

# Making use of observable parameters in evolutionary dynamic optimization

Tao Zhu<sup>a</sup>, Wenjian Luo<sup>b,1,\*</sup>, Chenyang Bu<sup>c</sup>, Huansheng Ning<sup>d</sup>

<sup>a</sup>School of Computer, University of South China, Hengyang, Hunan 421001, China

<sup>b</sup>Anhui Province Key Laboratory of Software Engineering in Computing and Communication, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China

<sup>c</sup>School of Computer and Information, Hefei University of Technology, Hefei, Anhui, China

<sup>d</sup>School of Computer and Communication Engineering, University of Science & Technology Beijing, Beijing, China

## ARTICLE INFO

### Article history:

Received 12 January 2019

Revised 4 October 2019

Accepted 14 October 2019

Available online 14 October 2019

### Keywords:

Evolutionary dynamic optimization  
gray-box optimization problems  
learning  
observable parameters  
benchmark problems

## ABSTRACT

In evolutionary dynamic optimization (EDO), most of the existing studies have assumed that dynamic optimization problems are black boxes. However, for many real-world problems, the dynamic parameters that cause the problems to change are observable. However, determining the utility of these parameters in improving optimization performance has not yet been well studied. In this paper, we propose and compare three strategies for this task: rote learning, fitting data with a feedforward neural network and an ensemble strategy. The main idea of these strategies is to learn the relation between the observable parameters and the optimal solutions and then predict new optima once the environment changes. We also propose a set of test cases representing different kinds of characteristics of real-world problems. In the experiments, the proposed strategies are compared with existing methods that do not use observable parameters, and the results validate our proposed strategies.

© 2019 Elsevier Inc. All rights reserved.

## 1. Introduction

In the real world, there are many dynamic optimization problems (DOPs) with changing objectives and/or constraints over time [15,29]. For example, the dispatch optimization of power systems is a kind of DOP because the power demands are constantly changing [43,44]. Solving DOPs is challenging. The algorithms not only need to locate the optimal solution in the current environment while taking future influences into account but also must quickly respond to problem changes and track the movement of the optimal solutions. Evolutionary algorithms (EAs) and similar population-based stochastic search methods are effective approaches to DOPs due to their robustness and adaptability [31]. Using such methods to solve DOPs is called evolutionary dynamic optimization (EDO) [15,29].

To the best of our knowledge, most of the existing general purpose EDO studies have considered DOPs as black boxes [23,29]. EDO algorithms typically work only according to the evolving solutions and their time-varying fitness (see Fig. 1). Changes in the problems are usually only detected by re-evaluating certain solutions.

\* Corresponding author.

E-mail addresses: [tzhu@usc.edu.cn](mailto:tzhu@usc.edu.cn) (T. Zhu), [wjluo@ustc.edu.cn](mailto:wjluo@ustc.edu.cn) (W. Luo), [chenyangbu@hfut.edu.cn](mailto:chenyangbu@hfut.edu.cn) (C. Bu), [ninghuansheng@ustb.edu.cn](mailto:ninghuansheng@ustb.edu.cn) (H. Ning).

<sup>1</sup> This work is supported by the National Natural Science Foundation of China (No. 61573327), Natural Science Foundation of Hunan Province (2019JJ50499) and the double first class construct program of USC (No. 2017SYL16). (Corresponding author: Wenjian Luo.)

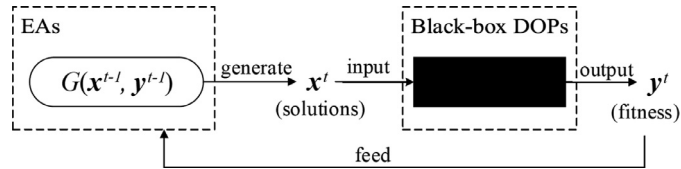


Fig. 1. Application of EAs to black-box DOPs.

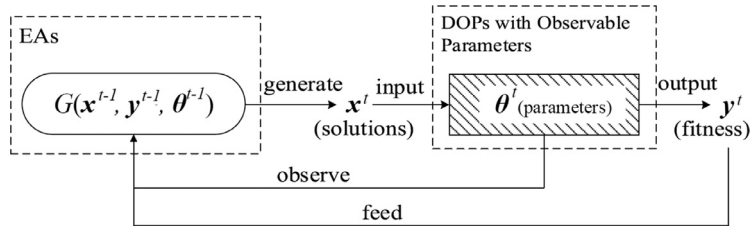


Fig. 2. Application of EAs to DOPs with observable parameters.

However, for many real-world problems, the dynamic parameters that make the problem change could be observable. These parameters include the external environmental factors and the internal system states. For the sake of convenience, all of them are collectively called “parameters”. Please note that “parameters” are different from a problem’s “control/decision variables” that are often called the “solution” of the problem. In this paper, we use “gray-box DOPs” to refer to DOPs with observable parameters.

Some examples of gray-box DOPs are listed below. In dynamic vehicle routing problems, observable parameters include customers’ changing locations and requests [24]. In dynamic optimal power flow, the changing factors (i.e., the power demands at the bus nodes) can also be observed [37,44]. Finally, in dynamic job scheduling problems, the arrival of new tasks, machine breakdowns and changes in economic and financial conditions are all observable. These parameters greatly affect the optimal solutions. If such important data could be utilized to guide the search process properly in EAs, the optimization performance could be enhanced. However, this has not been well studied in the EDO community.

In this paper, we propose to use EAs for gray-box DOPs with the framework in Fig. 2, where  $\theta$  represents the observable parameters. We also propose to use learning methods to leverage observable parameters in EAs. The key idea behind the methods is to learn the relation model between the parameters and optimal solutions from past optimization data and then use the learned model to predict new optimal solutions once the problems change. Specifically, we propose three learning strategies: rote learning (RL), fitting with a feedforward neural network (FNN) and an ensemble strategy (Ens). Rote learning predicts optimal solutions in terms of the similarities between parameters. With the FNN strategy, we train an FNN to estimate the relational model between the parameters and the optimal solution(s), called the parameter-optimum relation for short. It has been proven that multilayer feedforward networks are universal approximators of functions, [13]. The ensemble strategy incorporates the above two strategies.

In addition, we propose a set of numerical test problems in real-valued space to simulate gray-box DOPs, whose dynamic parameters are all observable. The proposed problem set includes 42 instances, which are dynamic variants of static problems from the CEC<sup>2</sup> optimization competition [18,36]. The problem generation method is based on the DOP definitions given by Nguyen [28], and the idea derives from [17,30]. With the proposed generation method, it is easy to construct gray-box DOPs from static optimization problems that exhibit various parameter-optimum relations.

The remainder of the paper is organized as follows. Sections 2 and 3 provide the related backgrounds. The proposed strategies are described in detail in Section 4. Section 5 illustrates the benchmark problems. The experimental settings and results analysis are given in Section 6 and Section 7, respectively. Section 8 concludes the paper and presents plans for future works.

## 2. General purpose test problems and generator

Some general purpose test problems have been designed in the EDO field. Some representative examples include the moving peaks problem[5,41], the XOR benchmark problem generator[40], the generalized dynamic benchmark generator [17], and dynamic constraint optimization problems [6,27]. Moreover, Bosman designed a dynamic time-linkage problem, where the decisions made by algorithms would influence the problem states in the future [4].

To fully describe the common characteristics of DOPs, such as various changing factors, we use Nguyen’s DOP definition [28], where a DOP mainly consists of a *full-description form* and a matching *dynamic driver*. Readers are encouraged to refer to the original definition in [28].

<sup>2</sup> IEEE Congress on Evolutionary Computation.

**Definition 1** (Full-description form [28]). Given a parameterized function  $\widehat{f}_\theta(\mathbf{x})$  with parameters  $\theta$  and a parameter vector set  $\Theta = \{\theta^1, \theta^2, \dots, \theta^n\}$ , the 2-tuple

$$\langle \widehat{f}_\theta(\mathbf{x}), \Theta \rangle \quad (1)$$

is a full description of a finite set of static optimization functions  $\{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})\}$ , such that

$$\begin{aligned} \widehat{f}_\theta(\mathbf{x}) &\xrightarrow{\theta=\theta^1} f_1(\mathbf{x}) \\ &\dots \\ \widehat{f}_\theta(\mathbf{x}) &\xrightarrow{\theta=\theta^n} f_n(\mathbf{x}) \end{aligned} \quad (2)$$

The functions  $f_i(\mathbf{x})$  ( $i = 1, 2, \dots, n$ ) are called *instances* of the full-description form at  $\theta = \theta^i$ . For convenience, we will refer to  $\langle \widehat{f}_\theta(\mathbf{x}), \Theta \rangle$  as  $\widehat{f}$ .

**Definition 2** (Dynamic driver [28]). Given a tuple  $\langle \widehat{f}, \theta^t, t, X_{\widehat{f}}^{G[1,t]} \rangle$ , where  $t$  is a time variable and  $\widehat{f}$  is a full-description form whose parameters are  $\theta^t \in \Theta$  at time  $t$ ,  $X_{\widehat{f}}^{G[1,t]}$  are the past solutions obtained by the algorithm  $G$  during the time period from 1 to  $t$ . We call the function  $D(\theta, X_{\widehat{f}}^{G[1,t]}, t)$  a dynamic driver of  $\widehat{f}$  if

$$\theta^{t+1} = D(\theta^t, X_{\widehat{f}}^{G[1,t]}, t) \in \Theta \quad (3)$$

and  $\theta^{t+1}$  is used as the parameter set of  $\widehat{f}$  at time  $t + 1$ .

A DOP in the real world can be simulated by combining a *full-description form*  $\widehat{f}$  with a suitable *dynamic driver*  $D$ . Driven by  $D$ , the parameters  $\theta$  of  $\widehat{f}$  would change over time, yielding different  $\widehat{f}$  instances at different time steps. In the EDO literature [29], the instance at a certain time is also called the “environment” or “state” of the DOP at that time. The goal of an optimization algorithm is to choose a solution (decision) for the corresponding instance of  $\widehat{f}$  at every time step such that the overall outcome is as good as possible during the optimization time period.

This dynamic driver definition classifies the influences that cause a DOP to change into two types, which are characterized by the current parameter vector  $\theta^t$  and decision influence  $X_{\widehat{f}}^{G[1,t]}$ . Decision influence means that a solution created in the past will affect the DOP’s state in the future. These types of problem are also termed “time-linkage” problems [4]. In contrast, when a DOP is free from the influence of  $X_{\widehat{f}}^{G[1,t]}$ , it is called a “non-time-linkage” DOP.

### 3. EAs for DOPs

The EAs for dynamic optimization can be mainly classified into diversity approaches, memory approaches, multipopulation approaches, prediction approaches and hybrid approaches [19,21,29]. Diversity approaches maintain or introduce population diversity by strategies such as niching [20]. Memory approaches take advantage of the information memorized from past environments to improve the optimization in similar environments [9,45,46]. The idea behind the multipopulation approaches is to divide the population into multiple subpopulations, and each one tracks optima in different promising search areas or with its own strategies [3,6,16,22,32].

