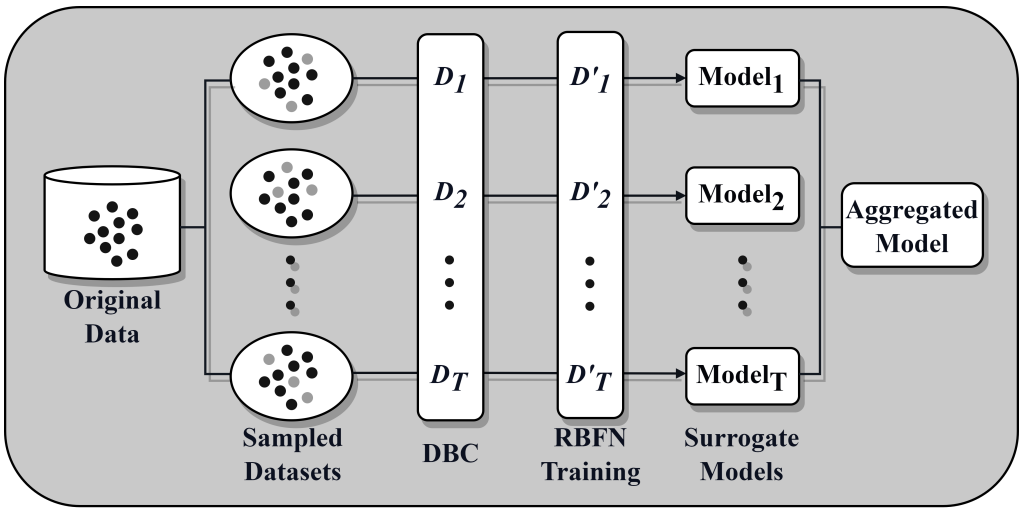


Figure 3. The procedure of W-bagging.

Algorithm 2 Weighted-bagging.

Input: TD—the original training dataset
k—the number of DBC clusters
b—the number of samples in each iteration
T—the number of surrogate models to be obtained
Output: $\hat{g}(\boldsymbol{x})$—the aggregated model
Begin
For $i = 1$ to T **Do**
Randomly select b points from TD to form TD_i
$ATD_i = DBC(TD_i, k)$ //refer to Algorithm 1.
Use ATD_i to train RBFN, build predictor $g_i(\boldsymbol{x})$
$MSE_i = \sum (y_j - g_i(\boldsymbol{x}_j))^2 / |ATD_i|, (\boldsymbol{x}_j, y_j) \in ATD_i$
End For
$\hat{g}(\boldsymbol{x}) = \sum \frac{g_i(\boldsymbol{x})}{MSE_i} / \sum \frac{1}{MSE_i}$
End

Next, we want to prove the effectiveness and stability of W-bagging by mathematical derivation. Without loss of generality, $1/MSE$ is taken as the W-bagging weight in the following proof.

Here, we suppose the results of the i-th predictors are $g_i(\boldsymbol{x})$, and the final results obtained by bagging and W-bagging are Equations (9) and (10).

$$\hat{g}_1(\boldsymbol{x}) = \frac{\sum_1^n g_i(\boldsymbol{x})}{n}, \tag{9}$$

$$\hat{g}_2(\boldsymbol{x}) = \sum \frac{g_i(\boldsymbol{x})}{MSE_i} / \sum \frac{1}{MSE_i}. \tag{10}$$

The first part compares the variance of the two methods. For convenience, let $\lambda_i = 1/(MSE_i)$. Calculate the variance of the fitting function obtained by the two methods

$$Var(\hat{g}_1(\boldsymbol{x})) = Var\left(\frac{\sum_1^n g_i(\boldsymbol{x})}{n}\right) = \frac{\sum_1^n Var(g_i(\boldsymbol{x}))}{n^2} = \frac{Var(g_i(\boldsymbol{x}))}{n}, \tag{11}$$

$$Var(\hat{g}_2(\boldsymbol{x})) = Var\left(\frac{\sum_1^n \lambda_i g_i(\boldsymbol{x})}{\sum_1^n \lambda_i}\right) = \frac{\sum_1^n \lambda_i^2 Var(g_i(\boldsymbol{x}))}{(\sum_1^n \lambda_i)^2}. \tag{12}$$

The reason why the last equal sign of Equation (11) holds is that $g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \cdots, g_n(\boldsymbol{x})$ are independent and identically distributed as they are generated by data sampled from the same dataset. Notice that both results contain $Var(g_i(\boldsymbol{x}))$, so the comparison of the variance is equivalent to the comparison of the coefficients of $Var(g_i(\boldsymbol{x}))$. The coefficient of Equation (12) is expanded as follows:

$$\frac{\sum_1^n \lambda_i^2}{(\sum_1^n \lambda_i)^2} = \frac{\sum_1^n \lambda_i^2}{\sum_1^n \lambda_i^2 + \sum_1^n (\sum_1^i \lambda_i \lambda_j + \sum_{i+1}^n \lambda_i \lambda_j)}. \tag{13}$$

The numerator and the denominator are divided by $\sum_1^n \lambda_i^2$:

$$\frac{\sum_1^n \lambda_i^2}{\sum_1^n \lambda_i^2 + \sum_1^n (\sum_1^i \lambda_i \lambda_j + \sum_{i+1}^n \lambda_i \lambda_j)} = \frac{1}{1 + \sum_1^n \frac{\sum_1^i \lambda_i \lambda_j + \sum_{i+1}^n \lambda_i \lambda_j}{\sum_1^n \lambda_i^2}}. \tag{14}$$

The expectation of the last part in the divisor is taken as:

$$\sum_1^n \frac{\sum_1^i \lambda_i \lambda_j + \sum_{i+1}^n \lambda_i \lambda_j}{\sum_1^n \lambda_i^2} = \frac{(n^2 - n) E(\lambda_i \lambda_j)}{n E(\lambda_i^2)} = \frac{(n-1) E(\lambda_i \lambda_j)}{E(\lambda_i^2)}. \tag{15}$$

Since

$$\begin{gathered} Var(\lambda_i) = E(\lambda_i^2) - E^2(\lambda_i) > 0, \\ \Updownarrow \\ E(\lambda_i^2) > E^2(\lambda_i). \end{gathered} \tag{16}$$

We then have:

$$\begin{gathered}\frac{(n-1)E(\lambda_i\lambda_j)}{E(\lambda_i^2)} < n-1,\\ \Downarrow \\ \frac{1}{1+\sum_1^n \frac{\sum_1^i \lambda_i\lambda_j+\sum_{i+1}^n \lambda_i\lambda_j}{\sum_1^n \lambda_i^2}} > \frac{1}{n}.\end{gathered} \tag{17}$$

Finally, with the above proofs, we can see that the variance of W-bagging is larger than the variance of bagging, but by how much?

To this end, we calculate as follows:

$$\begin{aligned}\frac{1}{1+\frac{(n-1)E^2(\lambda_i)}{E(\lambda_i^2)}} - \frac{1}{n} &= \frac{nE(\lambda_i^2) - E(\lambda_i^2) - (n-1)E^2(\lambda_i^2)}{n\left[E(\lambda_i^2) + (n-1)E^2(\lambda_i^2)\right]}\\ &= \frac{(n-1)Var(\lambda_i)}{nVar(\lambda_i) + n^2E(\lambda_i)},\end{aligned} \tag{18}$$

where $\lambda_i = 1/(MSE_i)$ is generally greater than 0 and less than 1. In addition, n is a large number in this case. Thus, we can draw the conclusion that W-bagging may be inferior to traditional bagging in stability, but the difference is very small, and this difference can be reduced by reducing MSE or increasing the number of samples.

The second part compares the MSE of the two methods. Assume the dataset $ATD = (x_i, y_i)$, $1 \le i \le m$, and the n predictors $g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \cdots, g_n(\boldsymbol{x})$ are the same as above. Assign y_{ij} as $g_j(x_i), 1 \le j \le n$.

