Evolutionary Optimization of High-Dimensional Multi- and Many-Objective Expensive Problems Assisted by a Dropout Neural Network

Dan Guo, Xilu Wang, Kailai Gao, Yaochu Jin, *Fellow, IEEE*, Jinliang Ding, *Senior Member, IEEE*, and Tianyou Chai, *Fellow, IEEE*

***Abstract*—Gaussian processes are widely used in surrogate-assisted evolutionary optimization of expensive problems mainly due to the ability to provide a confidence level of their outputs, making it possible to adopt principled surrogate management methods such as the acquisition function used in Bayesian optimization. Unfortunately, Gaussian processes become less practical for high-dimensional multi- and many-objective optimization as their computational complexity is cubic in the number of training samples. In this paper, we propose a computationally efficient dropout neural network (EDN) to replace the Gaussian process and a new model management strategy to achieve a good balance between convergence and diversity for assisting evolutionary algorithms to solve high-dimensional multi- and many-objective expensive optimization problems. While the conventional dropout neural network needs to save a large number of network models during the training for calculating the confidence level, only one single network model is needed in the EDN to estimate the fitness and its confidence level by randomly ignoring neurons in both training and testing the neural network. Extensive experimental studies on benchmark problems with up to 100 decision variables and 20 objectives demonstrate that, compared to state-of-the-art, the proposed algorithm is not only highly competitive in performance but also computationally more scalable to high-dimensional many-objective optimization problems. Finally, the proposed algorithm is validated on an operational optimization problem of crude oil distillation units, further confirming its capability of handling expensive problems given a limited computational budget.**

***Index Terms*—Neural network, dropout, Gaussian process, multi- and many-objective optimization, surrogate-assisted evolutionary algorithm, Bayesian optimization, expensive optimization**

I. Introduction

This work was supported in part by a Royal Society International Exchanges Program under No. IEC\NSFC\170279, in part by the National Key Research and Development Program of China under Grant 2018YFB1701104, the National Natural Science Foundation of China under Grant 61590922, 61525302, 61988101 the Xingliao Plan of Liaoning Province under Grant XLYC1808001 and the Science & Technology program of Liaoning Province under Grant 2020JH2/10500001 and 2020JH1/10100008. (*D. Guo and X. Wang contribute equally. Corresponding authors: Yaochu Jin; Jinliang Ding.*)

D. Guo was with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China. She is now with the Science and Technology on Electromagnetic Scattering Laboratory, Beijing 100854, China (e-mail: guodan717@163.com). K. Gao, J. Ding and T. Chai are with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: kailaigao@163.com; jlding@mail.neu.edu.cn; tychai@mail.neu.edu.cn).

X. Wang and Y. Jin are with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K. Y. Jin is also with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: xilu.wang@surrey.ac.uk; yaochu.jin@surrey.ac.uk).

MANY real-world optimization problems have multiple conflicting objectives to be optimized simultaneously, which are referred to as multi-objective optimization problems (MOPs). Evolutionary algorithms (EAs) have been widely employed to effectively solve MOPs, and various multi-objective EAs (MOEAs) have been proposed and shown to perform very well on a wide range of MOPs [1], [2]. However, MOEAs typically require a large number of fitness evaluations (FEs) to obtain a set of satisfactory solutions, and unfortunately, FEs in many real-world problems can be computationally very intensive or highly costly [3]–[5]. Examples include airfoil design [4], manufacturing engineering [6], the design of crude oil distillation units [7], and furnace optimization [8], where only a small number of FEs is affordable. Such expensive MOPs have posed great challenges to MOEAs due to the limited evaluation budget. Therefore, data-driven surrogate-assisted evolutionary algorithms (SAEAs) have become popular to reduce the number of required expensive FEs in EAs [9], [10]. Generally, SAEAs construct surrogate models to approximate the original expensive objective functions, then the surrogates can be used to evaluate part of the candidate solutions, thereby reducing the computational cost. To enhance their prediction performance, new samples should be identified to be evaluated by the true objective functions for updating the surrogates, which is referred to as model management [11]. SAEAs have been demonstrated to be able to find better solutions than their baseline EAs, when a limited number of expensive real FEs is allowed. Comprehensive reviews of SAEAs can be found in [12], [13].