Heuristic search algorithms, such as EAs and Tabu Search, have been used to optimize parameters of machine learning models [1]. In contrast, machine learning methods are also employed in heuristic search algorithms to create a meta-model for accelerating search. Prediction approaches for DOPs employ machine learning methods to learn from past optimization records and then provide information to speed up EAs.

Some prediction approaches predict the moving trajectory of the optimal solutions based on a sequence of best solutions obtained in the past [12,26,34,35,42]. In [40], a learning method in a binary search space was proposed, and the main idea was to learn the probability distribution of optima from the past best solutions and then generate new solutions based on the distribution in the new environments. In addition, some learning and prediction approaches have been devoted to time-deception problems [4,27]. The idea is to learn a time-dependent mathematical expression for the problems and then predict the influence of a solution in the future. Some approaches predict when the next change will happen in order to enhance the EAs’ adaptability to problem changes [35]. These learning and prediction methods are designed from black-box problems without observable parameters and thus are all different from the proposed strategies in this paper.

Additionally, in [33], a learning scheme, named *Abstract Memory*, was proposed for continuous search spaces. Initially, the search space is partitioned into rectangular cells. As the population evolves, it counts the number of appearances of the best solutions in each cell, and then a probability distribution of the optimal solutions over the cells is approximated. Once the environment changes, new individuals are sampled from that distribution and added into the population.

### 4. EAs learning from observable parameters

As seen in Definition 1, the specific state of a DOP is entirely determined by the value of its parameter vector  $\theta$ . Therefore, changes in a DOP are actually changes in  $\theta$ . If the value of  $\theta$  is the same at two specific change steps,  $t_0$  and  $t_1$ , the system

states are also the same, and consequently, the optimal solutions at  $t_0$  and  $t_1$  are the same. It can be concluded that at any change step, the optimal solutions are determined by the value of  $\theta$ . Therefore, there exists a relational model between the parameters and the optimal solutions. If this relational model is available, whenever  $\theta$  changes, it is likely to predict the new optimal solutions using the model, which avoids solving the problem again from scratch and enhances optimization performance.

In practice, the parameter-optimum relational model is always unavailable. However, it could be learned from past optimization records by machine learning methods. Even if the learned model is not exactly the same as the actual one, it can still provide useful information that can benefit the optimization algorithms. Fortunately, EAs are very suitable for testing this idea. We can add the predicted “optimal solutions” to the EA’s population. If the prediction is accurate, the problem is solved instantly, and if the predicted solutions deviate from the actual optimum to some degree, they can still help guide the population to search areas near the optimum. Finally, if the predicted solutions perform poorly, they would be eliminated by the EA’s competition mechanism.

#### 4.1. Preparing the database

To implement the idea above, a database should be prepared first. In our strategies, an entry in the database is a triple  $\langle \theta^t, \mathbf{x}^t, y^t \rangle$ , consisting of the parameter vector, the optimal solution under  $\theta^t$  and its fitness. The actual optimal solution is usually not available. We can use the best solution obtained by the optimization algorithm instead. However, as the population evolves, the best obtained solution under the same parameter vector would change over time. Therefore, the database should be updated. The quality of the best solutions and the size of the database affect learning greatly. If we can collect enough entries, and the best solutions are very close to the actual optima, good learning performance can be expected.

The database updating strategy used in this paper is described in Algorithm 1. When a new entry arrives, if the size of the database is less than a predefined value  $P$ , the candidate is simply added into the database. Otherwise, the entry with the minimum parameter distance is selected. If the Euclidean distance is less than a predefined small threshold  $\sigma$ , the two entries are regarded as similar or from the same environment, and then the selected one is replaced by the new entry if its fitness is worse than the fitness of the new entry.

---

#### Algorithm 1: Pseudocode of $UpdateData(\langle \theta, \mathbf{x}, y \rangle, ds)$ .

---

```

1 Input : the new entry  $\langle \theta, \mathbf{x}, y \rangle$ , database  $ds$ ;
2 Output : The updated database  $ds$ ;
3 Select the entry  $\langle \theta^k, \mathbf{x}^k, y^k \rangle$  in  $ds$  to minimize the Euclidean distance
    $d = \|\theta^k - \theta\|$ ;
4 if  $size(ds) < P$  then
5   | Add the new entry into  $ds$ ;
6 end
7 if  $d < \sigma$  AND  $y$  is better than  $y^k$  then
8   | Replace the selected entry with the new entry in  $ds$ ;
9 end

```

---

#### 4.2. Rote learning and fitting with an FNN

Many machine learning methods have been developed. These methods have successfully discovered many complex patterns in many fields. Rote learning (RL) is one of the earliest machine learning methods [10]. The idea behind rote learning is to memorize solved problems and their solutions and then reuse the appropriate solutions later. Rote learning is simple and can be used as a benchmark algorithm. An FNN can be used to fit the data. It has been shown that an FNN with one hidden layer (and an arbitrary number of hidden units) can approximate any function [13]; therefore, it may be a good choice to fit the parameter-optimum relational model.

When predicting optimal solutions with RL and an FNN, the input consists of the problem parameter vector  $\theta$ , and the output is the predicted best solution  $\mathbf{x}$ . The pseudocodes for the rote learning and the FNN strategies are listed in Algorithm 2 and Algorithm 3, respectively.

---

#### Algorithm 2: Pseudocode of $RoteLearning(\theta)$ .

---

```

1 Input : The parameter vector of the new environment  $\theta$ ;
2 Output : The predicted solution  $\mathbf{x}$ ;
3 Select the entry  $\langle \theta^k, \mathbf{x}^k, y^k \rangle$  in  $ds$  to minimize the Euclidean distance
    $d = \|\theta^k - \theta\|$  in  $ds$ ;
4  $\mathbf{x} \leftarrow \mathbf{x}^k$ ;

```

---

**Algorithm 3:** Pseudocode of  $FNN(\theta)$ .

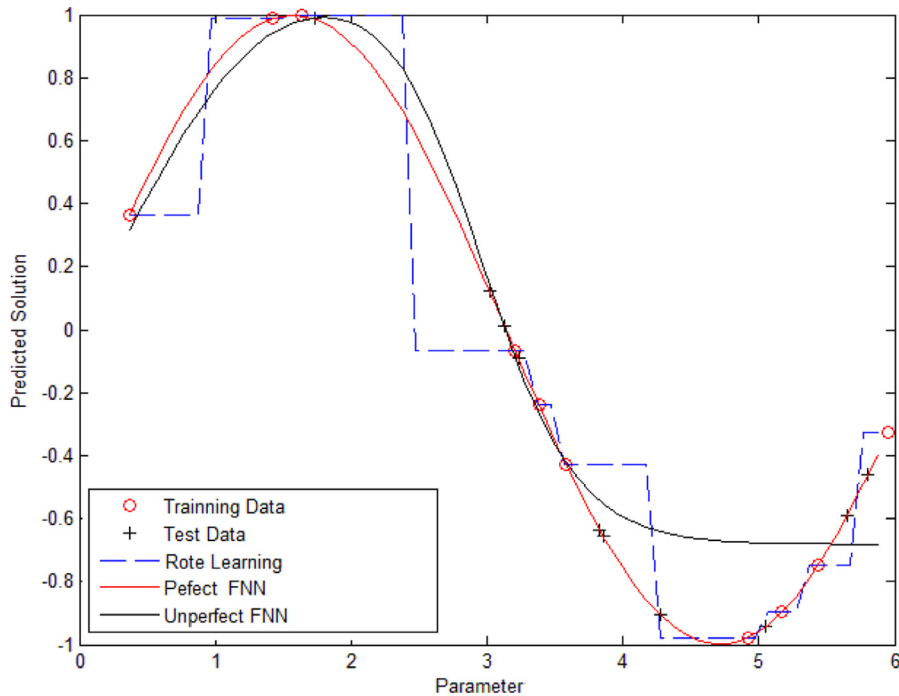
---

```

1 Input : The parameter vector of the new environment  $\theta$ ;
2 Output : The predicted solution  $\mathbf{x}$ ;
3  $net$  is a fully connected FNN, which receives a parameter vector as input and outputs a solution;
4 if Training condition is satisfied then
5   | Train  $net$  on the current database  $\{\dots, (\theta^i, \mathbf{x}^i, y^i), \dots\}$ ; the inputs are the parameters  $\theta^i$ , and the targets are the corresponding solutions  $\mathbf{x}^i$ ;
6 end
7 if  $net$  is available then
8   |  $\mathbf{x} \leftarrow net(\theta)$ ;
9 else
10  |  $\mathbf{x} \leftarrow \text{NULL}$ ;
11 end

```

---



**Fig. 3.** Rote learning vs feedforward neural network.

[Algorithm 2](#) approximates the parameters-optima relational model using the rote learning approach. It assumes that similar environments have similar optimal solutions; thus, it returns the solution of the memorized environment that is most similar to the new environment. The Euclidean distance between the observable parameters is used to measure the environmental similarity.

[Algorithm 3](#) uses an FNN  $net$  to approximate the relation between the parameters and the respective optimal solutions. Once the parameter vector changes, the algorithm is called. New parameters are input, and the predicted optimal solution is returned. This method has a cold start problem. If the size of the database is not large enough,  $net$  is not available. In this case, the algorithm returns an empty solution NULL. The model does not need to be updated every call and can be trained on the current database every few generations. Moreover, the model can be trained in parallel with the EA; therefore, it may not affect the problem-solving process.

In essence, [Algorithm 2](#) can also be regarded as a fitting method. The effects of rote learning and the FNN, as well as their differences, are illustrated in [Fig. 3](#), with only one observable parameter. The training data are the entries in the database from past environments. The test data represent the corresponding entries in the new environments. Rote learning has a problem of overfitting. Although it has no training error, one cannot expect it to generalize well on new test data. In contrast, although it is possible for an FNN to generalize well, it has the risk of underfitting, in which case it would not output correctly even on training data. It could be inferred that rote learning should perform better on recurrent problems, while an FNN is likely to perform better in relatively stochastic scenarios.