Then, we have:

$$MSE_j = \sum_{j=1}^{m} \frac{(y_i - \hat{y}_{ij})^2}{m}, \tag{19}$$

$$\hat{g}_1(\boldsymbol{x}_i) = \frac{\sum_{j=1}^n g_j(\boldsymbol{x}_i)}{n} = \frac{\sum_{j=1}^n \hat{y}_{ij}}{n}, \tag{20}$$

$$\hat{g}_2(\boldsymbol{x}_i) = \frac{\sum_{j=1}^n \frac{g_j(\boldsymbol{x}_i)}{MSE_j}}{\sum_{j=1}^n \frac{1}{MSE_j}} = \frac{\sum_{j=1}^n \frac{\hat{y}_{ij}}{MSE_j}}{\sum_{j=1}^n \frac{1}{MSE_j}}. \tag{21}$$

Next, we calculate the MSE of bagging and W-bagging, which are named as $\widehat{MSE}_1$ and $\widehat{MSE}_2$. Again, for the convenience of calculation, let $\lambda_j = 1/MSE_j, \varepsilon_{ij} = y_i - \hat{y}_{ij}$:

$$\begin{aligned}\widehat{MSE}_1 = \sum_{i=1}^{m} \frac{(y_i - \hat{g}_1(\boldsymbol{x}_i))^2}{m} &= \sum_{i=1}^{m} \frac{\left(y_i - \frac{\sum_{j=1}^n \hat{y}_{ij}}{n}\right)^2}{m}\\ = \sum_{i=1}^{m} \frac{\left(\frac{\sum_{j=1}^n (y_i - \hat{y}_{ij})}{n}\right)^2}{m} &= \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^n \varepsilon_{ij}\right)^2}{mn^2},\end{aligned} \tag{22}$$

$$\begin{aligned}\widehat{MSE}_2 &= \sum_{i=1}^{n} \frac{(y_i - \hat{g}_2(\boldsymbol{x}_i))^2}{n} \\ &= \sum_{i=1}^{n} \frac{\left(y_i - \sum_{j=1}^{t} \frac{\hat{y}_{ij}}{MSE_j} / \sum_{j=1}^{t} \frac{1}{MSE_j}\right)^2}{n} \\ &= \sum_{i=1}^{n} \frac{\left(\sum_{j=1}^{t} \frac{y_i}{MSE_j} - \sum_{j=1}^{t} \frac{\hat{y}_{ij}}{MSE_j}\right)^2}{n\left(\sum_{j=1}^{t} \frac{1}{MSE_j}\right)^2} \\ &= \sum_{i=1}^{n} \frac{\left(\sum_{j=1}^{t} \lambda_j \varepsilon_{ij}\right)^2}{n\left(\sum_{j=1}^{t} \lambda_j\right)^2}, \end{aligned} \tag{23}$$

$$\begin{aligned}\widehat{MSE}_1 - \widehat{MSE}_2 &= \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{n} \varepsilon_{ij}\right)^2}{mn^2} - \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{n} \lambda_j \varepsilon_{ij}\right)^2}{m\left(\sum_{j=1}^{n} \lambda_j\right)^2} \\ &= \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{n} \varepsilon_{ij} \sum_{j=1}^{n} \lambda_j\right)^2 - \left(n \sum_{j=1}^{n} \lambda_j \varepsilon_{ij}\right)^2}{m\left(n \sum_{j=1}^{n} \lambda_j\right)^2}. \end{aligned} \tag{24}$$

The numerator of Equation (24) equals

$$\begin{aligned}&\left(\sum_{j=1}^{n} \varepsilon_{ij} \sum_{j=1}^{n} \lambda_j\right)^2 - \left(n \sum_{j=1}^{n} \lambda_j \varepsilon_{ij}\right)^2 \\ &= \left(\sum_{j=1}^{n} \lambda_j\right)^2 \left(\sum_{j=1}^{n} \varepsilon_{ij}\right)^2 - n^2 \left(\sum_{j=1}^{n} \lambda_j \varepsilon_{ij}\right)^2. \end{aligned} \tag{25}$$

All parentheses are then expanded, where the $k \cdot l$th entry is:

$$\left[\left(\sum_{j=1}^{n} \lambda_j\right)^2 - n^2 \lambda_k \lambda_l\right] \cdot \varepsilon_{ik} \varepsilon_{il}. \tag{26}$$

Taking their expectation, where λ_k, λ_l is independent, $\varepsilon_{ik}, \varepsilon_{il}$ is independent:

$$\begin{aligned}&E\left\{\left[\left(\sum_{j=1}^{n} \lambda_j\right)^2 - n^2 \lambda_k \lambda_l\right] \varepsilon_{ik} \varepsilon_{il}\right\} \\ &= n^2 \left[E\left(\lambda^2\right) - E(\lambda)^2\right] E(\varepsilon)^2 \\ &= n^2 Var(\lambda) E(\varepsilon)^2 > 0. \end{aligned} \tag{27}$$

Thus, we have $E\left[\widehat{MSE}_1 - \widehat{MSE}_2\right] \geq 0$, which means $\widehat{MSE}_1 \geq \widehat{MSE}_2$ in most cases. Therefore, it can be concluded that W-bagging has a better effect on reducing MSE than traditional bagging. Furthermore, to better verify this conclusion, we will conduct further numerical experiments in Section 5.

After introducing DBC and W-bagging, we obtained the preliminarily complete algorithm named the weighted-bagging data-driven evolutionary algorithm (WDDEA-DBC), as shown in Figure 4. WDDEA-DBC can be divided into two parts: FS and OS. The OS part is similar to traditional EAs that contain initialization, variation, and FE. In the selection of EA, this paper uses a modified pollination algorithm (FPA). Other algorithms, such as

particle swarm optimization, the cuckoo algorithm, and the firefly algorithm, could also be used in future works.

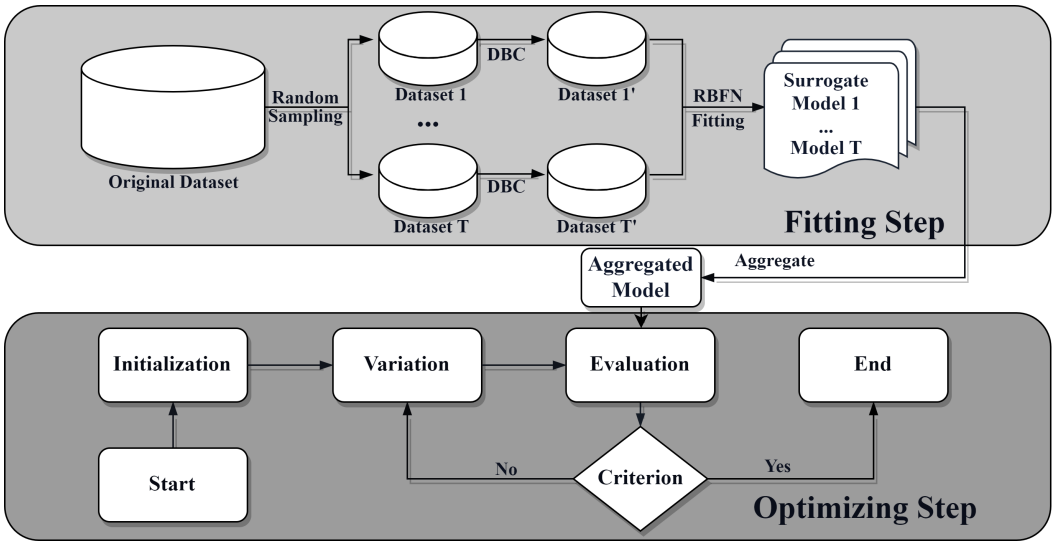


Figure 4. The diagram of complete WDDEA-DBC.

3.4. Whole Proposed Algorithm

The whole algorithm works as follows. First, based on historical data, the generated dataset is obtained through DBC to train the surrogate model. Then, the aggregated model is obtained through W-bagging. When the OS performs FE, the prediction results of the aggregated model are utilized as the fitness of the individual. In this way, the OS can use these predictions to drive EAs. When the stop criteria are met, the OS will output the best individual based on the prediction as the final solution, and then the algorithm completes.

4. Experimental Studies

4.1. Experimental Setup

In the experiments, five benchmark functions [50] were used to test WDDEA-DBC, and their functional forms and global optimum are the same as those set in the paper. Specific information is shown in Table 1. To reflect the effectiveness of the proposed algorithm, some state-of-the-art DDEA algorithms are used as comparison objects. These DDEAs are DDEA-SE [4], BDDEA-LDG [5], and TT-DDEA [16]. In addition to their good performance, these algorithms are chosen for other reasons. First, DDEA-SE, as the most classic offline DDEA, has strong performance and is very suitable for comparison among offline data-driven models. Second, BDDEA-LDG uses a boosting-like model management strategy, which can be well compared with the W-bagging framework in this paper, and its local data generation method LDG can also be used as a comparison with the DBC method in this paper, thus highlighting the effectiveness of WDDEA-DBC. Finally, TT-DDEA, as the latest proposed offline DDEA with strong optimization capability and very fast running speed, reflects the superiority of WDDEA-DBC by comparison.

Table 1. Benchmark problems.