Recently, complex MOPs with more than three objectives, known as many-objective optimization problems (MaOPs) have received increased attention [14], [15]. Various approaches to solving MaOPs have been proposed, which can be roughly divided into three categories. The first category of the methods aims to enhance the selection pressure towards the Pareto front (PF) by modifying the dominance relations. For example, ϵ-dominance [16], L-optimality [17], and preference order ranking [18]. Another group of ideas is known as decomposition-based approaches. By using a set of reference points, an MaOP can be decomposed into many single-objective subproblems. A representative work is the decomposition-based MOEA (MOEA/D) [19], which has been popular for handling both MOPs and MaOPs. Furthermore, an extension of NSGA-II [2] using a set of reference points to divide the objective space into a number of small subspaces,

termed NSGA-III, is proposed for handling MaOPs [20]. The third category of methods is based on a performance indicator that is able to account for both convergence and diversity properties of a solution set. Along this line of research, typical indicators including hypervolume [21], ϵ-indicator [22] and $R2$ indicator [23]. Although they have achieved great success on small to medium scale MOPs and MaOPs, the performance of MOEAs often seriously deteriorates for high-dimensional problems. For this reason, plenty of research efforts have been dedicated to extending MOEAs to high-dimensional MOPs/MaOPs [24], [25], many of which are based on the cooperative coevolution framework [26], [27].

It is not uncommon that high-dimensional MOPs/MaOPs also involve the expensive FEs as mentioned before, making it increasingly important to improve the scalability of SAEAs. However, most current SAEAs are not well scalable to expensive high-dimensional MOPs/MaOPs. Only recently have a few efforts been dedicated to developing surrogate techniques for tackling expensive high-dimensional single- or multi-objective problems, which are briefly reviewed below.

- **SAEAs for expensive MaOPs**: A surrogate-assisted reference vector guided EA, known as K-RVEA, was proposed to handle MaOPs, where Gaussian processes (GPs) are employed to approximate each expensive objective function [28]. K-RVEA balances the convergence and diversity by leveraging the uncertainty provided by GPs and limits the number of training data to control the computational complexity. By contrast, a classification surrogate based EA (CSEA) was proposed in [29], where a feedforward neural network is trained to learn the dominance relationship between the candidate solutions and a set of selected reference solutions, thereby requiring only one surrogate to solve MaOPs. The effectiveness of most SAEAs has been validated only on small to medium scale MaOPs, where the number of decision variables ranges from 10 to 30, and up to 15 decision variables for most GP-based SAEAs.
- **SAEAs for high-dimensional expensive single- and multi- objective problems**: Few GP-based SAEAs have been developed for high-dimensional problems due to the high computational complexity. A common idea is to replace GPs with approximation methods whose complexity is more scalable to the number of training samples. For example, Sun *et al.* employ particle swarm optimizers assisted by a radial-basis-function (RBF) network and fitness inheritance to solve expensive high-dimensional single-objective problems [30]. A heterogeneous ensemble consisting of a least square support vector machine (SVM) and two RBF networks is constructed to replace GPs in [31]. This way, the uncertainty of the approximated fitness can be obtained by calculating the variance of the outputs of the ensemble members. Although the heterogeneous ensemble has shown to be effective [31], the SVM used in the ensemble is single-output model, making it hardly scalable to the number of objectives. Furthermore, the success in generating a diverse and accurate ensemble heavily depends on feature selection and feature extraction, which may be problem-dependent.

GPs are a common choice for surrogate models as they can provide a confidence level of their predictions, which enables us to use mathematically sound model management techniques known as infill criteria or acquisition functions. Popular GP assisted MOEAs include ParEGO [32], MOEA/D-EGO [33] and K-RVEA [28], among many others. However, GP-based SAEAs that perform well on low-dimensional MOPs will become computationally intractable when they are used to solve high-dimensional MOPs/MaOPs, mainly due to the following three reasons [34]. First, the computational complexity of GPs will become prohibitively high for large datasets as their computational complexity is $\mathcal{O}(n^3)$, where n is the number of training samples [35]. The higher the dimensionality of the decision space is, the more training data is required for constructing GPs in SAEAs to obtain sufficiently good approximation accuracy. Second, since GPs are single-output regression models [36], the number of GP models we need equals to the number of objectives if the GPs are used to approximate the objective functions, making GPs unpractical for handling MaOPs. Thirdly, GP models must be completely rebuilt when new samples are included in the training data [36], which is very undesirable since the GP model needs to be frequently updated during the optimization.

Due to the above challenges, most existing SAEAs are devised for low-dimensional expensive single- or multi- objective optimization problems and their applications are severely restricted in the scope of expensive MOPs with small-scale decision variables [37], [38].

It is worth noting that in the field of machine learning, deep NNs have been used as a computationally efficient replacement of GP models [39]. Moreover, dropout is a commonly used technique in deep NNs to alleviate over-fitting [40], [41]. Gal and Ghahramani threw light on dropout properties by interpreting it as a Bayesian model, resulting from its capability of representing model uncertainty in deep learning [42]. Further, dropout deep NNs with an arbitrary depth and nonlinearities have been proposed and mathematically proved that they are equivalent to an approximation of the probabilistic deep Gaussian process model [42]. Existing dropout deep NNs, however, is computationally intensive in the training stage, because a large number of NNs needs to be stored and updated [43].