The details of the proposed EA with learning strategies are given in [Algorithm 4](#).  $\eta$  is the population size. The strategies are triggered once the an environment change is observed, which means  $\theta^t \neq \theta^{t-1}$ . Firstly, the database is updated with the best solution in the population in terms of fitness at  $\theta^{t-1}$ . Then, *RoteLearning* or *FNN* strategy predicts a solution, named

**Algorithm 4:** The proposed EA with learning strategies.

---

```

1 Set  $\eta, \eta_f, \eta_r, \eta_e, \lambda$ ;
2 Initializing the population pop and the database ds;
3  $I = \emptyset, \lambda_r = \lambda_f = \lambda, t \leftarrow 1$ ;
4 while Stop criteria not satisfied do
5   if  $t \neq 1$  AND  $\theta^t \neq \theta^{t-1}$  then
6     xb is the best in pop at  $t - 1$  with fitness yb;
7     UpdateData( $\{\theta^{t-1}, xb, yb\}, ds$ );
8     if the strategy is Rote Learning then
9       seed  $\leftarrow$  RoteLearning( $\theta^t$ );
10       $I \leftarrow$  Immigrants(seed,  $\eta_r, \lambda$ );
11    end
12    if the strategy is FNN then
13      seed  $\leftarrow$  FNN( $\theta^t$ );
14       $I \leftarrow$  Immigrants(seed,  $\eta_f, \lambda$ );
15    end
16    if the strategy is Ensemble then
17      seed_r  $\leftarrow$  RoteLearning( $\theta^t$ );
18      seed_f  $\leftarrow$  FNN( $\theta^t$ );
19       $I \leftarrow$  Immigrants(seed_r,  $\eta_r, \lambda_r$ );
20       $I \leftarrow I \cup$  Immigrants(seed_f,  $\eta_f, \lambda_f$ );
21    end
22    Replace the worst individuals in pop with I and  $\eta - \eta_r - \eta_f - \eta_e$  random immigrants;
23    Evaluate pop;
24    if the strategy is Ensemble then
25      if seed_r is better than seed_f AND  $\eta_f > 1$  then
26         $\eta_f \leftarrow \eta_f - 1, \eta_r \leftarrow \eta_r + 1$ ;
27         $\lambda_r \leftarrow \lambda, \lambda_f \leftarrow \frac{\lambda \eta_r}{\eta_f}$ ;
28      end
29      if seed_f is better than seed_r AND  $\eta_r > 1$  then
30         $\eta_f \leftarrow \eta_f + 1, \eta_r \leftarrow \eta_r - 1$ ;
31         $\lambda_f \leftarrow \lambda, \lambda_r \leftarrow \frac{\lambda \eta_f}{\eta_r}$ ;
32      end
33    end
34  else
35    Reproduction and selection operations of EA;
36    Evaluate pop;
37     $t \leftarrow t + 1$ ;
38  end
39 end

```

---

*seed*, and generates an immigrant set *I* with the *Immigrants* procedure. *Immigrants*(*seed*, *n*,  $\lambda$ ) generates *n* immigrant as follow:

$$\begin{aligned} x_d &= seed_d + rand(-1, 1) * \lambda, \\ d &= 1, \dots, D \end{aligned} \quad (4)$$

where  $\lambda$  is called as *immigrant radius*, *D* is the solution dimension, and *rand*(−1, 1) uniformly returns a real number from −1 to 1, if *seed* is NULL, *Immigrants* returns an empty set.

The worst individuals of the population in terms of fitness at  $\theta^t$  are replaced by *I* and random immigrants. Random immigrants are uniformly sampled from the whole search space in order to increase the population diversity. The number of random immigrants is  $\eta - \eta_r - \eta_f - \eta_e$ , where  $\eta_e$  is the number of elites from the last environment that we want to preserve.

#### 4.3. The ensemble strategy

According to the “no free lunch” theorem, there is no single learning strategy that can always yield an accurate prediction in any scenario. Therefore, combining two or more basic strategies to form an ensemble is a popular approach in the machine learning community. EAs provide a very convenient framework to combine the proposed RL and FNN strategies, which is the *Ensemble Strategy* described in Algorithm 4. In the ensemble strategy, two sets of immigrants are generated based on the two seeds predicted by *RoteLearning* and the *FNN*, and then the two sets of immigrants are merged and added into the population.

Using two learning strategies provides us with another advantage with which we can evaluate the performance of the strategies and then adjust the algorithm behaviors properly. In our ensemble strategy, after a new population is evaluated,

the performance of the two predicted solutions are compared. If  $seed\_f$  is better than  $seed\_r$ , increase  $\eta_f$  and decrease  $\eta_r$ , and vice versa. In this way, more immigrants can be generated by the better strategy, while the misleading risk by the worse strategy can be lowered. In addition, we suggest amplifying the immigrant radius of the worse strategy, since the actual optimal solution is less likely to appear near the worse seed.

**5. Benchmark problem set**

*5.1. Method to generate benchmark problems*

According to the Nguyen’s DOP definition in Section 2, we can combine different full descriptions with dynamic drivers to generate various DOPs. We need full descriptions to represent different parameter-optima relations and dynamic drivers to represent various ways in which the parameters change.

*5.1.1. Full-function form*

According to Eqs. 1, 2, the parameter vector  $\theta^i$  determines an instance of a full description  $f_i(\mathbf{x})$  and further determines the optimal solutions. In real-world DOPs, there are two kinds of parameter-optima relations. For the first kind of parameter-optima relation, there exists only one optimal solution for any instance  $f_i(\mathbf{x})$  determined by  $\theta^i$ . For the second kind of relation, there could exist multiple equally optimal solutions for a single  $\theta^i$ . Since it is generally more difficult for EAs to obtain multiple optimal solutions simultaneously than to obtain one, it is more difficult to collect enough data to completely reflect the actual parameter-optima relation of the second kind.

To simulate these two kinds of parameter-optima relations, we construct full descriptions as follows. The idea is derived from [17,30].

Given an  $m$ -dimensional static optimization problem as in Eq. 5 that has only one optimal solution  $\mathbf{x}_f^*$  and  $k$  constraints, where  $L$  and  $U$  are the lower and upper search boundaries, respectively,

$$\text{Minimize : } f(\mathbf{x}), \mathbf{x} \in [L, U]^m \tag{5}$$

subject to

$$g_i(\mathbf{x}) \leq 0, i = 1, \dots, k \tag{6}$$

We can transform this problem into a new problem  $F_\theta(\mathbf{x})$  with the following format:

$$\text{Minimize : } F_\theta(\mathbf{x}) = f(r(\mathbf{x}, \theta)), \mathbf{x} \in \mathbb{R}^m \tag{7}$$

subject to

$$\begin{aligned} r(\mathbf{x}, \theta) &\in [L, U]^m \\ g_i(r(\mathbf{x}, \theta)) &\leq 0, i = 1, \dots, k \end{aligned} \tag{8}$$

where  $\mathbf{x}$  represents the solution of  $F_\theta$ ,  $\theta \in \Theta$  represents the vector of observable parameters, and  $r : \mathbb{R}^m \times \Theta \rightarrow \mathbb{R}^m$ , named the *relation function*, is used to explicitly model the interaction of  $\theta$  and  $\mathbf{x}$ . Then, we have a full description as shown below:

$$\langle \widehat{F}_\theta(\mathbf{x}), \Theta \rangle. \tag{9}$$

such that for any  $\theta^i \in \Theta$

$$\widehat{F}_\theta(\mathbf{x}) \xrightarrow{\theta=\theta^i} F_{\theta^i}(\mathbf{x}) = f(r(\mathbf{x}, \theta^i)) \tag{10}$$

It can be inferred that the optimal solutions of any instance  $F_{\theta^i}(\mathbf{x})$  are those that make  $r(\mathbf{x}, \theta^i) = \mathbf{x}_f^*$ . By changing the form of  $r$ , we can flexibly simulate different kinds of parameter-optima relations from real-world problems.

To simulate the first kind of parameter-optima relation, a full description is constructed by using the following relation function  $r_1$  as an example in this paper, which can be decomposed by dimension. Letting  $\theta = \{\theta_1, \theta_2, \dots, \theta_m\} \in \Theta = [-1, 1]^m$ , where  $x_{f_i}^*$  denotes the  $i$ th element of  $\mathbf{x}_f^*$ , for each dimension  $i (i = 1, \dots, m)$ ,

$$r_1(\mathbf{x}, \theta)_i = x_i - \left( L + (U - L) \left( \sin \frac{(\theta_i + 0.1i)\pi}{2} + 1 \right) / 2 \right) + x_{f_i}^* \tag{11}$$

Given any valid value of  $\theta$ , there exists only one optimal solution  $\mathbf{x}$  for  $F_\theta$  such that  $r_1(\mathbf{x}, \theta) = \mathbf{x}_f^*$ .

To simulate the second kind of parameter-optima relation, a full description can be constructed by using the following relation function  $r_2$  as an example, which can also be decomposed by dimension. Letting  $\Theta = [-1, 1]^m$ , for each dimension  $i (i = 1, \dots, m)$ ,

$$r_2(\mathbf{x}, \theta)_i = ((2x_i - L - U)^2 - |\theta_i|(U - L)^2) + x_{f_i}^*. \tag{12}$$

For each dimension, given any valid value of  $\theta_i$  (except  $\theta_i = 0$ ), there are two values for  $x_i$  such that  $r_2(\mathbf{x}, \theta)_i = x_{f_i}^*$ ; therefore, there could be as many as  $2^m$  optimal solutions for  $F_\theta$  such that  $r_2(\mathbf{x}, \theta) = \mathbf{x}_f^*$ .

### 5.1.2. Dynamic drivers

We propose three dynamic drivers to simulate cyclic, random and time-linkage changing parameters as follows.

**Cyclic:** This type of dynamic driver can be modeled by a Fourier function  $fr$  as proposed in [39]:

$$\begin{aligned} \theta_i^t &= fr(\mathbf{a}, \mathbf{A}, \mathbf{b}, \mathbf{B}, t) \\ &= \sum_{j=1}^{|\mathbf{A}|} a_j \cos(2\pi A_j t) + \sum_{j=1}^{|\mathbf{B}|} b_j \sin(2\pi B_j t) \end{aligned} \tag{13}$$

where  $\mathbf{a}$ ,  $\mathbf{A}$ ,  $\mathbf{b}$  and  $\mathbf{B}$  are tunable numerical vectors. The vectors  $\mathbf{a}$  and  $\mathbf{A}$  are the same size, and the vectors  $\mathbf{b}$  and  $\mathbf{B}$  are the same size. Using this function we can flexibly represent periodic functions with different degrees of complexity.

**Random:** This type of dynamic driver is simply represented as a uniform random variable ranging from  $\alpha$  to  $\beta$ .

$$\theta_i^t \sim U(\alpha, \beta) \tag{14}$$

**Time-linkage:** The time-linkage dynamic driver is modeled by a chaotic function as [25]:

$$\begin{aligned} \theta_i^t &= tl(\gamma, t, x_i^{G_{t-1}}) \\ &= \gamma \frac{x_i^{G_{t-1}} - L}{U - L} (1 - \frac{x_i^{G_{t-1}} - L}{U - L}) \end{aligned} \tag{15}$$

where  $x_i^{G_{t-1}}$  is the  $i$ th element of a decision made by algorithm  $G$  at time step  $t - 1$ .

### 5.2. The benchmark problem set

Benchmark problem sets are generated based on 7 static optimization problems, named from  $f_1$  to  $f_7$ .  $f_1$  and  $f_2$  are two-dimensional constrained optimization problems, selected from the CEC'2006 constrained optimization competition [18]. The last five, from  $f_3$  to  $f_7$ , are from the CEC'2005 real-parameter optimization competition [36]. These five problems are scalable, meaning that the dimension can be set to any number desired. Each of these problems has only one optimal solution. Please refer to the Appendix section for more details.

By replacing  $\mathbf{x}$  in each static problem with  $r_1(\mathbf{x}, \theta)$  and  $r_2(\mathbf{x}, \theta)$ , we create two new problems:  $F_{i,\theta}^1(\mathbf{x}) = f_i(r_1(\mathbf{x}, \theta))$  and  $F_{i,\theta}^2(\mathbf{x}) = f_i(r_2(\mathbf{x}, \theta))$ ,  $1 \leq i \leq 7$ . It should be noted that the new problems generated from constrained problems are subjected to the constraints in Eq. 8. For  $F_{i,\theta}^1(\mathbf{x})$ , given any valid value of  $\theta$ , there is only one optimal solution, while for  $F_{i,\theta}^2(\mathbf{x})$ , there are  $2^m$  optimal solutions if each dimension of  $\theta$  is not zero.