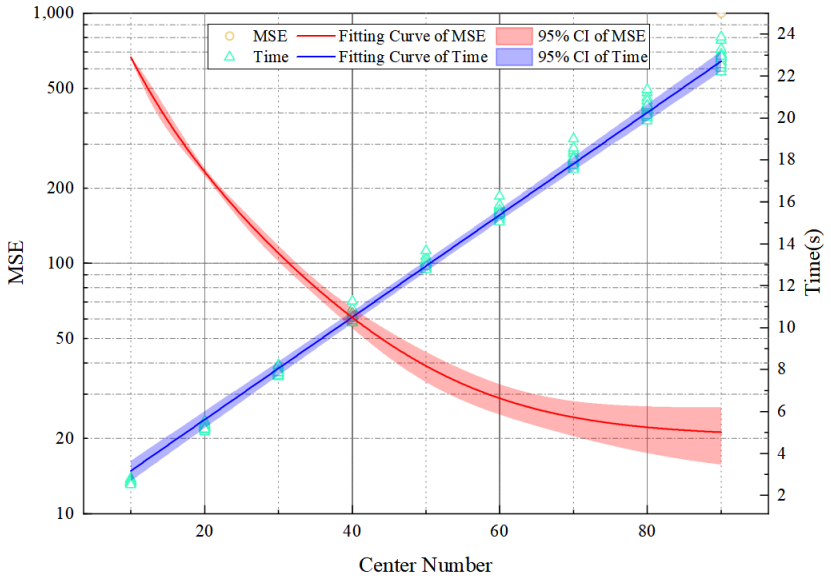
Problem	Optimum	Characteristics	Dimension
Ellipsoid	0	Uni-modal	10, 30, 50, 100
Rosenbrock	0	Multi-modal	10, 30, 50, 100
Ackley	0	Multi-modal	10, 30, 50, 100
Griewank	0	Multi-modal	10, 30, 50, 100
Rastrigin	0	Multi-modal	10, 30, 50, 100

In the experiments, all the compared algorithms are configured based on their original papers. For WDDEA-DBC, the underlying optimization algorithm uses a variant of FPA [51]. In addition, to ensure fair comparisons, the parameter configurations were kept consistent across optimization algorithms: the population size was 100 and the variance probability was $1/D$, where D is the function dimension.

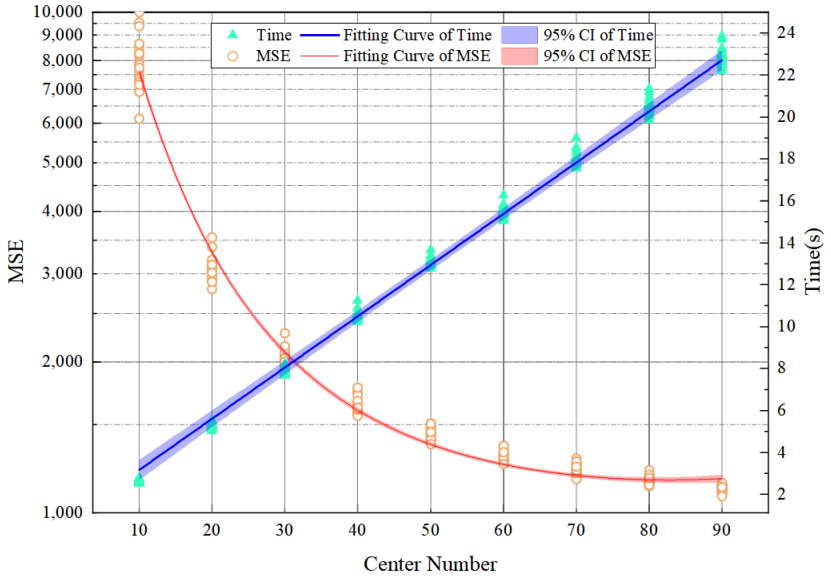
For the surrogate model part, WDDEA-DBC uses RBFN. There are two main reasons for using RBF neural networks. First, RBFN has the advantages of simple structure, high approximation ability, and high computational efficiency. Second, the DDEA-SE, BDDEA-LDG and TT-DDEA algorithms also used RBFN as the surrogate model, which can ensure the fairness of the comparison.

In terms of network parameters, the activation function of WDDEA-DBC is the Gaussian radial basis function, and the number of neurons in its hidden layer is taken as 50 after the numerical experimental analyses of MSE and the time change with the neuron number on the Ellipsoid function and Rastrigin function shown in Figure 5. The learning rate of the RBFN training process is a random number between 1×10^{-5} and 1×10^{-7}, and the regularization coefficient is 0.02. For the compared algorithms, the parameters of the RBFN are also configured according to their original papers.

To further ensure the fairness of the numerical experiments, this paper also takes the same data sampling and testing methods as those in the compared works. First, the maximum number of available FEs for all algorithms is $11 - D$. For the offline DDEA, sampling is performed by the Latin hypercube sampling method (LHS) [52] before optimization, and no new real FEs can be introduced during the optimization process. For the online algorithms, the parameter configurations are the same as the original papers, and the FEs are obtained and used according to the strategies in the respective papers; however, it must be ensured that the maximum number does not exceed $11 - D$. Second, to reduce statistical errors, all algorithms were executed 25 times independently on each benchmark function and averaged as a result. To ensure that we could describe the results of the experiments using the mean and variance, we performed the Anderson–Darling test on the results, which is a test applicable to small sample situations. The results of the test demonstrate that the experimental results in all the tables that follow in this paper obey a normal distribution. To discriminate whether the two algorithms were significantly superior or inferior, the Wilcoxon rank sum test with significance level $\alpha = 0.05$ was used as a hypothesis test to compare the algorithms. The symbols "+" "−" and " ≈" were used to indicate that the results of the algorithms were significantly larger, significantly smaller, and close to the algorithm being compared, which are the ones without brackets attached in the tables. To compare and rank among multiple algorithms, the Friedman test with significance level $\alpha = 0.05$ and the Bergmann–Hommel posterior test were used [53].



(**a**)



(**b**)

Figure 5. The trade-off between MSE and time on 10D ellipsoid and rastrigin. (**a**) Ellipsoid. (**b**) Rastrigin.

4.2. Trade-Off between Time Cost and Optimization Accuracy

Before comparison with other algorithms, the trade-off between the number of agents and the number of RBFN trainings is considered so that WDDEA-DBC can achieve a high optimization accuracy with low runtime. Both values are critical to the performance of the algorithm: if the number of agents is too small, the final result will not be stable; if the number of RBFN trainings is too small, the global optimal solution will not be found no matter how long the search time is; and if both values are too large, expensive time overheads will be incurred. As shown in Figure 6, except for the Griewank function which fluctuates at $T = 20$, all the other four benchmark functions obtain better search results as T increases; thus, it can be concluded that increasing T can improve the search accuracy.

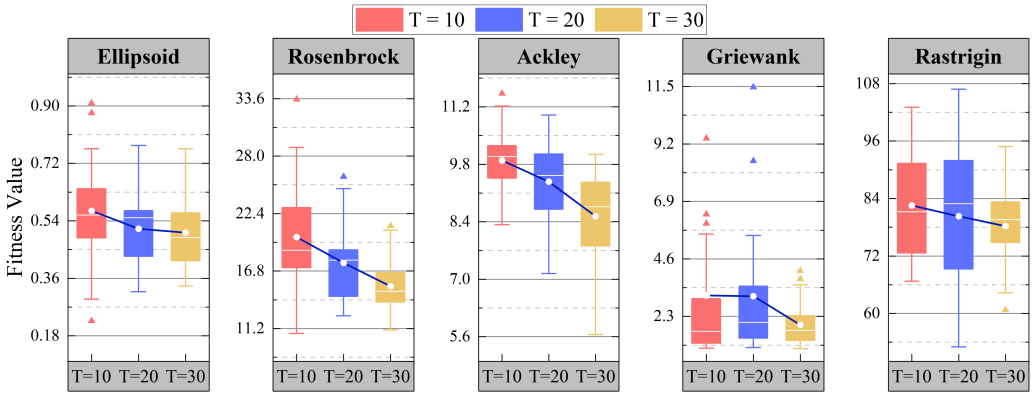


Figure 6. The fitness of five benchmarks with 10/20/30 surrogates.

First, the time overheads of WDDEA-DBC with ten surrogates, trained 250 times, are tested on the benchmark function. For convenience, the number of surrogates is denoted by T and the number of training times is denoted by n in the following. We used the single-peaked function Ellipsoid and the multi-peaked function Rastrigin as the test functions for this part, considering their time costs (in seconds) for different T and n. In addition, BDDEA-LDG was considered a benchmark in the Friedman test and Bergmann Hommel posterior (significance level = 0.05), as it is a valid DDEA that also draws on integrated learning and data generation methods.