Given that expensive high-dimensional MOPs/MaOP are widely seen in the real world, this work aims to develop an SAEA with a surrogate that is functionally similar to but computationally more scalable than the GP model to solve high-dimensional MOPs/MaOPs. To the best of our knowledge, this is the first time that surrogates have been adopted for tackling expensive high-dimensional MOPs/MaOPs, where complexities resulted from a large search space together with a large number of objectives. Main contributions of this work are summarized as follows.

- An efficient dropout NN, called EDN, is proposed as a computationally scalable alternative of the GP model for assisting the solution of expensive high-dimensional MOPs/MaOPs. Different from the original dropout technique, EDN uses dropout not only at training phase, but

at the test phase as well. As a result, only one single network model is needed in EDN, which is different from traditional SAEAs that adopt multiple surrogates for estimating a degree of uncertainty [31], [44] or for robust fitness estimation [45]. It is worth noting that the original dropout technique cannot be directly applied to SAEAs due to its highly time-consuming training process, resulting from the storage and update of many NNs required for the estimations of fitness and uncertainty.

- A model management strategy for choosing new samples and training data is suggested and integrated into an indicator-based MOEA with reference point adaptation (AR-MOEA) [46] to achieve a better balance between exploration and exploitation during the search, making the proposed algorithm well suited for both multi- and many-objective optimization. Specifically, two sets of reference vectors are introduced based on AR-MOEA to estimate the quality of the samples obtained so far, and to determine whether convergence or the diversity should be prioritized when selecting new samples.
- The proposed EDN-assisted AR-MOEA, termed EDN-ARMOEA, is compared with state-of-the-art surrogate-assisted MOEAs on high-dimensional MOPs and MaOPs. Empirical results on two suites of test problems and a real-world operational optimization problem of crude oil distillation units demonstrate EDN-ARMOEA is very competitive in performance and computationally much more scalable than existing surrogate-assisted MOEAs.

The rest of this paper is organized as follows. Section II provides the background of the work, including a problem definition and an introduction to AR-MOEA and the dropout NN. The details of the proposed EDN-ARMOEA are elaborated in Section III. The results of all comparative and ablation studies, as well as sensitivity analysis are presented in Section IV. Finally, conclusions and future work are given in Section V.

II. Background

In this section, we at first give a brief problem definition, then introduce the main components of AR-MOEA. Finally, dropout neural networks are briefly described.

A. Problem Definition

In this work, we consider the following expensive high-dimensional multi- and many-objective optimization problems:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{x} \in X \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ is the decision vector with d decision variables, X denotes the decision space, the objective vector $\mathbf{f}$ consists of $m(m > 2)$ objectives. Note that for MaOPs, the number of objectives m is larger than 3. Moreover, the objective functions are black-box that can be evaluated by either time-consuming numerical simulations, or costly physical experiments. Hence, surrogates must be built based on data collected via numerical simulations of experiments to solve the optimization problem [9]. Note that in the context of surrogate-assisted evolutionary optimizations, problems with $d > 30$ is known as high-dimensional problems and the majority of the existing SAEAs can solve problems with less than 30 decision variables. It should also be noted that for MOPs/MaOPs, there exists a set of trade-off Pareto optimal solutions, known as the Pareto set (PS), rather than a single solution that optimizes all objectives. The projection of the PS in the objective space is known as the PF.

B. AR-MOEA

The performance of most MOEAs is heavily dependent on the shape of the PF. AR-MOEA aims to remedy this issue by developing a versatile algorithm whose performance is less sensitive to various MOPs or MaOPs having disconnected, degenerate, concave or biased PFs [46]. AR-MOEA adopts the enhanced inverted generational distance (IGD-NS) indicator [47] to calculate the fitness value of an individual in mating and environmental selection. The reference points (reference solutions) for calculating the IGD-NS are composed of two parts, one containing uniformly distributed solutions and the other are solutions stored in an external archive.

The general framework of AR-MOEA is described in Algorithm 1. The two main components of AR-MOEA, i.e., the IGD-NS indicator and the reference point adaptation method will be briefly described in the following.

Algorithm 1 AR-MOEA

Input: Population size N; Maximum iterations $iter$;
Output: Final population P;
1: Create an initial population P_1 of size N;
2: Uniformly sample reference points W from a unit hyperplane;
3: $[A_1, R_1] \leftarrow UpdateRefPoint(P_1, W)$;
4: **for** $t = 1$ to $iter$ **do**
5: Use the IGD-NS values with respect to R_t to select solutions from P_t for a mating pool;
6: Create offspring O_t using simulated binary crossover and polynomial mutation;
7: $[A_{t+1}, R_{t+1}] \leftarrow UpdateRefPoint([A_t; O_t], W)$;
8: Use the IGD-NS values with respect to R_{t+1} to select the population P_{t+1} from P_t and O_t for next generation;
9: **end for**