The corresponding full descriptions of each newly created problems are as below,  $1 \leq i \leq 7$  in total.

$$\widehat{F}_{i,\theta}^1 : (\widehat{F}_{i,\theta}^1(\mathbf{x}), \Theta), \tag{16}$$

and

$$\widehat{F}_{i,\theta}^2 : (\widehat{F}_{i,\theta}^2(\mathbf{x}), \Theta), \tag{17}$$

where  $\Theta$  is the set of all possible values of  $\theta$ . There are  $7 \times 2 = 14$  full-description forms.

Cyclic, random and time-linkage dynamic drivers from Section 5 are used in the experiments. For Eq. 13, if  $i \% 2 = 1$ ,  $a = [0, 1]$ ,  $A = [0, 0.05]$ ,  $b = B = []$ ; otherwise,  $a = A = []$ ,  $b = [0, 1]$ ,  $B = [0, 0.05]$ . For Eq. 14,  $\alpha = 0$  and  $\beta = 1$ . For Eq. 15,  $\gamma = 3.5$ .

Finally, we generated 42 concrete benchmark DOPs by combining the 12 full-description forms and 3 dynamic drivers. The benchmark problems are denoted as  $f_{i,dd}^r$ , where  $1 \leq i \leq 7$  stands for the static function  $f_i$ ,  $dd \in \{c, ur, tl\}$  is the dynamic driver, and  $r \in \{1, 2\}$  is the relation function. For  $dd$ ,  $c$ ,  $ur$  and  $tl$  denote the cyclic, random and time-linkage dynamic drivers, respectively. Twelve of the 42 DOPs are constrained that are based on  $f_1$  and  $f_2$ , while the remaining 30 are unconstrained DOPs.

Let us take  $f_{3,c}^1$  as an example.  $f_3(\mathbf{x}) = \sum_{i=1}^D x_i^2$ ; if  $D = 2$ ,  $t = 0$ , from Eq. 13, we have  $\theta = [1, 0]$ ; therefore, the instance of  $f_{3,c}^1$  at  $t = 0$  is

$$f_{3,c}^1(\mathbf{x}) = \sum_{i=1}^2 r_1(\mathbf{x}, [1, 0])_i^2.$$

## 6. Experimental settings

### 6.1. Algorithms and settings

In the experiments, for constrained DOPs, the investigated strategies were combined with the constrained version of the SaDE algorithm [14] from the CEC'2006 constrained optimization competition. For unconstrained DOPs, the strategies were combined with a mean-variance optimization algorithm (MVOA) from the CEC'2010 optimization competition [11].

For **Algorithm 1**,  $P$  was set to 20 times the problem's dimension. For **Algorithm 3**, we used MATLAB's built-in function *netfit* to create an FNN with the default tan-sigmoid activation function in the hidden layer and a linear activation function in the output layer. The MATLAB *train* function was used for training. We use one hidden layer for all test cases, the number of hidden neurons was 7, 17 and 32 for 2D (2-dimensional), 5D and 10D problems, respectively. The model, *net*, was unavailable at the beginning until the size of  $ds$  reaches 10 for 2D problems, and 100 for 5D or 10D problems. The training condition was  $rand < 0.1$ , where  $rand$  was a uniform random variable in (0,1). For **Algorithm 1**, we set  $\sigma = 0.02$ . In **Algorithm 4**, the population size  $\eta = 50$ ,  $\eta_e = \eta_f = \eta_r = 10$ .  $\lambda$  was set to one percent of the search range. The above parameter settings were determined by simple preliminary tests.

To validate the proposed learning strategies, we compared them with two existing dynamic optimization approaches: *abstract memory* (AM) proposed in [33] and the elite strategy with random immigrants (ES for short) in [38]. To fairly compare the algorithms, for AM we used the algorithmic settings suggested in [33], which meant a population size of 50, the side of a cell of 0.1 and the number of samples set at 20. For ES, the population size was set to 50, and 40 immigrants were randomly generated in the search space to replace the worst individuals in the population once the parameter changes.

## 6.2. Test problems settings

All of the 42 benchmark DOPs in the previous section were employed in the experiments. The default changing frequency,  $frq$  for short, of the problems is 1/1000, which meant the problems change every 1000 fitness evaluations.

To analyze the influence of the training data quality on the proposed strategies, we also tested the algorithm on constrained DOPs with  $frq$  at 1/500 and 1/2000.

The default dimension of the unconstrained DOPs was 5. However, in order to investigate the scalability of the proposed strategies, we also tested them on some 10D DOPs.

To eliminate the effect of randomness, experimental results were averaged over 30 independent runs. For 2D and 10D problems, each run included 200 environmental changes; for 5D problems, each run included 400 changes.

## 6.3. Performance measurements

For constrained DOPs based on  $f_1$  and  $f_2$ , we used the *modified offline error* (MOE) to measure the performance of the algorithms, which is calculated as follows [27]:

$$MOE = \frac{\sum_{i=1}^E e_{mo}(i)}{E}, \quad (18)$$

where  $E$  is the total number of environments during the optimization, and  $e_{mo}(i)$  is the best *feasible* error obtained in the  $i$ th environment. If the best solution is feasible,  $e_{mo}(i)$  is the fitness difference between the obtained best solution and the real best fitness. Otherwise,  $e_{mo}(i)$  is the worst possible error that a feasible solution could have, which was set to 2.5 as in [6,29].

For unconstrained DOPs, the generational logarithmic offline error (GLOE) was used to measure the performance as follows.

$$GLOE = \frac{1}{G} \sum_{i=1}^G \frac{1}{E} \sum_{j=1}^E \log_{10}(e_{ij}^{BOG}) \quad (19)$$

where  $G$  is the number of generations, and  $e_{ij}^{BOG}$  the error between the global optimal fitness and the fitness of the best individual of generation  $i$  at environment  $j$ . Compared to the offline error, logarithmic offline error can reduce the influence of few large offline errors.

In these experiments, to evaluate the predicted optimal solutions of RL, FNN and Ens, the seed modified offline error (SMOE) was proposed as follows.

$$SMOE = \frac{1}{N} \sum_{i=1}^N se(i) \quad (20)$$

In the above equation,  $N$  is the number of predicted optimal solutions by a strategy in a single run, and  $se(i)$  is the error between the fitness of the  $i$ th predicted optimal solution and the global optimal fitness.

## 7. Results and analysis

### 7.1. Experimental comparisons

MOE or GLOE of the algorithms on the test problems are given in **Table 1** and **Table 2**. To the left of '±' is the average MOE or GLOE over 30 independent runs, while to the right is the variance. From the experimental results, it can be seen that the proposed rote learning (RL) and ensemble strategies (Ens) achieve better MOE and GLOE than the ES and AM do in