As shown in Table 2, WDDEA-DBC (T = 10, n = 250) performs similarly to BDDEA in terms of time overheads according to the p-value of the Friedman test. To reduce the time, the number of training times was first considered to be reduced. Lowering n from 250 to 100 will reduce the time to a certain extent, but again, there will be a loss of optimization accuracy. For insurance reasons, we reduce n and adjust the number of surrogates T upward from 10 to 20 to compensate for the lost training times. It can be seen that the running time of WDDEA-DBC (T = 20, n = 100) is further reduced in the low-dimensional problem, and the overall overheads is significantly lower than that of BDDEA. We attribute this to the advantage of the DBC in which the center and CP generated by clustering can also be used as parameters to the radial basis function of the RBFN to train the surrogate model. Therefore, the DBC process is carried out simultaneously with the RBFN training, thus saving considerable time. However, in the high-dimensional problem with d = 100, WDDEA-DBC (T = 20, n = 100) does not run as fast as WDDEA-DBC (T = 10, n = 250). This is because the time to train the surrogates becomes significantly greater as the dimensionality rises. Therefore, holding n constant and setting T back to 10 reveals a significant reduction in the time overhead, which is nearly double compared to WDDEA-DBC (T = 20, n = 100) for the high-dimensional problem.

To further investigate the impact of this trade-off on the optimization accuracy, Table 3 compares the optimization results under the above three settings. The results show that the choice of T = 10, n = 100 is appropriate when the time cost is considered. First, comparing the cases where T is 10, n = 250 has a slightly better finding accuracy than n = 100 for low-dimensional problems but does not perform as well as n = 100 in higher dimensions. Moreover, n = 100 has a better performance according to the Friedman test ranking. This phenomenon suggests that n = 250 may have been overfitted in these tests. Since WDDEA (T = 10, n = 100) has a faster speed and overall higher accuracy than WDDEA-DBC (T = 10, n = 250), the former is chosen from between the two. Second, comparing the cases where n is 100, the accuracy of T = 20 is better than T = 10, but the difference is not significant and the stability is similar. Considering that the operation speed of T = 10 is significantly better than that of T = 20, the former is a better choice for the algorithm in general. In summary, WDDEA-DBC (T = 10, n = 100) is the best configuration, and the following experiments will also use it.

Table 2. The average of time cost (unit: second) over 25 independent runs of different offline algorithms on Ellipsoid and Rastrigin problems (the best results and the lowest p-values are shown in bold).

Problems	D	WDDEA-DBC (T = 10, n = 100)	WDDEA-DBCA (T = 10, n = 250)	WDDEA-DBC (T = 20, n = 100)	BDDEA-LDG
Ellipsoid	10	$\mathbf{1.15 \times 10^1}(-)$	$1.92 \times 10^1(-)$	$2.14 \times 10^1(\approx)$	2.08×10^1
	30	$\mathbf{2.62 \times 10^1}(-)$	$4.77 \times 10^1(-)$	$5.25 \times 10^1(-)$	8.75×10^1
	50	$\mathbf{4.00 \times 10^1}(-)$	$7.49 \times 10^1(-)$	$7.91 \times 10^1(-)$	1.61×10^2
	100	$\mathbf{7.81 \times 10^1}(-)$	$1.58 \times 10^2(-)$	$1.45 \times 10^2(-)$	6.19×10^2
Rastrigin	10	$\mathbf{1.13 \times 10^1}(-)$	$1.86 \times 10^1(-)$	$2.14 \times 10^1(\approx)$	2.05×10^1
	30	$\mathbf{2.53 \times 10^1}(-)$	$4.80 \times 10^1(-)$	$5.12 \times 10^1(-)$	7.59×10^1
	50	$\mathbf{4.03 \times 10^1}(-)$	$7.52 \times 10^1(-)$	$7.82 \times 10^1(-)$	1.61×10^2
	100	$\mathbf{7.74 \times 10^1}(-)$	$1.58 \times 10^2(-)$	$1.44 \times 10^2(-)$	6.32×10^2
$+/\approx/-$		0/0/8	0/0/8	0/2/6	NA
Average-Ranking		1	2.25	3	3.75
Adjusted p-value		**0.0010**	**0.0926**	0.6330	NA

Table 3. The comparisons on optimization results between variants of the proposed algorithm with different settings (the best results and the lowest p-values are shown in bold).

Problem	D	WDDEA-DBC (T = 10, n = 100)	WDDEA-DBC (T = 20, n = 100)	WDDEA-DBC (T = 10, n = 250)
Ellipsoid	10	0.57 ± 0.16	$0.52 \pm 0.11(\approx)$	$\mathbf{0.45 \pm 0.11}(-)$
	30	0.28 ± 0.14	$\mathbf{0.23 \pm 0.13}(\approx)$	$0.25 \pm 0.15(\approx)$
	50	0.28 ± 0.16	$\mathbf{0.23 \pm 0.13}(\approx)$	$0.30 \pm 0.19(\approx)$
	100	**0.23 ± 0.09**	$0.26 \pm 0.18(\approx)$	$0.47 \pm 0.17(+)$
Rastrigin	10	82.55 ± 11.04	$80.31 \pm 14.58(\approx)$	$\mathbf{79.98 \pm 12.02}(\approx)$
	30	57.33 ± 24.40	$\mathbf{51.42 \pm 18.96}(-)$	$64.67 \pm 14.32(+)$
	50	58.20 ± 21.99	$\mathbf{50.61 \pm 21.75}(-)$	$68.45 \pm 21.76(+)$
	100	52.11 ± 22.44	$\mathbf{50.63 \pm 25.57}(\approx)$	$113.70 \pm 46.29(+)$
$+/\approx/-$		NA	0/6/2	4/3/1
Average Ranking		2.25	1.375	2.375
Adjusted p-value		**NA**	0.1869	0.9000

4.3. Comparisons with Traditional Methods and Data-Driven Evolutionary Algorithms

In this section, WDDEA-DBC is compared with other algorithms. The comparison includes, in addition to the three previously mentioned DDEAs, a GA with SBX (denoted as GA-SBX). This GA-SBX has the same configuration as BDDEA-LDG, with the difference that GA-SBX uses only real FEs seeking, while BDDEA-LDG is driven by data and surrogate models. Since GA-SBX does not include data generation methods, it uses fewer FEs compared to DDEA for the same sampling numbers. Therefore, we expand the sampling number of GA-SBX to 550-D, which can compensate for its disadvantages while highlighting the advantages of WDDEA-DBC.

The comparison results in Table 4 were analyzed using the Friedman test and Bergmann–Hommel post hoc test (significance level = 0.05), where the control method was WDDEA-DBC. The results show that WDDEA-DBC is efficient. First, in comparison with DDEA (both sampling numbers are $11 - D$), WDDEA-DBC is significantly better than and significantly worse than BDDEA-LDG and DDEA-SE on 16 problems, respectively, and it is significantly better than DDEA-SE overall. Regarding the state-of-the-art offline DDEA algorithm TT-DDEA, according to Wilcoxon's rank sum test, WDDEA-DBC significantly outperforms it on fifteen problems, is close in performance on one problem, and is significantly worse than it is on four problems. The experiments show that WDDEA-DBC performs more generally only on Ackley in 10 and 30 dimensions and Griewank and Rastrigin in 10 dimensions, and it performs better in the remaining cases. This result illustrates that WDDEA-DBC has better performance than the other DDEAs on these problems in general. Second, in

the comparison with GA-SBX, WDDEA is significantly better, close to, and significantly worse than it on thirteen, two, and five problems, respectively, and it is clear from the test p-value of $0.047 < 0.05$ that WDDEA-DBC of $11 - D$ performs better than GA-SBX of 550-D overall, i.e., WDDEA-DBC uses only 2% of the FES used by the GA-SBX, yet it produces significantly better results, which illustrates the importance of the DBC. The impact of the DBC will be introduced in the next section. In general, on the one hand, WDDEA stands out in comparison with other representative algorithms and performs very well in terms of both the accuracy and time overhead of the search; on the other hand, WDDEA may be less suitable for the optimization of certain problems, such as low-dimensional multi-peaked functions (e.g., Ackley and Griewank).

Table 4. The comparisons on optimization results between the proposed algorithm and other representative DDEAs (the best results and the lowest p-values are shown in bold).