1) IGD-NS Indicator: IGD with noncontributing solution detection, termed IGD-NS, is an enhanced IGD indicator [47]. Similar to IGD, IGD-NS evaluates the quality of a set of solutions with respect to both convergence and diversity. Furthermore, IGD-NS takes the non-contributing solutions, which are ignored in IGD, into consideration, thereby providing a more precise evaluation of a given solution set. A solution is called non-contributing if it is not the nearest neighbor of any reference point. The IGD-NS indicator is defined as follows:

$$\text{IGD-NS}(Q, R) = \sum_{r \in R} \min_{q \in Q} dis(q, r) + \sum_{q' \in Q'} \min_{r \in R} dis(q', r) \tag{2}$$

where function $dis(\cdot)$ denotes the Euclidean distance between two solutions in objective space, and Q, $Q'(Q' \in Q)$ and R

represent the non-dominated solution set, the non-contributing solution set, and the reference point set, respectively. The first term calculates the sum of the minimum distance from each non-dominated solution to the points in R, and the second term calculates the sum of the minimum distance from each non-contributing solution to the points in R, aiming to minimise the number of non-contributing solutions. The fitness of a candidate solution p is calculated by IGD-NS$(P\backslash\{p\}, R)$, where "\" denotes the complementary set. Therefore, in AR-MOEA, a solution with a larger IGD-NS value will be the winner during the tournament selection and the population truncation. The efficient non-dominated sorting method [48] is first used in the environmental selection to screen solutions in the combined population of parents and offspring, then the solutions from the last known front are selected based on the IGD-NS indicator.

As the different ranges of different objectives can bias the uniform distribution of the solutions, the objective values of solutions in P and Q are normalized by subtracting z^*, following the calculation of the ideal point z^* and nadir point z^{nad}. Note that z^* and z^{nad} can be obtained by minimizing and maximizing each objective individually. Subsequently, the reference point set R is adjusted for a fair calculation of the distance between the solutions. More specifically, the solution $q \in Q$ having the minimum perpendicular distance to vector $\overrightarrow{z^*r}$ is detected by minimizing $\|\mathbf{f}(q)\| \sin(\overrightarrow{z^*r}, \mathbf{f}(q))$, where $\mathbf{f}$ is the mapping from the decision space to the objective space. Then, each $r \in R$ is adjusted to the orthogonal projection of $\mathbf{f}(q)$ on vector $\overrightarrow{z^*r}$, described as follows:

$$r_i' = \frac{r_i}{\|r\|} \times \|\mathbf{f}(q)\| \cos(\overrightarrow{z^*r}, \mathbf{f}(q)), \quad i = 1, 2, \ldots, m \tag{3}$$

where m is the number of objectives.

2) Reference Point Adaptation: Before the adaptation of the reference points R_t, the external archive A_t is updated to A_{t+1} according to the fixed reference points W and the offspring population O_t. The update of A_t is also based on the definition of contribution in IGD-NS. Firstly, we delete the duplicate and dominated solutions in A_t. Next, the contributing solutions in A_t with respect to W, denoted as A_t^c, are copied to A_{t+1}. Lastly, the point p from $A_t \setminus A_{t+1}$ with the maximum value of $\min_{q \in A_{t+1}} \arccos(\mathbf{f}(p), \mathbf{f}(q))$ is continuously chosen to fill A_{t+1} until the size of A_{t+1} reaches $\min(3|W|, |A_t|)$. This way, the solutions having the maximum angles to the selected solutions can be added to the archive to ensure a good distribution.

After A_{t+1} is obtained, the adapted reference points R_t are updated to R_{t+1}. Firstly, the fixed reference points W closest to the any solution in $A_t'^c$, are identified and denoted as the valid reference point set W^v. W^v are copied to R_{t+1}. Then, the point q from $A_{t+1} \setminus R_{t+1}$ with the maximum value of $\min_{r \in R_{t+1}} \arccos(\mathbf{f}(q), r)$ is continuously chosen to fill R_{t+1} until the size of R_{t+1} reaches $\min(|W|, |W^v| + |A_{t+1}|)$. The requirement on the minimal angle of filling points can ensure that the solutions in A_{t+1} and the reference points in R_{t+1} distribute evenly.

Since the external archive A is able to reflect the geometry of PFs, the adaptive reference points in R guided by A and the uniformly distributed reference points W can adapt to various shapes of the PFs. Empirical results in [46] demonstrate that the parameter-free method for reference point adaptation significantly contributes to the robust performance of AR-MOEA. The procedure of the reference point adaptation is presented in Algorithm S1 in the Supplementary material. We use the notation 'S' to indicate tables, figures and algorithms in the Supplementary materials in order to avoid confusion.

C. Dropout Neural Networks

Dropout has been proposed to alleviate overfitting problems for NNs by randomly removing neurons along with their connections in a NN [40]. Studies such as [43], [49] have suggested that dropout can be regarded as an effective regularization method. Randomly dropping out neurons can discourage excessive co-adaptation of units, which can be interpreted as adding noise to make the units robust [41]. We consider the situation when dropout is applied to NNs containing multiple fully-connected layers.