**Table 1**  
MOE of the algorithms on the constrained DOPs,  $frq=1/500,1/1000,1/2000$ .

$frq=1/500$	$f_{1,c}^1$	$f_{1,ur}^1$	$f_{1,tl}^1$	$f_{1,c}^2$	$f_{1,ur}^2$	$f_{1,tl}^2$
ES	2.95E-2 ± 1.7E-3	3.00E-2 ± 1.7E-3	4.48E-2 ± 2.1E-3	9.57E-2 ± 9.9E-4	9.57E-2 ± 6.9E-4	9.51E-2 ± 6.5E-4
RL	9.53E-3 ± 1.1E-3	1.24E-2 ± 8.4E-4	1.70E-2 ± 1.7E-3	9.42E-2 ± 1.4E-3	9.52E-2 ± 8.5E-4	9.47E-2 ± 1.3E-3
FNN	1.40E-2 ± 3.7E-3	1.05E-2 ± 1.6E-3	1.81E-2 ± 3.4E-3	9.55E-2 ± 7.7E-4	9.55E-2 ± 5.6E-4	9.52E-2 ± 7.3E-4
AM	3.01E-2 ± 1.8E-3	2.36E-2 ± 2.1E-3	2.74E-2 ± 2.2E-3	9.54E-2 ± 7.3E-4	9.55E-2 ± 7.2E-4	9.51E-2 ± 7.4E-4
Ens	<b>3.75E-3 ± 5.6E-4</b>	<b>4.07E-3 ± 9.3E-4</b>	<b>5.08E-3 ± 6.2E-4</b>	<b>8.88E-2 ± 2.4E-3</b>	<b>9.39E-2 ± 8.7E-4</b>	<b>9.18E-2 ± 1.3E-3</b>
$frq=1/500$	$f_{2,c}^1$	$f_{2,ur}^1$	$f_{2,tl}^1$	$f_{2,c}^2$	$f_{2,ur}^2$	$f_{2,tl}^2$
ES	2.48E-1 ± 8.4E-3	2.96E-1 ± 1.1E-2	2.84E-1 ± 1.1E-2	1.54 ± 7.3E-2	1.09 ± 5.1E-2	1.08 ± 3.4E-2
RL	1.36E-1 ± 8.1E-3	1.91E-1 ± 8.9E-3	1.65E-1 ± 7.8E-3	1.07 ± 6.4E-2	8.60E-1 ± 4.6E-2	8.74E-1 ± 3.4E-2
FNN	1.77E-1 ± 2.4E-2	2.06E-1 ± 2.7E-2	1.81E-1 ± 2.3E-2	1.50 ± 1.1E-1	1.09 ± 6.4E-2	1.07 ± 3.4E-2
AM	2.55E-1 ± 1.2E-2	2.55E-1 ± 9.2E-3	2.46E-1 ± 1.1E-2	1.45 ± 8.2E-2	1.05 ± 4.8E-2	1.03 ± 3.7E-2
Ens	<b>8.84E-2 ± 6.4E-3</b>	<b>1.27E-1 ± 1.4E-2</b>	<b>1.06E-1 ± 8.8E-3</b>	<b>7.69E-1 ± 4.0E-2</b>	<b>7.01E-1 ± 3.0E-2</b>	<b>7.32E-1 ± 3.0E-2</b>
$frq=1/1000$	$f_{1,c}^1$	$f_{1,ur}^1$	$f_{1,tl}^1$	$f_{1,c}^2$	$f_{1,ur}^2$	$f_{1,tl}^2$
ES	2.71E-3 ± 2.9E-4	1.83E-3 ± 1.8E-4	5.30E-3 ± 5.0E-4	9.16E-2 ± 1.2E-3	9.27E-2 ± 1.6E-3	9.30E-2 ± 1.4E-3
RL	7.30E-4 ± 1.2E-4	7.92E-4 ± 1.1E-4	1.50E-3 ± 1.4E-4	8.30E-2 ± 1.4E-3	8.98E-2 ± 2.0E-3	8.79E-2 ± 2.0E-3
FNN	1.11E-3 ± 3.4E-4	5.93E-4 ± 2.2E-4	1.66E-3 ± 2.1E-4	9.07E-2 ± 1.7E-3	9.21E-2 ± 1.3E-3	9.28E-2 ± 1.3E-3
AM	2.64E-3 ± 2.9E-4	1.45E-3 ± 1.2E-4	3.02E-3 ± 2.8E-4	9.04E-2 ± 1.5E-3	9.30E-2 ± 1.7E-3	9.31E-2 ± 1.5E-3
Ens	<b>3.66E-4 ± 1.2E-4</b>	<b>2.04E-4 ± 4.5E-5</b>	<b>4.38E-4 ± 9.1E-5</b>	<b>6.47E-2 ± 2.2E-3</b>	<b>7.98E-2 ± 2.2E-3</b>	<b>6.76E-2 ± 2.9E-3</b>
$frq=1/1000$	$f_{2,c}^1$	$f_{2,ur}^1$	$f_{2,tl}^1$	$f_{2,c}^2$	$f_{2,ur}^2$	$f_{2,tl}^2$
ES	1.07E-1 ± 4.1E-3	1.25E-1 ± 5.0E-3	1.21E-1 ± 5.2E-3	8.15E-1 ± 2.7E-2	6.25E-1 ± 2.5E-2	6.51E-1 ± 3.0E-2
RL	6.13E-2 ± 2.9E-3	8.72E-2 ± 3.3E-3	7.61E-2 ± 3.7E-3	6.02E-1 ± 2.5E-2	5.11E-1 ± 2.7E-2	5.33E-1 ± 2.2E-2
FNN	7.60E-2 ± 8.1E-3	8.36E-2 ± 1.0E-2	7.73E-2 ± 5.6E-3	8.08E-1 ± 3.4E-2	6.14E-1 ± 3.0E-2	6.37E-1 ± 2.3E-2
AM	1.20E-1 ± 4.2E-3	1.15E-1 ± 5.8E-3	1.14E-1 ± 4.9E-3	7.44E-1 ± 2.1E-2	5.99E-1 ± 2.1E-2	6.28E-1 ± 2.1E-2
Ens	<b>4.04E-2 ± 2.5E-3</b>	<b>5.66E-2 ± 6.9E-3</b>	<b>4.90E-2 ± 3.7E-3</b>	<b>4.72E-1 ± 1.4E-2</b>	<b>4.12E-1 ± 2.4E-2</b>	<b>4.42E-1 ± 2.1E-2</b>
$frq=1/2000$	$f_{1,c}^1$	$f_{1,ur}^1$	$f_{1,tl}^1$	$f_{1,c}^2$	$f_{1,ur}^2$	$f_{1,tl}^2$
ES	6.54E-5 ± 2.9E-5	7.60E-6 ± 2.0E-6	8.95E-5 ± 3.2E-5	5.89E-2 ± 2.1E-3	6.01E-2 ± 2.7E-3	5.68E-2 ± 1.9E-3
RL	1.05E-5 ± 5.1E-6	2.82E-6 ± 8.9E-7	1.75E-5 ± 5.8E-6	3.35E-2 ± 2.3E-3	4.35E-2 ± 2.0E-3	3.45E-2 ± 2.3E-3
FNN	2.13E-5 ± 1.5E-5	1.44E-6 ± 5.7E-7	2.23E-5 ± 1.3E-5	5.83E-2 ± 1.9E-3	5.97E-2 ± 2.9E-3	5.85E-2 ± 4.4E-3
AM	5.73E-5 ± 1.9E-5	6.53E-6 ± 1.6E-6	4.36E-5 ± 9.5E-6	5.42E-2 ± 1.7E-3	5.96E-2 ± 3.1E-3	5.53E-2 ± 2.1E-3
Ens	<b>5.78E-6 ± 4.6E-6</b>	<b>6.18E-7 ± 3.4E-7</b>	<b>4.71E-6 ± 2.4E-6</b>	<b>1.18E-2 ± 1.9E-3</b>	<b>2.20E-2 ± 1.7E-3</b>	<b>1.47E-2 ± 2.7E-3</b>
$frq=1/2000$	$f_{2,c}^1$	$f_{2,ur}^1$	$f_{2,tl}^1$	$f_{2,c}^2$	$f_{2,ur}^2$	$f_{2,tl}^2$
ES	1.79E-2 ± 7.3E-4	2.01E-2 ± 1.0E-3	2.08E-2 ± 1.2E-3	4.62E-1 ± 9.0E-3	3.08E-1 ± 1.7E-2	3.30E-1 ± 1.1E-2
RL	1.19E-2 ± 4.4E-4	1.67E-2 ± 7.9E-4	1.59E-2 ± 6.7E-4	3.67E-1 ± 9.4E-3	2.56E-1 ± 1.1E-2	2.85E-1 ± 9.1E-3
FNN	1.44E-2 ± 1.2E-3	1.50E-2 ± 8.8E-4	1.54E-2 ± 1.1E-3	4.58E-1 ± 1.4E-2	2.98E-1 ± 1.4E-2	3.30E-1 ± 1.7E-2
AM	2.05E-2 ± 8.8E-4	1.90E-2 ± 7.7E-4	2.16E-2 ± 1.2E-3	4.42E-1 ± 9.7E-3	2.98E-1 ± 1.3E-2	3.30E-1 ± 1.3E-2
Ens	<b>8.23E-3 ± 5.4E-4</b>	<b>1.12E-2 ± 5.2E-4</b>	<b>1.10E-2 ± 8.6E-4</b>	<b>3.06E-1 ± 7.6E-3</b>	<b>2.03E-1 ± 1.3E-2</b>	<b>2.39E-1 ± 9.9E-3</b>

almost all test problems, and the FNN strategy achieves better MOE and GLOE than ES, AM and RL in all the test problems with the  $r_1$  relation and random dynamic driver.

Recently, the conventional null hypothesis significance testing (NHST) has been questioned, and new statistical analysis approaches are recommended as alternatives [2,7,8]. Therefore, instead of NHST, we employed a Bayesian analysis approach, named the Bayesian signed-rank test [2], to statistically compare the algorithms based on the results in Table 1 and Table 2. *rope*, the parameter of the Bayesian analysis approach, is set to 0.001 for constrained problems and 0.01 for unconstrained problems. The pairwise comparison results are given in Table 3 and Table 4. The first column of the two tables denotes the set of benchmark DOPs. The result of each comparison is a triple of probabilities. Taking the first triple of Table 3, (0.99998, 2e-05, 0.0), for example, it means there is a 99.998% probability that the average performance of Ens on  $r_1$  problems is better than that of ES, and there is a 0.002% chance that the difference between the two classifiers is negligible (that is, smaller than *rope* = 0.001). From Tables 3, 4, it can be seen that almost on all benchmark problems, Ens outperforms other algorithms with a high probability ( $\approx 1$ ), and the probability for any other algorithm to outperform Ens is 0 in all test cases. These results validate the proposed strategies.

### 7.2. Influence of relation function

From Table 1 and Table 2, it can be seen that relation function impacts the algorithmic performance greatly. The errors of algorithms on  $r_1$ -relevant problems are much less than the errors on  $r_2$ -relevant problems, meaning that  $r_2$ -relevant problems are much more difficult to optimize. For example, on  $f_{1,ur}^1$  with  $frq = 1/2000$ , the error of ES is less than  $10^{-5}$ , while on  $f_{1,ur}^2$  with  $frq = 1/2000$ , the error of ES is near 0.06. This phenomenon has also been observed in other cases.

**Table 2**GLOE of the algorithms on the unconstrained DOPs,  $frq=1/1000$ ,  $Dim=5$ .

probs	$f_{3,c}^1$	$f_{3,ur}^1$	$f_{3,tl}^1$	$f_{3,c}^2$	$f_{3,ur}^2$	$f_{3,tl}^2$
ES	2.65 ± 6.0E-3	2.63 ± 9.2E-3	2.67 ± 9.9E-3	6.66 ± 1.6E-2	6.65 ± 1.5E-2	6.60 ± 1.6E-2
RL	9.97E-2 ± 1.5E-3	2.41 ± 1.1E-2	2.51 ± 1.4E-2	2.49E-1 ± 2.6E-3	6.34 ± 1.2E-2	6.15 ± 1.7E-2
FNN	6.24E-1 ± 7.3E-3	7.09E-1 ± 3.5E-2	9.69E-1 ± 5.1E-2	6.66 ± 1.5E-2	6.67 ± 1.7E-2	6.61 ± 1.5E-2
AM	3.12E-1 ± 3.0E-2	2.55 ± 1.0E-2	2.59 ± 9.4E-3	2.66 ± 1.4E-2	6.62 ± 1.5E-2	6.37 ± 1.6E-2
Ens	<b>9.56E-2 ± 1.4E-3</b>	<b>6.59E-1 ± 5.0E-2</b>	<b>8.84E-1 ± 4.8E-2</b>	<b>2.48E-1 ± 2.6E-3</b>	<b>6.27 ± 1.3E-2</b>	<b>6.08 ± 1.6E-2</b>
probs	$f_{4,c}^1$	$f_{4,ur}^1$	$f_{4,tl}^1$	$f_{4,c}^2$	$f_{4,ur}^2$	$f_{4,tl}^2$
ES	2.73 ± 1.1E-2	2.73 ± 9.9E-3	2.75 ± 1.3E-2	6.73 ± 1.5E-2	6.73 ± 1.4E-2	6.68 ± 1.5E-2
RL	1.03E-1 ± 1.4E-3	2.50 ± 1.2E-2	2.60 ± 1.2E-2	2.50E-1 ± 2.6E-3	6.41 ± 1.5E-2	6.24 ± 1.8E-2
FNN	6.46E-1 ± 1.1E-2	7.73E-1 ± 4.7E-2	1.04 ± 5.2E-2	6.73 ± 2.1E-2	6.75 ± 1.9E-2	6.70 ± 1.5E-2
AM	3.35E-1 ± 4.9E-2	2.63 ± 1.2E-2	2.68 ± 1.2E-2	2.79 ± 1.8E-2	6.68 ± 1.6E-2	6.50 ± 1.8E-2
Ens	<b>1.01E-1 ± 1.3E-3</b>	<b>6.86E-1 ± 3.9E-2</b>	<b>9.75E-1 ± 6.9E-2</b>	<b>2.49E-1 ± 2.1E-3</b>	<b>6.33 ± 1.8E-2</b>	<b>6.15 ± 1.9E-2</b>
probs	$f_{5,c}^1$	$f_{5,ur}^1$	$f_{5,tl}^1$	$f_{5,c}^2$	$f_{5,ur}^2$	$f_{5,tl}^2$
ES	3.36 ± 1.2E-2	3.37 ± 1.4E-2	3.39 ± 1.0E-2	8.08 ± 9.6E-3	8.16 ± 1.4E-2	8.16 ± 1.3E-2
RL	9.19E-1 ± 4.0E-2	3.11 ± 1.3E-2	3.26 ± 1.3E-2	5.78 ± 8.8E-2	7.89 ± 1.4E-2	8.01 ± 1.5E-2
FNN	2.22 ± 4.9E-2	2.48 ± 5.5E-2	2.49 ± 5.6E-2	8.08 ± 9.6E-3	8.18 ± 9.3E-3	8.18 ± 1.3E-2
AM	2.68 ± 5.8E-2	3.26 ± 1.2E-2	3.34 ± 1.2E-2	6.50 ± 1.3E-1	8.12 ± 1.1E-2	8.10 ± 1.4E-2
Ens	<b>6.72E-1 ± 3.9E-2</b>	<b>2.15 ± 4.5E-2</b>	<b>2.21 ± 6.7E-2</b>	<b>5.20 ± 7.0E-2</b>	<b>7.79 ± 1.9E-2</b>	<b>7.93 ± 1.6E-2</b>
probs	$f_{6,c}^1$	$f_{6,ur}^1$	$f_{6,tl}^1$	$f_{6,c}^2$	$f_{6,ur}^2$	$f_{6,tl}^2$
ES	6.60 ± 2.0E-2	6.56 ± 2.1E-2	6.64 ± 2.1E-2	1.52E1 ± 3.2E-2	1.54E1 ± 3.2E-2	1.53E1 ± 3.6E-2
RL	2.93E-1 ± 1.1E-2	6.06 ± 2.9E-2	6.32 ± 2.5E-2	2.76 ± 1.8E-1	1.48E1 ± 3.4E-2	1.42E1 ± 4.3E-2
FNN	2.82 ± 1.2E-1	3.76 ± 1.6E-1	3.34 ± 1.2E-1	1.52E1 ± 2.9E-2	1.55E1 ± 2.6E-2	1.53E1 ± 3.2E-2
AM	2.70 ± 1.6E-1	6.35 ± 2.1E-2	6.50 ± 2.6E-2	7.61 ± 6.7E-2	1.54E1 ± 2.2E-2	1.45E1 ± 4.8E-2
Ens	<b>2.90E-1 ± 1.6E-2</b>	<b>3.31 ± 1.5E-1</b>	<b>3.05 ± 8.3E-2</b>	<b>2.40 ± 2.1E-1</b>	<b>1.46E1 ± 2.4E-2</b>	<b>1.39E1 ± 5.0E-2</b>
probs	$f_{7,c}^1$	$f_{7,ur}^1$	$f_{7,tl}^1$	$f_{7,c}^2$	$f_{7,ur}^2$	$f_{7,tl}^2$
ES	3.15 ± 8.4E-3	3.14 ± 9.3E-3	3.17 ± 8.5E-3	7.07 ± 1.4E-2	7.13 ± 1.2E-2	7.08 ± 1.3E-2
RL	1.29 ± 3.8E-2	2.93 ± 1.3E-2	3.05 ± 8.1E-3	1.78 ± 5.3E-2	6.82 ± 1.5E-2	6.61 ± 1.5E-2
FNN	2.08 ± 1.8E-2	2.19 ± 1.8E-2	2.17 ± 3.5E-2	7.07 ± 1.5E-2	7.14 ± 1.3E-2	7.09 ± 1.5E-2
AM	2.30 ± 6.1E-2	3.06 ± 8.2E-3	3.12 ± 9.0E-3	3.10 ± 2.1E-2	7.09 ± 9.0E-3	6.84 ± 1.5E-2
Ens	<b>1.06 ± 2.9E-2</b>	<b>2.02 ± 2.2E-2</b>	<b>1.97 ± 3.6E-2</b>	<b>1.72 ± 5.1E-2</b>	<b>6.74 ± 1.4E-2</b>	<b>6.52 ± 1.6E-2</b>