Problem	D	WDDEA-DBC (11d Offline Data)	BDDEA-LDG (11d Offline Data)	DDEA-SE (11d Offline Data)	GA-SBX (550 Online Data)	TT-DDEA (11d Offline Data)
Ellipsoid	10	**0.57 ± 0.16**	1.01 ± 0.40(+)	1.02 ± 0.49(+)	1.15 ± 0.39(+)	1.20 ± 0.70(+)
	30	**0.28 ± 0.14**	6.66 ± 2.09(+)	5.09 ± 1.30(+)	8.55 ± 2.15(+)	2.80 ± 1.00(+)
	50	**0.28 ± 0.16**	13.10 ± 3.19(+)	15.10 ± 4.63(+)	13.10 ± 2.52(+)	9.30 ± 3.00(+)
	100	**0.23 ± 0.09**	55.50 ± 11.20(+)	312.00 ± 61.30(+)	22.80 ± 6.27(+)	66.90 ± 10.20(+)
Rosenbrock	10	20.12 ± 5.75	35.20 ± 8.58(+)	29.50 ± 5.04(+)	**19.30 ± 5.45**(≈)	32.30 ± 7.30(+)
	30	**43.60 ± 2.71**	50.00 ± 7.35(+)	56.7 ± 5.34(+)	48.20 ± 6.89(+)	56.10 ± 7.10(+)
	50	73.24 ± 3.65	98.10 ± 8.90(+)	84.10 ± 4.05(+)	**61.30 ± 4.08**(−)	81.30 ± 7.20(+)
	100	125.69 ± 5.77	193.00 ± 22.60(+)	265.00 ± 24.80(+)	**108.00 ± 2.96**(−)	150.00 ± 8.60(+)
Ackley	10	9.90 ± 0.84	6.39 ± 0.83(−)	6.40 ± 1.14(−)	8.68 ± 1.26(−)	**6.00 ± 1.00**(−)
	30	6.91 ± 1.12	5.57 ± 0.63(−)	4.83 ± 0.51(−)	6.36 ± 0.77(≈)	**4.60 ± 0.40**(−)
	50	**3.84 ± 0.62**	4.81 ± 0.37(+)	4.82 ± 0.38(+)	4.83 ± 0.35(+)	4.40 ± 0.30(+)
	100	**3.58 ± 0.47**	4.71 ± 0.31(+)	7.27 ± 0.71(+)	4.36 ± 0.33(+)	4.80 ± 0.20(+)
Griewank	10	3.14 ± 3.34	1.29 ± 0.13(−)	1.31 ± 0.15(−)	1.74 ± 0.30(−)	**1.30 ± 0.30**(−)
	30	**1.18 ± 0.39**	1.37 ± 0.10(+)	1.34 ± 0.07(+)	3.52 ± 0.89(+)	1.20 ± 0.10(≈)
	50	**1.17 ± 0.51**	1.42 ± 0.08(+)	1.94 ± 0.02(+)	3.05 ± 0.53(+)	1.60 ± 0.10(+)
	100	**1.15 ± 0.23**	1.80 ± 0.23(+)	18.10 ± 2.12(+)	2.93 ± 0.53(+)	4.50 ± 0.60(+)
Rastrigin	10	82.55 ± 11.04	65.10 ± 29.60(−)	65.90 ± 1.89(−)	**47.3 ± 7.72**(−)	59.40 ± 21.70(−)
	30	**57.33 ± 24.40**	146.00 ± 43.40(+)	185.00 ± 16.10(+)	219.00 ± 11.50(+)	83.10 ± 18.50(+)
	50	**58.20 ± 21.99**	190.00 ± 31.80(+)	187.00 ± 30.30(+)	383.00 ± 19.80(+)	112.40 ± 21.20(+)
	100	**52.11 ± 22.44**	405.00 ± 144.00(+)	811.00 ± 82.60(+)	810.00 ± 22.70(+)	300.80 ± 52.80(+)
$+/\approx/-$		NA	16/0/4	16/0/4	13/2/5	15/1/4
Average Ranking		1.9	3.2	3.9	3.3	2.65
Adjusted p-value		NA	**0.12909206**	**0.001**	**0.04703911**	0.55406

Figure 7 visually compares the performance of the four DDEAs in Ellipsoid and Rastrigin of different dimensions in terms of optimization accuracy and time cost. From Figure 7a,b, we can see that the accuracy of BDDEA-LDG, DDEA-SE and TT-DDEA decreases as the dimension increases, while WDDEA-DBC achieves a higher accuracy, which indicates that WDDEA-DBC has advantages that other DDEAs cannot achieve when dealing with high-dimensional data. In addition, from Figure 7c,d, it can be seen that the running speed of WDDEA-DBC is significantly less affected by the high-dimensional data than the other three, so it is able to save time overheads in the case of large data amounts.

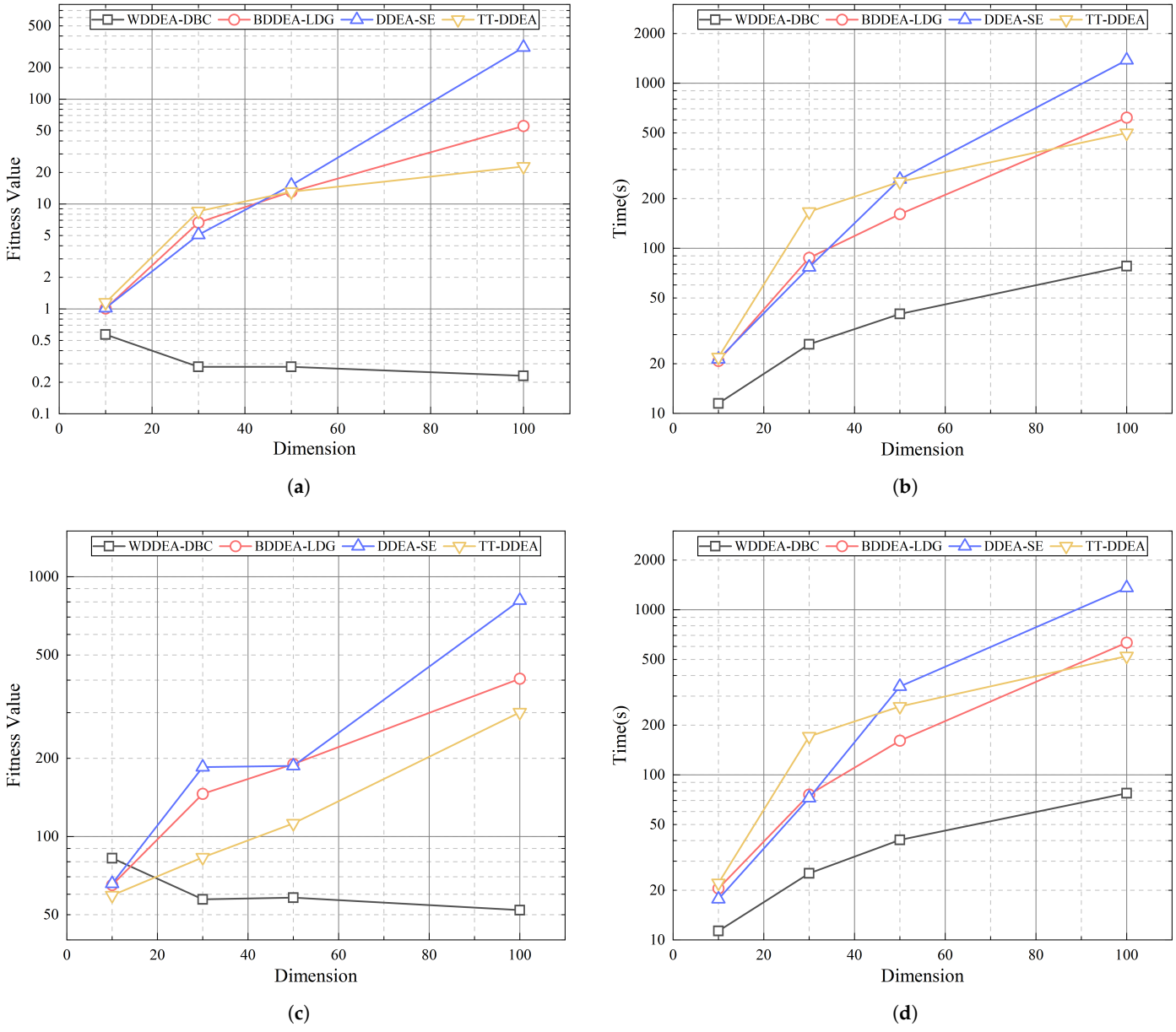


Figure 7. The fitness and time of the Ellipsoid function and the Rastrigin function change with dimension on different algorithms. (**a**) Fitness of the Ellipsoid function. (**b**) Time of the Ellipsoid function. (**c**) Fitness of the Rastrigin function. (**d**) Time of the Rastrigin function.