Note that neurons are dropped with a specified probability at the training phase although they are present at test phase [40]. Assume $\mathbf{d}_l$ is a row vector for a layer l ($l \in [1, L]$) of a NN, and each of its elements is sampled from a Bernoulli distribution:

$$d_l \sim Bernoulli(p), \quad \forall d_l \in \mathbf{d}_l \tag{4}$$

where $0.5 \leq p < 1$. With dropout, the output $\mathbf{y}_l$ of layer l becomes

$$\hat{\mathbf{y}}_l = f_l\left(\frac{1}{p}(\mathbf{x}_l \circ \mathbf{d}_l)\mathbf{W}_l + \mathbf{b}_l\right) \tag{5}$$

where $\circ$ denotes the Hadamard product, and f_l is the activation function of layer l. $\mathbf{x}_l$, $\mathbf{W}_l$ and $\mathbf{b}_l$ are the D-dimensional input vector, the $D \times K$ weight matrix and the K-dimensional bias vector of layer l, respectively. The unsuppressed neurons of $\mathbf{x}_l$ are scaled by $\frac{1}{p}$ to make sure that the expected output of the NN with dropout will be the same as that of the NN without dropout. $\mathbf{d}_l$ is regenerated for different training cases during the forward pass. $\mathbf{W}_l$ is updated by backpropagation, which is the same as that in the standard NNs. During the backward pass for a training case, the weights in $\mathbf{W}_l$ connected to the suppressed neurons of $\mathbf{x}_l$ are not updated. As every unit has two choices, i.e., a dropout NN with n units is an ensemble of 2^n pruned NNs, which explains why dropout can improve the generalization capability of NNs.

In the original dropout technique, no suppressing takes place in prediction. The output of the dropout NN at a new input $\mathbf{x}_{new}$ is

$$\hat{\mathbf{y}}_L = f_L\left(\mathbf{x}_L(\mathbf{x}_{new}) \times \mathbf{W}_L + \mathbf{b}_L\right). \tag{6}$$

To estimate the mean $\hat{\mathbf{y}}_{new}$ and variance $\hat{\mathbf{s}}_{new}^2$ of a given new decision variable, T sets of vectors of realisations from the Bernoulli distribution are sampled, resulting in T NNs. This is equivalent to performing T stochastic forward passes through the network and calculating the mean and the variance of the

output of each NN. This way, $\hat{\mathbf{y}}_{new}$ and $\hat{\mathbf{s}}^2_{new}$ of the output for $\mathbf{x}_{new}$ can be expressed by [42], [43]:

$$\begin{aligned}\hat{\mathbf{y}}_{new} &= \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_L^t \\ \hat{\mathbf{s}}^2_{new} &= \frac{1}{T}\sum_{t=1}^{T}(\hat{\mathbf{y}}_L^t)^{\mathrm{T}}\hat{\mathbf{y}}_L^t - \hat{\mathbf{y}}_{new}^{\mathrm{T}}\hat{\mathbf{y}}_{new} + \tau^{-1}\mathbf{1}\end{aligned} \tag{7}$$

where $\tau = \frac{pl_p^2}{2N\lambda}$ and $\mathbf{1}$ denotes an N-dimensional row vector of ones. l_p is the prior length-scale, and λ and N represent the hyperparameter for weight decay and the number of training data, respectively. Every several or dozens of forward-backward passes during the training, the output of the dropout NN is saved for the variance prediction. It should be pointed out that the original dropout technique cannot be directly applied in SAEAs because the training process of the dropout NNs is highly time-consuming. If the original dropout NN is used as the surrogate in SAEAs, NNs at T time instants need to be stored for the estimations of fitness and uncertainty. Every time when new samples are included in the training data during the optimization, all T NN models will be updated.

III. An Efficient Dropout Neural Network-Assisted AR-MOEA

In this section, the framework of the proposed EDN-ARMOEA is introduced at first. Then, the proposed model management is given in terms of the selection of new samples and training data, respectively. Finally, the mechanism of the proposed efficient dropout neural networks is presented.

A. The Overall Algorithm

One main difference between the proposed EDN-ARMOEA and other GP-assisted MOEAs is that EDN-ARMOEA adopts an efficient dropout NN (EDN) to replace the GP for fitness approximation, in which the EDN estimates the uncertainty of the approximated fitness by randomly suppressing neurons during the test process. Besides, EDN-ARMOEA uses the reference vectors of AR-MOEA to strike a balance between diversity and convergence in model management.