In terms of algorithm performance ranking, FNN is the algorithm most affected by the relation function. On all of the  $r_1$  problems, FNN achieves better errors than ES and AM do, and on the random  $r_1$ -relevant problems, FNN outperforms the RL strategy. However, on most of the  $r_2$ -relevant problems, FNN is beaten by the AM strategy. Considering AM does not use observable parameters, the loss of FNN means that using the observed parameters does not always work. In contrast, the RL is relatively robust; it outperforms ES and AM on all of the problems despite the relation function.

The reason that this phenomenon occurs can be partly explained from the seed errors. Table 5 lists the SMOE of the algorithms on the constrained DOPs. It can be seen that the SMOE of the FNN on  $r_1$  relevant problems differs greatly from  $r_2$  relevant problems, while the difference of RL's SMOE is relatively slight. For example, in the right of Table 5, on  $f_{1,ur}^1$  and  $f_{1,ur}^2$ , the SMOE of FNN is 1.07E-02 and 9.58E-02, while for RL it is 7.64E-2 and 9.58E-2 respectively. These results show that prediction accuracy of FNN degrades greatly on  $r_2$  relevant problems.

The proposed Ens strategy outperforms all of the others on all test problems regardless of the type of relation function. These results show that the ensemble strategy successfully combines the advantages of RL and FNN.

### 7.3. Influence of dynamic driver

In our experimental setting, with the cyclically dynamic driver, there are 20 states in a cycle. From Table 1 and Table 2, it can be clearly seen that on cyclic problems, RL outperforms AM and FNN. These results are expected because with the observable parameters, RL can accurately retrieve the past best solutions obtained for any parameter vector. AM does not store the accurate positions of the best solutions but rather the cells where they locate, and thus it is difficult to recover the past best solutions exactly. For FNN, as discussed in Section 4, it could have the problem of underfitting. Therefore, it might also fail to recover the past solutions.

In uniformly random environments, it can be clearly seen that FNN achieves better errors than RL, AM and ES do on  $r_1$  relevant problems. However, similar phenomenon are not clearly seen on  $r_2$  relevant problems, and the reason is given in the subsection above. In contrast to recurrent environments, a parameter vector could never reappear exactly, which means the test data could always be different from the training data. As stated in Section 4, RL has the problem of overfitting and cannot generalize well; therefore, it is expected to be less effective in random environments than in recurrent environments.

**Table 3**  
Comparing algorithms using Bayesian signed-ranks test with data from Table 1.

Problem set	Ens-ES	Ens-AM	Ens-RL	Ens-FNN	FNN-RL
r1 problems	(0.99998, 2e-05, 0.0)	(0.99998, 2e-05, 0.0)	(0.9993, 0.0007, 0.0)	(0.99954, 0.00046, 0.0)	(0.00552, 0.25802, 0.73646)
r2 problems	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(0.0, 0.0, 1.0)
cyclic problems	(0.99998, 2e-05, 0.0)	(0.99998, 2e-05, 0.0)	(0.99992, 8e-05, 0.0)	(0.99986, 0.00014, 0.0)	(0.0, 0.0004, 0.9996)
random problems	(0.9999, 0.0001, 0.0)	(0.9998, 0.0002, 0.0)	(0.9996, 0.0004, 0.0)	(0.99982, 0.00018, 0.0)	(0.01274, 0.03058, 0.95668)
time-linkage problems	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(0.99998, 2e-05, 0.0)	(0.99996, 4e-05, 0.0)	(0.0, 0.04906, 0.95094)
<i>frq</i> = 1/500	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(6e-05, 0.00792, 0.99202)
<i>frq</i> = 1/1000	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(0.99952, 0.00048, 0.0)	(0.9994, 0.0006, 0.0)	(0.0016, 0.00804, 0.99036)
<i>frq</i> = 1/2000	(0.99908, 0.00092, 0.0)	(0.99908, 0.00092, 0.0)	(0.99892, 0.00108, 0.0)	(0.99906, 0.00094, 0.0)	(0.00016, 0.02058, 0.97926)

**Table 4**  
Comparing algorithms using Bayesian signed-ranks test with data from Table 2.

Problem set	Ens-ES	Ens-AM	Ens-RL	Ens-FNN	FNN-RL
r1 problems	(0.95698, 0.04302, 0.0)	(0.96456, 0.03544, 0.0)	(0.6215, 0.3785, 0.0)	(0.6389, 0.3611, 0.0)	(0.0, 0.99708, 0.00292)
r2 problems	(1.0, 0.0, 0.0)	(0.99998, 2e-05, 0.0)	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(0.0, 0.00162, 0.99838)
cyclic problems	(0.9947, 0.0053, 0.0)	(0.9963, 0.0037, 0.0)	(0.95946, 0.04054, 0.0)	(0.98064, 0.01936, 0.0)	(0.0, 0.175, 0.825)
random problems	(0.98706, 0.01294, 0.0)	(0.98018, 0.01982, 0.0)	(0.91154, 0.08846, 0.0)	(0.92802, 0.07198, 0.0)	(0.0, 0.63346, 0.36654)
time-linkage problems	(0.99494, 0.00506, 0.0)	(0.9967, 0.0033, 0.0)	(0.97484, 0.02516, 0.0)	(0.98062, 0.01938, 0.0)	(0.0, 0.39208, 0.60792)

**Table 5**

SEMO of the algorithms on the constrained DOPs, the left half with  $frq=1/500$ , the right half with frequency=1/2000.

probs	$f_{1,c}^1$	$f_{1,ur}^1$	$f_{1,tl}^1$	$f_{1,c}^1$	$f_{1,ur}^1$	$f_{1,tl}^1$
RL	2.22E-2 ± 1.55E-3	7.84E-2 ± 2.54E-3	4.70E-2 ± 3.02E-3	1.36E-2 ± 1.28E-3	7.64E-2 ± 2.02E-3	2.61E-2 ± 1.19E-3
FNN	5.26E-2 ± 1.57E-2	2.97E-2 ± 7.96E-3	3.78E-2 ± 1.17E-2	3.86E-2 ± 1.46E-2	<b>1.07E-2 ± 2.48E-3</b>	1.60E-2 ± 7.16E-3
Ens	<b>1.58E-2 ± 1.75E-3</b>	<b>2.71E-2 ± 9.98E-3</b>	<b>1.93E-2 ± 3.18E-3</b>	<b>1.26E-2 ± 1.36E-3</b>	1.46E-2 ± 1.87E-3	<b>1.10E-2 ± 1.68E-3</b>
probs	$f_{2,c}^1$	$f_{2,ur}^1$	$f_{2,tl}^1$	$f_{2,c}^1$	$f_{2,ur}^1$	$f_{2,tl}^1$
RL	9.11E-1 ± 5.59E-2	4.74 ± 2.28E-1	3.96 ± 2.57E-1	8.43E-1 ± 4.45E-2	6.88 ± 1.74E-1	6.31 ± 2.27E-1
FNN	4.76 ± 1.05	4.38 ± 1.09	3.95 ± 1.25	5.58 ± 1.32	4.13 ± 8.89E-1	4.53 ± 9.60E-1
Ens	<b>7.79E-1 ± 6.61E-2</b>	<b>3.05 ± 8.17E-1</b>	<b>2.69 ± 6.93E-1</b>	<b>7.50E-1 ± 3.87E-2</b>	<b>3.99 ± 7.91E-1</b>	<b>3.94 ± 8.46E-1</b>
probs	$f_{1,c}^2$	$f_{1,ur}^2$	$f_{1,tl}^2$	$f_{1,c}^2$	$f_{1,ur}^2$	$f_{1,tl}^2$
RL	9.43E-2 ± 1.37E-3	9.59E-2 ± 1.70E-4	9.59E-2 ± 5.18E-4	3.99E-2 ± 2.20E-3	9.58E-2 ± 2.03E-4	9.50E-2 ± 9.98E-4
FNN	9.50E-2 ± 2.21E-3	9.58E-2 ± 9.50E-17	9.58E-2 ± 3.57E-5	9.55E-2 ± 1.20E-3	9.58E-2 ± 1.41E-17	9.58E-2 ± 1.41E-17
Ens	<b>8.94E-2 ± 3.05E-3</b>	<b>9.58E-2 ± 1.52E-4</b>	<b>9.58E-2 ± 5.45E-4</b>	<b>2.02E-2 ± 1.80E-3</b>	<b>9.58E-2 ± 1.84E-4</b>	<b>9.32E-2 ± 1.14E-3</b>
probs	$f_{2,c}^2$	$f_{2,ur}^2$	$f_{2,tl}^2$	$f_{2,c}^2$	$f_{2,ur}^2$	$f_{2,tl}^2$
RL	1.77 ± 6.38E-2	5.73 ± 2.39E-1	5.82 ± 2.64E-1	1.11 ± 1.55E-2	6.69 ± 1.70E-1	6.90 ± 1.74E-1
FNN	7.48 ± 6.54E-1	7.80 ± 1.34E-1	7.77 ± 1.90E-1	7.58 ± 5.12E-1	7.79 ± 1.35E-1	7.80 ± 1.56E-1
Ens	<b>1.51 ± 4.10E-2</b>	<b>5.63 ± 2.62E-1</b>	<b>5.67 ± 3.41E-1</b>	<b>1.06 ± 1.45E-2</b>	<b>6.56 ± 1.97E-1</b>	<b>6.78 ± 2.73E-1</b>