4.4. Comparisons of Different Variants of Bagging Methods

In Section 3.3, the effectiveness of W-bagging-MSE (10) compared to bagging (9) is demonstrated by mathematical derivation in Equations (11)–(27). This section further illustrates the superiority of W-bagging through numerical experiments. In addition to bagging, the comparisons are W-bagging-RMSE with $1/RMSE$ as the weight and W-bagging-Fitness with $1/Fitness$ as the weight. Their expressions are shown in Equations (28) and (29):

$$\hat{g}_3(\boldsymbol{x}) = \sum \frac{g_i(\boldsymbol{x})}{RMSE_i} / \sum \frac{1}{RMSE_i}, \tag{28}$$

$$\hat{g}_4(\boldsymbol{x}) = \sum \frac{g_i(\boldsymbol{x})}{Fitness_i} / \sum \frac{1}{Fitness_i}. \tag{29}$$

Notably, where Equation (28) approximates Equation (10), the difference is that the weights of Equation (28) are based on the RMSE, which is widely used to standardize the unit's MSE measurements [54]. At the same time, the motivation for proposing Equation (29) is to evaluate the merits of the model directly based on fitness to obtain the aggregated model with the strongest optimization capability. Table 5 shows the optimization results under the four bagging methods in the four dimensions of the five benchmark functions. To ensure fairness, all algorithms are configured equally. The comparison results were analyzed using the Friedman test and Bergmann–Hommel posterior test (significance level = 0.05), where the control method was bagging. W-bagging-MSE performed the best among the four methods according to the overall ranking, and it can be inferred from the *p*-value that it significantly outperformed bagging. In addition, all the frameworks of W-bagging work better than bagging, which further illustrates the feasibility of W-bagging. In comparison with other W-bagging methods, W-bagging-MSE achieves better results on more problems, so it can be concluded that W-bagging with a weight of $1/MSE$ has a higher accuracy when finding the best outcome.

Table 5. The comparisons on optimization results between variants of W-bagging and bagging (the best results and the lowest *p*-values are shown in bold).

Problem	D	Bagging	W-Bagging (MSE)	W-Bagging (RMSE)	W-Bagging (Fitness)
Ellipsoid	10	$\mathbf{3.22 \times 10^{-1}}$	$5.72 \times 10^{-1}(+)$	$5.65 \times 10^{-1}(+)$	$4.24 \times 10^{-1}(+)$
	30	3.65×10^{-1}	$2.84 \times 10^{-1}(-)$	$3.85 \times 10^{-1}(+)$	$\mathbf{2.65 \times 10^{-1}(-)}$
	50	2.90×10^{-1}	$\mathbf{2.77 \times 10^{-1}(-)}$	$2.90 \times 10^{-1}(\approx)$	$3.17 \times 10^{-1}(+)$
	100	2.68×10^{-1}	$\mathbf{2.33 \times 10^{-1}(-)}$	$2.66 \times 10^{-1}(\approx)$	$2.56 \times 10^{-1}(-)$
Rosenbrock	10	2.57×101	$2.01 \times 10^{1}(-)$	$\mathbf{1.97 \times 10^{1}(-)}$	$2.22 \times 10^{1}(-)$
	30	5.26×10^{1}	$\mathbf{4.36 \times 10^{1}(-)}$	$4.62 \times 10^{1}(-)$	$4.96 \times 10^{1}(-)$
	50	7.72×10^{1}	$7.32 \times 10^{1}(-)$	$7.59 \times 10^{1}(-)$	$\mathbf{6.99 \times 10^{1}(-)}$
	100	1.62×10^{2}	$\mathbf{1.26 \times 10^{2}(-)}$	$1.52 \times 10^{2}(-)$	$1.78 \times 10^{2}(+)$
Ackley	10	$\mathbf{8.47 \times 10^{0}}$	$9.90 \times 10^{0}(+)$	$9.56 \times 10^{0}(+)$	$1.09 \times 10^{1}(+)$
	30	7.19×10^{0}	$6.91 \times 10^{0}(-)$	$\mathbf{6.39 \times 10^{0}(-)}$	$7.02 \times 10^{0}(-)$
	50	3.96×10^{0}	$3.84 \times 10^{0}(-)$	$3.91 \times 10^{0}(\approx)$	$\mathbf{3.56 \times 10^{0}(-)}$
	100	$\mathbf{3.55 \times 10^{0}}$	$3.58 \times 10^{0}(\approx)$	$3.98 \times 10^{0}(+)$	$3.76 \times 10^{0}(+)$
Griewank	10	4.17×10^{0}	$\mathbf{3.14 \times 10^{0}(-)}$	$3.26 \times 10^{0}(-)$	$3.32 \times 10^{0}(-)$
	30	2.98×10^{0}	$\mathbf{1.18 \times 10^{0}(-)}$	$1.30 \times 10^{0}(-)$	$1.65 \times 10^{0}(-)$
	50	1.38×10^{0}	$1.17 \times 10^{0}(-)$	$\mathbf{1.16 \times 10^{0}(-)}$	$1.51 \times 10^{0}(-)$
	100	1.23×10^{0}	$\mathbf{1.15 \times 10^{0}(-)}$	$\mathbf{1.15 \times 10^{0}(-)}$	$1.26 \times 10^{0}(\approx)$
Rastrigin	10	9.64×10^{1}	$\mathbf{8.26 \times 10^{1}(-)}$	$8.69 \times 10^{1}(-)$	$1.11 \times 10^{2}(+)$
	30	6.89×10^{1}	$\mathbf{5.73 \times 10^{1}(-)}$	$5.95 \times 10^{1}(-)$	$5.90 \times 10^{1}(-)$
	50	6.56×10^{1}	$5.82 \times 10^{1}(-)$	$\mathbf{5.32 \times 10^{1}(-)}$	$5.87 \times 10^{1}(-)$
	100	6.02×10^{1}	$\mathbf{5.21 \times 10^{1}(-)}$	$5.49 \times 10^{1}(-)$	$5.70 \times 10^{1}(-)$
$+/\approx/-$		NA	2/1/17	4/3/13	6/1/13
Average Ranking		3.3	1.73	2.3	2.67
Adjusted *p*-value		NA	**0.001**	**0.09184123**	0.76955915

4.5. Influence of Data Generation Based on Clustering Strategy

This section examines the optimization results of the algorithm performed 25 times independently for each of the five benchmark functions in 10 dimensions with and without DBC. For a fair comparison, both algorithms have the same configuration and use the same amount ($11 - D$) of offline data for model training, with the only difference being whether the DBC method is used. As shown in Figure 8, WDDEA with DBC achieves a better mean value of outperformance on all four benchmark functions except Ellipsoid, and this result is extremely evident for Rosenbrock and Ackley. For Griewank and Ackley, although the improvement in accuracy is smaller, the stability is improved to some extent. This is due to the DBC's ability to enhance the RBFN's ability to utilize the data and thus

train a more accurate and faster converging agent model. The following visualization illustrates the advantage of the DBC, which is, since the range of l is controlled, that the generated data points are essentially still within the boundary to avoid noise to obtain better optimization results.

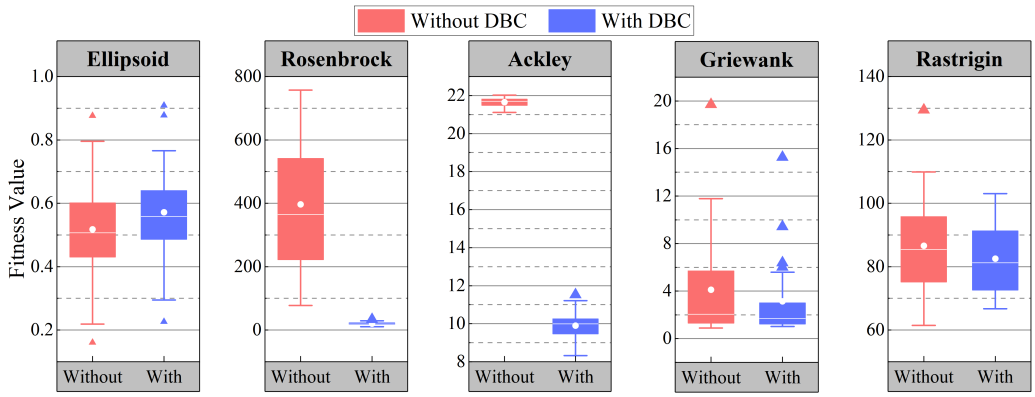


Figure 8. The fitness of five benchmarks with or without DBC.

4.6. Influences of Configuration Settings in Data Generation Based on Clustering

This section investigates the impact of the configuration in the DBC on performance. As selection criteria, the width parameter l of the normal distribution obeyed by the random vectors generated by the DBC and the number of clusters w for performing the DBC may have an impact on the algorithm, and the effects of these two settings are described below.

First, the size of l affects the quality of the generated data. Specifically, if l is too small, it will lead to a lack of heterogeneity between the newly generated data and the original data, and it will not effectively restore the missing information in the original dataset. If l is too large, it will lead to the newly generated data being more volatile and unstable, and the new samples obtained will be unreliable. Therefore, the correct choice of l is crucial. Second, the number of w affects the amount of generated data. For that, it is difficult to calculate and analyze its optimal value, because the w to obtain the best result is often different for different functions. However, an increase in w generally increases the time overheads of the algorithm because it implies an increase in the amount of data.