Algorithm 2 presents the overall framework of the proposed EDN-ARMOEA. Before optimization starts, a data set of size $11d-1$ for training the EDN is generated using the Latin hypercube sampling (LHS) method, where d is the number of decision variables. The main reason for setting the offline training data size to $11d-1$ is due to the LHS method [50]. To further investigate the training data size in Gaussian process based optimization algorithms, Jones et al. [51] recommended to have a convenient, finite-decimal value for the spacing between points, thereby setting e.g., 21 points for two dimensions. This setting has been followed by many SAEAs [10], [37], [52]. Once the EDN is constructed the first time or updated during the optimization (Line 5 in Algorithm 2), AR-MORA will perform evolutionary optimization for $iter$ generations based on the objective functions approximated by the EDN (Line 6). Then, the model management strategy will select k solutions from those in the final generation ($\mathbf{x}_{pop}$, $\hat{\mathbf{y}}_{pop}$, $\hat{\mathbf{s}}_{pop}$) for real function evaluations (Line 8). Thus, k new samples are added in the database (X, Y), refer to Line 9. Then $11d-1$ samples are selected from the database (X, Y) as the training data (X_1, Y_1) so that the number of training data is limited to $11d-1$ (Line 10). Note that both the decision variables and the objective values of the training data are scaled within the range $[-1, 1]$, denoted as (X_s, Y_s), before they are used for the construction or update of the EDN (Line 5). Details of the training and test procedures of the EDN will be provided in the next Section III-C and Algorithm 3.

Algorithm 2 EDN-ARMOEA

Input: **f** (expensive functions), FE_{max} (total function evaluations), d (search dimension), $delta \in [0, 1]$ is a parameter, k (the number of real fitness evaluations in each generation), P is the population size, and $iter$ is the maximum generation)

Output: X and Y (the decision variables and objective values of final solutions);

1: $X \leftarrow LhsDesign(d, 11d-1)$, $Y \leftarrow \mathbf{f}(X)$;
2: $X_1 = X$, $Y_1 = Y$, $FE = 11d-1$;
3: **while** $FE \leq FE_{max}$ **do**
4: $(X_s, Y_s) \leftarrow Normalize(X_1, Y_1)$;
5: The EDN is initialized by $Net \leftarrow EDN(X_s, Y_s)$ or updated by $Net \leftarrow EDN(X_s, Y_s, Net)$;
6: $(\mathbf{x}_{pop}, \hat{\mathbf{y}}_{pop}, \hat{\mathbf{s}}_{pop}, rf, rf^0) \leftarrow AR\text{–}MOEA(N, iter, Net)$;
7: If there is no rb, rb is initializd by $rb = rf^0$;
8: $X_{new} \leftarrow SelectIndividual(\mathbf{x}_{pop}, \hat{\mathbf{y}}_{pop}, \hat{\mathbf{s}}_{pop}, rf, rb, delta, k)$, $Y_{new} \leftarrow \mathbf{f}(X_{new})$;
9: $X \leftarrow X \cup X_{new}$, $Y \leftarrow Y \cup Y_{new}$;
10: $(X_1, Y_1) \leftarrow SelectTrainData(X, Y)$;
11: $rb = rf$, $FE = FE + |X_{new}|$;
12: **end while**

In the following, we explain the details of two main components in the model management strategy of EDN-ARMOEA, namely, the selection of solutions for real function evaluations (Line 8) and the selection of training data (Line 10).

B. Proposed Model Management Strategy

Deciding which new samples should be collected, i.e., which candidate solutions should be evaluated by the expensive objective functions, and how to update the surrogate are known as model management in SAEAs [3], [13]. The model management is vital in SAEAs and therefore many ideas for surrogate management have been proposed. Among them, mathematically more principled methods such as infill criteria [51], also known as acquisition functions in Bayesian optimization [53], are popular in SAEAs [37]. However, the original infill criteria are meant for single-objective optimization, which cannot be directly adopted for MOPs/MaOPs. In the following, we present a new model management strategy for effectively handling high-dimensional MOPs/MaOPs.

1) Selection of solutions for expensive function evaluations: The new samples to be evaluated by the expensive objective function are chosen from the solutions in the final population of AR-MOEA, which optimizes the objective functions

approximated by the EDN (Line 6 in Algorithm 2). To start with, the solutions in the final population are categorized into k groups using the k-means clustering algorithm, aiming to select a representative subset which can serve as good new samples for updating the surrogates and achieve a balanced search towards the Pareto front.