**Table 6**

The average running time (in seconds) of the algorithms for 1000 fitness evaluations on  $f_{6,ur}^1$  problem.

	ES	AM	RL	FNN	Ens
2D	0.37	0.41	0.41	0.63	0.69
5D	0.433	0.447	0.44	0.907	0.93
10D	0.467	0.583	0.459	0.971	0.99
10D pretrained	0.467	0.589	0.464	0.702	0.72

This explanation is confirmed by the SMOEs in Table 5. For example, with  $frq = 1/500$ , the SMOE of RL on  $f_{1,c}^1$  is 2.22E-2, while on  $f_{1,ur}^1$ , it is 7.84E-2.

The time-linkage problems of this paper are not time-deceptive; therefore, no special treatment is required. From the experimental results, it can be seen that the RL algorithm is better than ES, AM and FNN, which indicates its robustness.

The results show that the ensemble strategy is the best among all the algorithms. It is interesting that in some cases such as  $f_{1,ur}^1$  with  $frq = 1/2000$ , FNN achieves a better SMOE than does Ens, but its MOE is worse than that of Ens. These cases indicate that Ens is robust.

#### 7.4. Influence of database quality

Database quality has a great effect on the performance of machine learning algorithms. The quality of the database includes the quality of each data entry and the number of entries. In our experiments, we use three change frequencies: 1/500, 1/1000, and 1/2000. With a lower changing frequency, algorithms have more time to locate the optima in each environment, and thus the obtained best solutions are closer to the real optima and can more truly reflect the real-parameter-optima relations. Moreover, the difficulty of the problems also affects the entry quality. In our experiments,  $r_2$  relevant problems are more difficult than  $r_1$  relevant problems. The dynamic driver determines the number of effective entries. In the cyclic environments of our experiments, the number of parameter vectors is 20, while in random environments, the number of parameter vectors is usually equal to the number of environmental changes.

From the SMOE tables, the influence of the database quality can be clearly observed. The SMOE of the algorithms with  $frq = 1/2000$  is generally smaller than those with  $frq = 1/500$ . Compared with RL, the impact of database quality on FNN is more significant. For example, on  $f_{1,ur}^1$ , the SMOE values of RL are 7.84E-2 and 7.64E-2 for  $frq = 1/500$  and  $frq = 1/2000$ , respectively, while the SMOEs of FNN are 2.97E-2 and 1.07E-2, respectively, and substantially differ.

The database quality even changes the algorithm performance ranking on some test problems. For example, on  $f_{2,ur}^1$  and  $f_{2,tl}^1$ , FNN is slightly worse than RL with  $frq = 1/500$ , while with  $frq = 1/2000$ , FNN is better than RL.

#### 7.5. The scalability of the proposed strategies

To analyze the scalability of the algorithms, the running time of the algorithms on the  $f_{6,ur}^1$  problem is recorded and given in Table 6. In addition, the algorithms are tested on 10D DOPs, and the results are given in Table 7.

**Table 7**

GLOE of the algorithms on the unconstrained DOPs,  $freq = 1/1000$ , Dim=10.

probs	$f_{3,c}^1$	$f_{3,ur}^1$	$f_{3,tl}^1$
ES	$3.43 \pm 9.8E-3(+)$	$3.47 \pm 1.1E-2(+)$	$3.51 \pm 9.8E-3$
RL	$2.59E-1 \pm 2.8E-3$	$3.41 \pm 1.5E-2$	$3.46 \pm 1.1E-2$
FNN	$6.42E-1 \pm 3.4E-3$	$4.88E-1 \pm 1.1E-2$	$4.40E-1 \pm 1.2E-2$
AM	$3.09 \pm 9.1E-2$	$3.47 \pm 1.4E-2$	$3.50 \pm 1.0E-2$
Ens	<b><math>4.61E-2 \pm 7.3E-4</math></b>	<b><math>4.84E-1 \pm 1.5E-2</math></b>	<b><math>4.35E-1 \pm 1.2E-2</math></b>
probs	$f_{4,c}^1$	$f_{4,ur}^1$	$f_{4,tl}^1$
ES	$3.59 \pm 1.1E-2$	$3.65 \pm 1.5E-2$	$3.64 \pm 1.4E-2$
RL	$2.71E-1 \pm 2.9E-3$	$3.60 \pm 1.5E-2$	$3.60 \pm 9.9E-3$
FNN	$9.89E-1 \pm 1.3E-2$	$8.83E-1 \pm 2.8E-2$	$8.24E-1 \pm 1.8E-2$
AM	$3.25 \pm 8.3E-2$	$3.66 \pm 1.2E-2$	$3.63 \pm 9.2E-3$
Ens	<b><math>6.81E-2 \pm 2.6E-3</math></b>	<b><math>8.62E-1 \pm 1.9E-2</math></b>	<b><math>7.91E-1 \pm 2.6E-2</math></b>
probs	$f_{5,c}^1$	$f_{5,ur}^1$	$f_{5,tl}^1$
ES	$4.30 \pm 6.6E-3$	$4.36 \pm 1.2E-2$	$4.35 \pm 9.0E-3$
RL	$3.23 \pm 5.7E-2$	$4.30 \pm 1.2E-2$	$4.32 \pm 1.1E-2$
FNN	$1.23 \pm 1.3E-2$	$1.13 \pm 2.5E-2$	$1.08 \pm 5.1E-2$
AM	$4.29 \pm 1.3E-2$	$4.37 \pm 1.4E-2$	$4.34 \pm 1.0E-2$
Ens	<b><math>8.67E-1 \pm 3.8E-2</math></b>	<b><math>1.08 \pm 2.5E-2</math></b>	<b><math>1.06 \pm 4.8E-2</math></b>
probs	$f_{6,c}^1$	$f_{6,ur}^1$	$f_{6,tl}^1$
ES	$8.59 \pm 1.6E-2$	$8.61 \pm 2.0E-2$	$8.65 \pm 2.4E-2$
RL	$2.18 \pm 9.9E-2$	$8.50 \pm 2.3E-2$	$8.58 \pm 1.9E-2$
FNN	$3.18 \pm 9.2E-3$	$2.96 \pm 2.5E-2$	$2.89 \pm 2.8E-2$
AM	$8.18 \pm 1.0E-1$	$8.63 \pm 2.2E-2$	$8.64 \pm 1.9E-2$
Ens	<b><math>9.39E-1 \pm 6.3E-2</math></b>	<b><math>2.81 \pm 2.0E-2</math></b>	<b><math>2.75 \pm 2.8E-2</math></b>
probs	$f_{7,c}^1$	$f_{7,ur}^1$	$f_{7,ur}^1$
ES	$3.96 \pm 6.4E-3$	$3.99 \pm 1.1E-2$	$4.00 \pm 7.8E-3$
RL	$2.39 \pm 3.3E-2$	$3.93 \pm 9.4E-3$	$3.96 \pm 9.2E-3$
FNN	$1.92 \pm 4.3E-3$	$1.89 \pm 7.2E-3$	$1.89 \pm 1.6E-2$
AM	$3.87 \pm 3.0E-2$	$3.99 \pm 8.4E-3$	$4.00 \pm 9.1E-3$
Ens	<b><math>1.50 \pm 4.8E-2</math></b>	<b><math>1.86 \pm 6.3E-3</math></b>	<b><math>1.86 \pm 1.3E-2</math></b>

Scalability is a key issue of the proposed strategies. Many real-world problems have a large parameter vector or search space. In these cases, it would be difficult and time-consuming to train an FNN to fit the relation between parameter vectors and “good solutions”. Therefore, it could be unfeasible or unaffordable to use the proposed strategies.

One solution of this issue is using a pretrained FNN instead of training online. With enough data, many modern machine learning algorithms can train models that well fit very complex functions with high dimensions. To simulate this situation, a pretrained scheme is tested. With this scheme, all the strategies are provided with 200 data entries at the beginning. Each entry is composed of a uniformly sampled parameter vector and a corresponding solution obtained by the MVOA. For AM, RL and Ens, the memory sets are initialized with the provided data entries. For FNN and Ens, *net* is trained in advance, and never updated during optimization.

From Table 6, it can be seen that RL is likely a scalable strategy because its running time increases little with the dimension. FNN and Ens need more time than the others, and with the dimension increases, their running time increases the most. These results indicate that the scalability of FNN and Ens may not be very competitive. However, with the pretrained scheme, the running time of FNN and Ens on 10D problems can be reduced to the amount close to that on 2D problems.

In Table 7, with the pretrained scheme, the ranking of the algorithms on 10D DOPs is similar to that on 2D and 5D problems. Furthermore, in some cases, the superiorities of FNN and Ens are even expanded. These results indicate that with the pretrained scheme, FNN and Ens could also be scalable.

## 8. Conclusion and future works

To the best of our knowledge, this is the first paper devoted to general purpose DOPs with observable parameters in EDO. We suggested enhancing the optimization performance by learning from the observable parameters. We also proposed three learning strategies: rote learning, fitting with a feedforward neural network and an ensemble strategy. Moreover, we proposed a set of test problems. In the experiments, we compared the proposed strategies with two existing representative algorithms, and the results demonstrated the effectiveness of the proposed strategies.

We compared the strategies for different relation functions, dynamic drivers and database quality. It can be concluded that the performance of the rote learning strategy is comparatively stable. Compared with the ES and AM algorithms, rote

learning achieved clear improvements in all test cases. Compared with rote learning, the performance of the FNN is dependent on the database quality and problem properties. However, for random problems with one optimal solution for one parameter vector, the FNN shows the potential to outperform rote learning. The proposed ensemble method performs best on all of the test problems, and it can be concluded that it has successfully combined the advantages of rote learning and an FNN. These results demonstrate that an evolutionary algorithm enhanced by learning from observable parameters is a promising approach to solving gray-box DOPs.