The heat maps in Figure 9 visualize the result and running time of the optimization search for different combinations of l and w in 10 dimensions, Ellipsoid and Rastrigin. As analyzed above, the effect of w on the search results for a given value of l does not show a clear pattern but shows an overall positive correlation over time. For a given value of w, both functions obtain the optimal solution when l is taken as 0.1. The above analyses illustrate that the values of l and w have a significant impact on the performance of the algorithm, and we take this part as the future improvement direction of the algorithm.

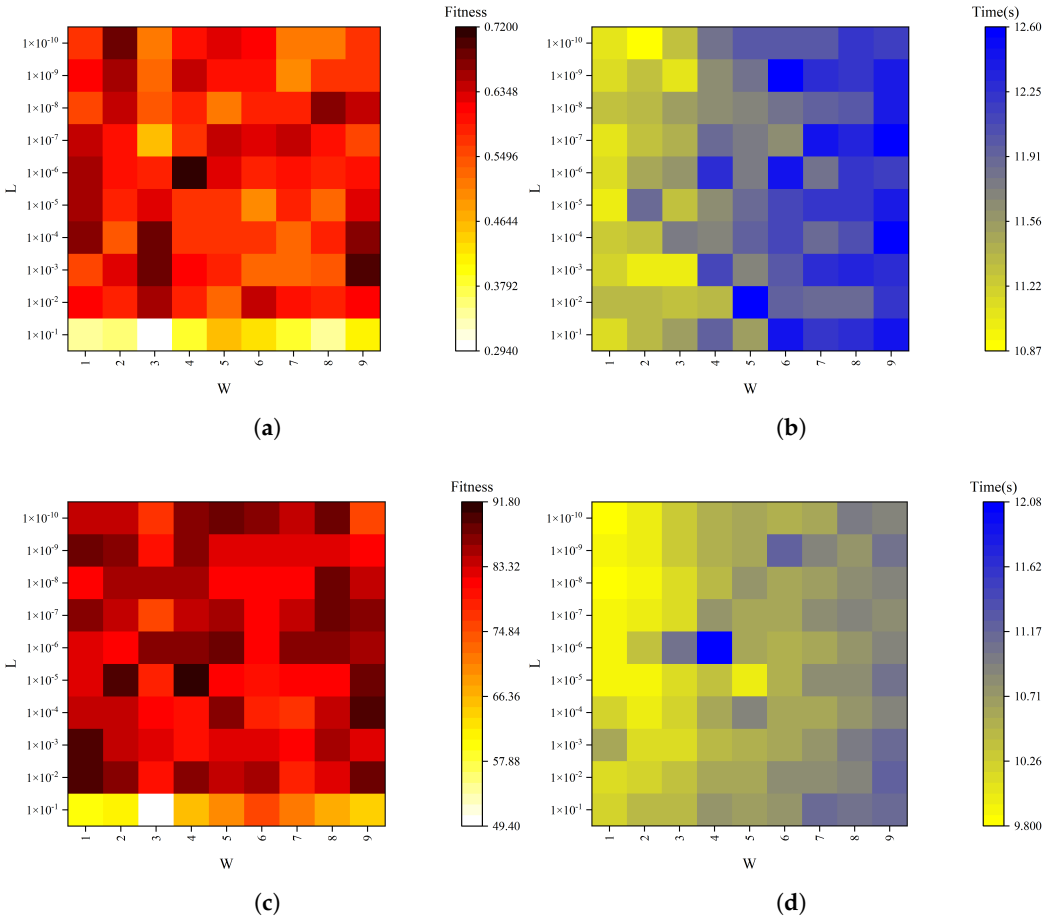


Figure 9. The fitness and time of the Ellipsoid function and the Rastrigin function change with coefficients of DBC. (**a**) Fitness of the Ellipsoid function. (**b**) Time of the Ellipsoid function. (**c**) Fitness of the Rastrigin function. (**d**) Time of the Rastrigin function.

5. Conclusions

To address the three problems suffered by the widely studied offline DDEAs, i.e., the redundancy of agent models, poor ability to utilize data, and unsatisfactory solutions to overcome data deficiency, this paper proposes a new WDDEA-DBC algorithm, which is able to achieve good optimization results with less time cost by means of a new model management strategy and a new data generation method. Its time advantage is attributed to the good performance and stability of W-bagging. The weighted average property of W-bagging based on MSE allows it to make better use of data and evaluate surrogates so that it can use fewer surrogates to obtain better optimization results. Moreover, the experimental results of Section 4.4 show that the potential of the DBC has not yet been fully exploited, and more research work should be conducted in the future. In addition, the proposed DBC also gives a better optimization effect of this algorithm, which has been confirmed by a large number of numerical experiments in Section 4. The results of numerical experiments also show that the potential of the DBC has not been fully exploited, so more research work can be carried out on the DBC in the future.

To demonstrate the good performance of the proposed algorithm, we compared WDDEA-DBC with some representative DDEAs and some traditional algorithms under the same algorithm configuration and computer configuration conditions using a number of commonly used benchmark functions. The numerical experimental results show the superiority of WDDEA-DBC and reveal another advantage of it: namely, it is less affected by the increasing problem dimensions, and higher problem dimensions better reflect this advantage.

For example, as shown in Tables 4 and 5, the results of the WDDEA-DBC optimization of the Ackley function are not very satisfactory, and it may not be suitable for optimizing such problems. Another example is that the time overheads of the algorithm increase linearly as the number of surrogate models increases, which prevents it from using more surrogate models in situations where the time overheads are demanding.

In future research work, we think there are several directions to consider:

1. Migrate the W-bagging and DBC of this algorithm to the domain of online DDEAs [3] and hope to propose a new online DDEA by combining online learning [55], stream data processing [56], etc.
2. This algorithm has a good optimization effect in the case of RBFN as a surrogate model and FPA as an optimization algorithm, so it could be combined with more surrogate models and more optimization algorithms (e.g., CS [57] and BA [58]) in the future to achieve better performance.
3. Due to space limitations, the performance of this algorithm in solving practical problems cannot be analyzed in this paper, and we hope to cover this aspect more in future work.

Author Contributions: Conceptualization, Z.G. and X.Z.; methodology, Z.G.; software, Z.G. and S.L.; validation, Z.G., S.L. and R.S.; formal analysis, Z.G.; investigation, S.L.; resources, R.S.; data curation, R.S.; writing—original draft preparation, S.L.; writing—review and editing, Z.G. and X.Z.; visualization, Z.G.; supervision, X.Z.; project administration, X.Z.; funding acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Shenzhen Natural Science Fund (the Stable Support Plan Program GXWD20220811170436002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yang, X.S. Flower pollination algorithm for global optimization. In *Unconventional Computation and Natural Computation, Proceedings of the International Conference on Unconventional Computing and Natural Computation, Orléans, France, 3–7 September 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 240–249.
2. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
3. Jin, Y.; Wang, H.; Chugh, T.; Guo, D.; Miettinen, K. Data-driven evolutionary optimization: An overview and case studies. *IEEE Trans. Evol. Comput.* **2019**, *23*, 442–458. [CrossRef]
4. Wang, H.; Jin, Y.; Sun, C.; Doherty, J. Offline data-driven evolutionary optimization using selective surrogate ensembles. *IEEE Trans. Evol. Comput.* **2019**, *23*, 203–216. [CrossRef]
5. Li, J.Y.; Zhan, Z.H.; Wang, C.; Jin, H.; Zhang, J. Boosting data-driven evolutionary algorithm with localized data generation. *IEEE Trans. Evol. Comput.* **2020**, *24*, 923–937. [CrossRef]
6. Wang, H.; Jin, Y.; Jansen, J.O. Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system. *IEEE Trans. Evol. Comput.* **2016**, *20*, 939–952. [CrossRef]
7. Mazumdar, A.; Chugh, T.; Hakanen, J.; Miettinen, K. An interactive framework for offline data-driven multiobjective optimization. In *Bioinspired Optimization Methods and Their Applications, Proceedings of the International Conference on Bioinspired Methods and Their Applications, Brussels, Belgium, 19–20 November 2020*; Springer: Cham, Switerlands, 2020; pp. 97–109.
8. Wang, M.; Shan, Y.; Xu, F. Offline data-driven evolutionary optimization algorithm using k-fold cross. In *Advances in Swarm Intelligence, Proceedings of the International Conference on Sensing and Imaging*; Springer: Cham, Switerlands, 2022; pp. 305–316.
9. Rashki, M.; Azarkish, H.; Rostamian, M.; Bahrpeyma, A. Classification correction of polynomial response surface methods for accurate reliability estimation. *Struct. Saf.* **2019**, *81*, 101869. [CrossRef]
10. Van Beers, W.C.; Kleijnen, J.P. Kriging interpolation in simulation: A survey. In Proceedings of the 2004 Winter Simulation Conference, Washington, DC, USA, 5–8 December 2004; p. 1.