As we mentioned before, it is well recognized that these new samples should be carefully selected due to their influence on the balance between exploitation and exploration. In SAEAs, we tune the balance between exploration and exploitation by detecting whether convergence or diversity is in demand. More specifically, the solutions contributing to the improvement of convergence should be selected for real function evaluations unless the diversity of the population must be prioritized, similar to the idea suggested in [54]. When convergence is the selection criterion, the solution having the minimum Euclidean distance to the origin in the objective space in each cluster will be selected for real function evaluations. By contrast, the solution that has the maximum mean uncertainty in each cluster will be chosen for evaluation when diversity must be promoted. The diversity of the population is measured by the ratio of $r = |W^v|/|W|$, where $|W^v|$ and $|W|$ are the number of valid reference points and total number of fixed reference points, respectively. Here, a reference point is called valid if there is a contributing solution (defined by the IGD-NS) in the population associated with the reference point. If the difference of r_{i-1} and r_i (the diversity of the population in the $i-1$th and ith generations, respectively) larger than a given threshold, i.e., $r_{i-1} - r_i > delta$, then the diversity of the population should be enhanced and therefore the most uncertain solution in each cluster should be selected for real function evaluations. Note that all objective values should be translated by the minimum objective values before calculating the Euclidean distance.

2) Selection of Training Data: The number of the training data (X_1,Y_1) is limited to $11d-1$ to reduce the computation time in updating the EDN. In selection, priority is given to the newly evaluated solutions. Firstly, the newly evaluated solutions (X_{new},Y_{new}) are added to data set (X,Y) for storage of all real evaluations. Hence, the empty (X_1, Y_1) is filled with (X_{new},Y_{new}). Subsequently, the objective value set Y of the augmented data set (X,Y) is normalized with its minimum. After that, solutions from data set ($X \setminus X_1$, $Y \setminus Y_1$) are added to (X_1, Y_1) one by one, with each chosen p solution having the maximum value of $\min_{q \in Y_1} \arccos(\mathbf{f}(p), \mathbf{f}(q))$, where $\mathbf{f}(p)$ is the objective vector of p in ($X \setminus X_1$, $Y \setminus Y_1$), and $\mathbf{f}(q)$ is the objective vector of a solution q in (X_1, Y_1). This way, the solutions that have the maximum angles to solutions in (X_1, Y_1) will be added to the training data to make sure that the training data distribute as evenly as possible in the objective space. The newly evaluated samples together with the data selected in terms of the diversity form the data set (X_1, Y_1) for training the EDN in the next generation.

C. Efficient Dropout Neural Networks

The dropout technique was originally developed for NNs containing a large number of fully-connected layers. However, such complex NNs are not suited for surrogates. Therefore, the proposed EDN contains two hidden layers [55], [56] to reduce the computational complexity while making sure that dropout can be performed on it. The first hidden layer uses the ReLU activation function, which enables fast training and is easy to implement [57]. The activation function of the second hidden layer is the Tanh function, which makes the EDN well suited for the regression of normalized training data. The mini-batch learning is adopted in the training and $batchsize$ is used to represent the number of training data in one pass. Fig. 1 shows an example of the EDN, where d (the dimensionality of decision space) and m (the number of objectives) are the number of inputs ($\mathbf{x}_s$) and outputs ($\hat{\mathbf{y}}_s$), and J and K are the number of neurons in the first and second hidden layers, respectively. The weight matrices and the bias vectors are denoted by W_1, W_2, W_3, and B_1, B_2, B_3, respectively. Finally, a symbol D in the input and first hidden layers denotes that these nodes may be dropped out.

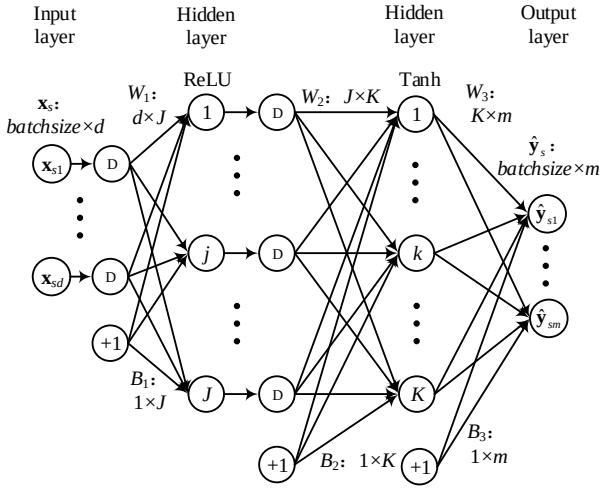


Fig. 1. The structure of the efficient dropout neural network used in EDN-ARMOEA.