Although we have shown that the proposed strategies are effective for gray-box DOPs, there are still some issues that require further study:

- There are many real-world dynamic optimization problems with observable parameters. The application of the proposed strategies to these problems is an important research direction in the future.
- Many successful machine learning methods have been developed. Leveraging the advantage of more efficient machine learning methods in evolutionary algorithms for gray-box DOPs is a valuable research direction.
- The method used to select individuals for the training set greatly affects the performance of the learning strategies, and this is an important issue that is not addressed in depth in this paper.
- If only part of the parameters can be observed, it is a challenging task to learn valuable information from them.

### Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

We further confirm that any aspect of the work covered in this manuscript that has involved either experimental animals or human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript.

We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author and which has been configured to accept email from wjliao@ustc.edu.cn.

### Acknowledgments

We extend our gratitude to Dr. Huang, Dr. Qin and Prof. Suganthan for the codes of SaDE, and to Prof. István Erlich and his colleagues for the codes of MVOA.

The details of the test problems are given below, which are from the CEC'2005 and CEC'2006 optimization competitions [18,36]. Both  $f_1$  and  $f_2$  are constrained optimization problems, and the others are not.  $D$  represents the dimensions. For problems from  $f_3$  to  $f_7$ ,  $\mathbf{x} \in [-100, 100]^D$ , and the global optimum is located at the origin point  $\mathbf{x}^* = \mathbf{0}$ , and  $f(\mathbf{x}^*) = 0$ .

$f_1$ : **g08 Function**

$$f_1(\mathbf{x}) = -\frac{\sin^2(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

subject to

$$g_1(\mathbf{x}) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(\mathbf{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

$$0 \leq x_1, x_2 \leq 10$$

The optimum is located at  $\mathbf{x}^* = (1.22797, 4.24537)$  where  $g08(\mathbf{x}^*) = -0.095825$ . The ratio of the feasible area is 0.8560%.

$f_2$ : **g24 Function**

$$f_2(\mathbf{x}) = -x_1 - x_2$$

subject to:

$$g_1(\mathbf{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$g_2(\mathbf{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

$$0 \leq x_1 \leq 3, \quad 0 \leq x_2 \leq 4$$

The optimum is located at  $\mathbf{x}^* = (2.32952, 3.178493)$  where  $g_2(\mathbf{x}^*) = -5.508$ . The ratio of the feasible area is 79.6556%.

### $f_3$ : Sphere Function

$$f_3(\mathbf{x}) = \sum_{i=1}^D x_i^2$$

### $f_4$ : Schwefel's Problem 1.2

$$f_4(\mathbf{x}) = \sum_{i=1}^D \left( \sum_{j=i}^i x_j \right)^2$$

### $f_5$ : Schwefel's Problem 1.2 with Noise in Fitness

$$f_5(\mathbf{x}) = \sum_{i=1}^D \left( \sum_{j=i}^i x_j \right)^2 * (1 + 0.4|N(0, 1)|)$$

### $f_6$ : Rosenbrock's Function

$$f_6(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

### $f_7$ : Rastrigin's Function

$$f_7(\mathbf{x}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

## References

- [1] X. Bai, R. Padman, Tabu search enhanced Markov blanket classifier for high dimensional data sets, in: *The Next Wave in Computing, Optimization, and Decision Technologies*, Springer US, 2006, pp. 337–354.
- [2] A. Benavoli, G. Corani, J. Demsar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, *J. Mach. Learn. Res.* 18 (1) (2017) 1–36.
- [3] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, *IEEE Trans. Evolution. Comput.* 10 (4) (2006) 459–472.
- [4] P.A.N. Bosman, Learning, anticipation and time-deception in evolutionary online dynamic optimization, in: *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, ACM Press, New York, USA, 2005, pp. 39–47.
- [5] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, IEEE, 1999, pp. 1875–1882.
- [6] C. Bu, W. Luo, L. Yue, Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies, *IEEE Trans. Evolution. Comput.* 21 (1) (2017) 14–33.
- [7] B. Calvo, J. Ceberio, J.A. Lozano, Bayesian inference for algorithm ranking analysis, in: *Proceedings of the 20th Annual Conference on Genetic and Evolutionary Computation*, ACM Press, New York, New York, USA, 2018, pp. 324–325.
- [8] B. Calvo, G. Santafé, Scamp: statistical comparison of multiple algorithms in multiple problems, *R J.* 8 (1) (2016) 248–256.
- [9] L. Cao, L. Xu, E.D. Goodman, A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems, *Inform. Sci.* 453 (2018) 463–485.
- [10] J.G. Carbonell, R.S. Michalski, T.M. Mitchell, An overview of machine learning, in: *Machine learning: an artificial intelligence approach*, TIOGA Publishing Co., 1983, pp. 3–23.
- [11] I. Erlich, G.K. Venayagamoorthy, N. Worawat, A mean-variance optimization algorithm, *Proceed. 2010 IEEE Cong. Evolution. Comput.* (2010) 18–23.
- [12] I. Hatzakis, D. Wallace, Dynamic multi-objective optimization with evolutionary algorithms : a forward-looking approach, in: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 1201–1208.
- [13] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neur. Netw.* 2 (1989) 359–366.
- [14] V.L. Huang, A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for constrained real-parameter optimization, in: *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, IEEE, Vancouver, BC, Canada, 2006, pp. 17–24.
- [15] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments—a survey, *IEEE Trans. Evolution. Comput.* 9 (3) (2005) 303–317.
- [16] J.K. Kordestani, A.E. Ranginkaman, M.R. Meybodi, P. Novoa-Hernández, A novel framework for improving multi-population algorithms for dynamic optimization problems: a scheduling approach, *Swarm Evolution. Comput.* 44 (2019) 788–805.
- [17] C. Li, M. Mavrovouniotis, S. Yang, X. Yao, Benchmark generator for the IEEE WCCI-2014 competition on evolutionary computation for dynamic optimization problems, Technical Report, De Montfort University, 2013.
- [18] J.J. Liang, T.P. Runarsson, M. Clerc, P.N. Suganthan, C.A.C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, *J. Appl. Mech.* 41 (8) (2006) 8–31.
- [19] Z. Liang, S. Zheng, Z. Zhu, S. Yang, Hybrid of memory and prediction strategies for dynamic multiobjective optimization, *Inform. Sci.* 485 (2019) 200–218.
- [20] W. Luo, X. Lin, T. Zhu, P. Xu, A clonal selection algorithm for dynamic multimodal function optimization, *Swarm Evolution. Comput.* in press (2019).
- [21] W. Luo, J. Sun, C. Bu, H. Liang, Species-based particle swarm optimizer enhanced by memory for dynamic optimization, *Appl. Soft Comput.* 47 (2016) 130–140.
- [22] W. Luo, J. Sun, C. Bu, R. Yi, Identifying species for particle swarm optimization under dynamic environments, in: *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, IEEE, 2018, pp. 1921–1928.

- [23] M. Mavrouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: algorithms and applications, *Swarm Evolution. Comput.* 33 (2017) 1–17.
- [24] M. Mavrouniotis, S. Yang, Ant algorithms with immigrants schemes for the dynamic vehicle routing problem, *Inform. Sci.* 294 (October) (2014) 456–477.
- [25] R.M. May, Simple mathematical models with very complicated dynamics, *Nature* 261 (5560) (1976) 459–467.
- [26] A. Muruganatham, K.C. Tan, P. Vadakkepat, Evolutionary dynamic multiobjective optimization via kalman filter prediction, *IEEE Trans. Cybernet.* 46 (12) (2016) 2862–2873.
- [27] T. Nguyen, Z. Yang, S. Bonsall, Dynamic time-linkage problems—the challenges, in: *IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future*, 2012, pp. 1–6.
- [28] T.T. Nguyen, Continuous dynamic optimisation using evolutionary algorithms, University of Birmingham, 2010 Phd thesis.
- [29] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, *Swarm Evolution. Comput.* 6 (2012) 1–24.
- [30] T.T. Nguyen, X. Yao, Benchmarking and solving dynamic constrained problems, in: *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 690–697.
- [31] T.T. Nguyen, X. Yao, Continuous dynamic constrained optimization - the challenges, *IEEE Trans. Evolution. Comput.* 16 (6) (2012) 769–786.
- [32] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, *IEEE Trans. Evolution. Comput.* 10 (4) (2006) 440–458.
- [33] H. Richter, S. Yang, Learning behavior in abstract memory schemes for dynamic optimization problems, *Soft Comput.* 13 (12) (2009) 1163–1173.
- [34] C. Rossi, M. Abderrahim, J.C. Díaz, Tracking moving optima using kalman-based predictions, *Evolution. Comput.* 16 (1) (2008) 1–30.
- [35] A. Simões, E. Costa, Prediction in evolutionary algorithms for dynamic environments using markov chains and nonlinear regression, in: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, 2009, pp. 883–890.
- [36] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report, KanGAL report, 2005.
- [37] W.J. Tang, M.S. Li, Q.H. Wu, J.R. Saunders, Bacterial foraging algorithm for optimal power flow in dynamic environments, *IEEE Trans. Circuit Syst.* 55 (8) (2008) 2433–2442.
- [38] R. Tinós, S. Yang, A self-organizing random immigrants genetic algorithm for dynamic optimization problems, *Genet. Program. Evol. Mach.* 8 (3) (2007) 255–286.
- [39] R.K. Ursem, T. Krink, M.T. Jensen, Z. Michalewicz, Analysis and modeling of control tasks in dynamic systems, *IEEE Trans. Evolution. Comput.* 6 (4) (2002) 378–389.
- [40] S. Yang, X. Yao, Population-based incremental learning with associative memory for dynamic environments, *IEEE Trans. Evolution. Comput.* 12 (5) (2008) 542–561.
- [41] D. Yazdani, M.N. Omidvar, J. Branke, T.T. Nguyen, X. Yao, Scaling up dynamic optimization problems: a divide-and-Conquer approach, *IEEE Trans. Evolution. Comput.* in press (2019).
- [42] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization, in: *Evolutionary Multi-Criterion Optimization*, 2007, pp. 832–846.
- [43] T. Zhu, Y. Hao, W. Luo, H. Ning, Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, *Appl. Soft Comput.* 67 (2018) 812–821.
- [44] T. Zhu, W. Luo, C. Bu, L. Yue, Accelerate population-based stochastic search algorithms with memory for optima tracking on dynamic power systems, *IEEE Trans. Power Syst.* 31 (1) (2016) 268–277.
- [45] T. Zhu, W. Luo, L. Yue, Dynamic optimization facilitated by the memory tree, *Soft Comput.* 19 (3) (2015) 547–566.
- [46] J. Zou, Q. Li, S. Yang, J. Zheng, Z. Peng, T. Pei, A dynamic multiobjective evolutionary algorithm based on a dynamic evolutionary environment model, *Swarm Evolution. Comput.* 44 (2019) 247–259.