11. She, C.; Wang, Z.; Sun, F.; Liu, P.; Zhang, L. Battery aging assessment for real-world electric buses based on incremental capacity analysis and radial basis function neural network. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3345–3354. [CrossRef]
12. Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* **2020**, *408*, 189–215. [CrossRef]
13. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef]
14. Montazer, G.A.; Giveki, D.; Karami, M.; Rastegar, H. Radial basis function neural networks: A review. *Comput. Rev. J.* **2018**, *1*, 52–74.
15. Huang, F.; Xie, G.; Xiao, R. Research on ensemble learning. In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence, Washington, DC, USA, 7–8 November 2009; Volume 3, pp. 249–252.
16. Huang, P.; Wang, H.; Jin, Y. Offline data-driven evolutionary optimization based on tri-training. *Swarm Evol. Comput.* **2021**, *60*, 100800. [CrossRef]
17. Kourakos, G.; Mantoglou, A. Development of a multi-objective optimization algorithm using surrogate models for coastal aquifer management. *J. Hydrol.* **2013**, *479*, 13–23. [CrossRef]
18. Li, X.; Hou, J.M.; Chai, J.; Du, Y.E.; Han, H.; Yang, S.X.; Gao, X.J.; Yang, X. An online data-driven evolutionary algorithm-based optimal design of urban stormwater-drainage systems. *J. Irrig. Drain. Eng.* **2022**, *148*, 04022041. [CrossRef]
19. Wang, H.D.; Zhang, Q.F.; Jiao, L.C.; Yao, X. Regularity model for noisy multiobjective optimization. *IEEE Trans. Cybern.* **2016**, *46*, 1997–2009. [CrossRef]
20. Chugh, T.; Chakraborti, N.; Sindhya, K.; Jin, Y. A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem. *Mater. Manuf. Process.* **2017**, *32*, 1172–1178. [CrossRef]
21. Jin, Y.C.; Sendhoff, B. Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Genetic and Evolutionary Computation—GECCO 2004, Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, WA, USA, 26–30 June 2004*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3102, pp. 688–699.
22. Guo, D.; Chai, T.; Ding, J.; Jin, Y. Small data driven evolutionary multi-objective optimization of fused magnesium furnaces. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–8.
23. Dua, M.; Gupta, R.; Khari, M.; Crespo, R.G. Biometric iris recognition using radial basis function neural network. *Soft Comput.* **2019**, *23*, 11801–11815. [CrossRef]
24. Ning, Y.; Wang, J.; Han, H.; Tan, X.; Liu, T. An optimal radial basis function neural network enhanced adaptive robust Kalman filter for GNSS/INS integrated systems in complex urban areas. *Sensors* **2018**, *18*, 3091. [CrossRef]
25. Tkachenko, R.; Kutucu, H.; Izonin, I.; Doroshenko, A.; Tsymbal, Y. Non-iterative Neural-like Predictor for Solar Energy in Libya. *ICTERI* **2018**, *1*, 35–45.
26. Izonin, I.; Tkachenko, R.; Dronyuk, I.; Tkachenko, P.; Gregus, M.; Rashkevych, M. Predictive modeling based on small data in clinical medicine: RBF-based additive input-doubling method. *Math. Biosci. Eng.* **2021**, *18*, 2599–2613. [CrossRef] [PubMed]
27. Izonin, I.; Tkachenko, R.; Fedushko, S.; Koziy, D.; Zub, K.; Vovk, O. RBF-Based Input Doubling Method for Small Medical Data Processing. In *Advances in Artificial Systems for Logistics Engineering, Proceedings of the The International Conference on Artificial Intelligence and Logistics Engineering, Kyiv, Ukraine, 22–24 January 2021*; Springer: Cham, Switerlands, 2021; pp. 23–31.
28. Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 881–892. [CrossRef]
29. Gomm, J.; Yu, D. Selecting radial basis function network centers with recursive orthogonal least squares training. *IEEE Trans. Neural Netw.* **2000**, *11*, 306–314. [CrossRef] [PubMed]
30. Chen, S.; Cowan, C.; Grant, P. Orthogonal least-squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Netw.* **1991**, *2*, 302–309. [CrossRef] [PubMed]
31. Chen, S.; Billings, S.; Grant, P. Recursive hybrid algorithm for nonlinear-system identification using radial basis function networks. *Int. J. Control* **1992**, *55*, 1051–1070. [CrossRef]
32. Walczak, B.; Massart, D.L. The radial basis functions—Partial least squares approach as a flexible non-linear regression technique. *Anal. Chim. Acta* **1996**, *331*, 177–185. [CrossRef]
33. Garg, A.; Tai, K. Comparison of statistical and Machine Learning methods in modelling of data with multicollinearity. *Int. J. Model. Identif. Control* **2013**, *18*, 295–312. [CrossRef]
34. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
35. Freund, Y. Boosting a weak learning algorithm by majority. *Inf. Comput.* **1995**, *121*, 256–285. [CrossRef]
36. Speiser, J.L.; Miller, M.E.; Tooze, J.; Ip, E. A comparison of random forest variable selection methods for classification prediction modeling. *Expert Syst. Appl.* **2019**, *134*, 93–101. [CrossRef]
37. Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; Herrera, F. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C-Appl. Rev.* **2011**, *42*, 463–484. [CrossRef]
38. Shahraki, A.; Abbasi, M.; Haugen, Ø. Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103770. [CrossRef]
39. Chen, T.Q.; Guestrin, C.; Machinery, A.C. XGBoost: A sacalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.

40. Ibrahem Ahmed Osman, A.; Najah Ahmed, A.; Chow, M.F.; Feng Huang, Y.; El-Shafie, A. Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia. *Ain Shams Eng. J.* **2021**, *12*, 1545–1556. [CrossRef]
41. Qiu, Y.; Zhou, J.; Khandelwal, M.; Yang, H.; Yang, P.; Li, C. Performance evaluation of hybrid WOA-XGBoost, GWO-XGBoost and BO-XGBoost models to predict blast-induced ground vibration. *Eng. Comput.* **2022**, *38*, 4145–4162. [CrossRef]
42. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
43. Hartigan, J.A.; Wong, M.A. Algorithm as 136: A k-means clustering algorithm. *J. R. Stat. Soc. Ser. C (Appl. Stat.)* **1979**, *28*, 100–108. [CrossRef]
44. Li, K.J.; Xue, W.P.; Tan, G.; Denzer, A.S. A state of the art review on the prediction of building energy consumption using data-driven technique and evolutionary algorithms. *Build. Serv. Eng. Res. Technol.* **2020**, *41*, 108–127. [CrossRef]
45. Huang, D.S. Radial basis probabilistic neural networks: Model and application. *Int. J. Pattern Recognit. Artif. Intell.* **1999**, *13*, 1083–1101. [CrossRef]
46. He, K.M.; Zhang, X.Y.; Ren, S.Q.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
47. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
48. Dietterich, T.G. Ensemble methods in Machine Learning. In *International Workshop on Multiple Classifier Systems*; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1857, pp. 1–15.
49. Dietterich, T.G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.* **2000**, *40*, 139–157. [CrossRef]
50. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; National University of Defense Technology: Changsha, China; Kyungpook National University: Daegu, Republic of Korea; Nanyang Technological University: Singapore, 2017.
51. Guo, Z.; Suo, R.; Zhang, X. An improved flower pollination algorithm based on logistic chaotic mapping and natural mutation. In Proceedings of the 7th International Conference on Big Data and Computing, Shenzhen, China, 27–29 May 2022; Association for Computing Machinery: New York, NY, USA, 2022; ICBDC '22; pp. 37–44.
52. Helton, J.C.; Davis, F.J. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliab. Eng. Syst. Saf.* **2003**, *81*, 23–69. [CrossRef]
53. Derrac, J.; Garcia, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
54. Chicco, D.; Warrens, M.J.; Jurman, G. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Comput. Sci.* **2021**, *7*, e623. [CrossRef] [PubMed]
55. Mocanu, E.; Mocanu, D.C.; Nguyen, P.H.; Liotta, A.; Webber, M.E.; Gibescu, M.; Slootweg, J.G. On-line building energy optimization using deep reinforcement learning. *IEEE Trans. Smart Grid* **2018**, *10*, 3698–3708. [CrossRef]
56. Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; Guizani, M. Deep learning for IOT big data and streaming analytics: A survey. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 2923–2960. [CrossRef]
57. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
58. Yang, X.S. A new metaheuristic bat-inspired algorithm. In Proceedings of the International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Granada, Spain, 12–14 May 2010; Volume 284, pp. 65–74.