The output of EDN in the forward pass for the training stage can be expressed as follows:

$$\begin{aligned} \mathbf{y}_R &= f_R\left(\frac{1}{p_I}\left(\mathbf{x}_s \circ \mathbf{d}_I\right) W_1 + B_1'\right) \\ \mathbf{y}_T &= f_T\left(\frac{1}{p_R}\left(\mathbf{y}_R \circ \mathbf{d}_R\right) W_2 + B_2'\right) \\ \hat{\mathbf{y}}_s &= \mathbf{y}_T W_3 + B_3' \end{aligned} \tag{8}$$

where $\mathbf{y}_R$ and $\mathbf{y}_T$ are the outputs of the first and second hidden layer, respectively. The elements of $\mathbf{d}_I$ (for the input layer) and $\mathbf{d}_R$ (for the first layer) are sampled from a Bernoulli distribution with parameter p_I and p_R, respectively. Matrices B_1', B_2' and B_3' are composed of $batchsize$ copies of B_1, B_2 and B_3. f_R and f_T represent the ReLU function and the Tanh function, respectively. Then the approximation errors are used in back-propagation,

$$\begin{aligned} \mathbf{e} &= \hat{\mathbf{y}}_s - \mathbf{y}_s \\ E &= \frac{1}{2}\left(\sum_{i=1}^{batchsize}\sum_{j=1}^{m} \mathbf{e}_{ij}^2 + wd \times W_{sum}\right) \end{aligned} \tag{9}$$

where $\mathbf{y}_s$ is a matrix composed of the real objective values of the decision variables $\mathbf{x}_s$, wd is the weight decay, while W_{sum} is the quadratic sum of all elements in W_1, W_2 and W_3. The update for weights and biases is based on the gradient method and the chain rule as in the standard NNs.

After the training stage is finished, the EDN will be used to evaluate the fitness of the population $\mathbf{x}_{pop}$ in AR-MOEA,

$$\hat{\mathbf{y}}_i = f_T\left(\frac{1}{p_R}\left(f_R\left(\frac{1}{p_I}\left(\mathbf{x}_{pop}\circ\mathbf{d}_I\right)\times W_1+B_1'\right)\circ\mathbf{d}_R\right)\times W_2+B_2'\right) \times W_3+B_3'. \tag{10}$$

Different from the original dropout technique, the dropout in EDN takes place not only at training time, but at test time as well. The output will be calculated for $iter_{test}$ times to approximate the fitness of new candidate solutions in the population, where $\mathbf{d}_I$ and $\mathbf{d}_R$ are regenerated in every time. Each output is mapped to its original range, and then the mean fitness estimations $\hat{\mathbf{y}}_{pop}$ of $\mathbf{x}_{pop}$ as well as the confidence level $\hat{\mathbf{s}}_{pop}$ of $\hat{\mathbf{y}}_{pop}$ can be given by

$$\begin{aligned}\hat{\mathbf{y}}_{pop} &= \frac{1}{iter_{test}}\sum_{i=1}^{iter_{test}}\hat{\mathbf{y}}_i\\ \hat{\mathbf{s}}_{pop} &= \sqrt{\frac{1}{iter_{test}}\sum_{i=1}^{iter_{test}}\hat{\mathbf{y}}_i^{\mathrm{T}}\hat{\mathbf{y}}_i-\hat{\mathbf{y}}_{pop}^{\mathrm{T}}\hat{\mathbf{y}}_{pop}}.\end{aligned} \tag{11}$$

The pseudo code describing the training and fitness approximation using the proposed EDN is listed in Algorithm 3. The update of EDN (Line 5 in Algorithm 2) is similar to the training of EDN (Lines 2-5 in Algorithm 3), except that the number of forward-backward passes is changed to $iter_r$. Note that the number of iterations $iter_r$ for updating the EDN during the optimization is typically much smaller than the number of iterations $iter_{train}$ for constructing the EDN the first time. The EDN will be used to predict the fitness and confidence level of populations in AR-MOEA (Line 6 of Algorithm 2), and the selection of solutions for real FEs (Line 8 of Algorithm 2) are based on $\mathbf{x}_{pop}$, $\hat{\mathbf{y}}_{pop}$ and $\hat{\mathbf{s}}_{pop}$ of the final population. When new samples are included in the training data during the optimization, there is only one single network model to be updated in EDN rather than multiple NN models in the conventional dropout NNs, so the proposed dropout method for estimating the confidence level is computationally much simpler than the conventional dropout NNs.

It should be noted that dropout has been shown effective in alleviating overfitting in deep NNs. In addition, a theoretical framework is developed in [42] that a dropout NN can be seen as a Bayesian network working in a similar way to a GP model [42]. It is also suggested that there is a trade-off between the complexity of the model and the accuracy in fitness and uncertainty estimation. The EDN we used in this work is relatively simple, which may reduce the accuracy in fitness and uncertainty estimation. In this case, a dropout NN can be seen as an ensemble of NNs that approximates the probability distribution described by a GP model.

Algorithm 3 EDN

Input: X_s and Y_s (training data pair), p_I and p_R (probability of retaining neurons), wd (weight decay), lr (learning rate), $iter_{train}$ (number of forward-backward passes), $batchsize$ (batch size), $iter_{test}$ (number of calculations for the tested outputs), $\mathbf{x}_{pop}$ (population of AR-MOEA)

Output: $\hat{\mathbf{y}}_{pop}$ and $\hat{\mathbf{s}}_{pop}$ (the fitness and the confidence estimations for $\mathbf{x}_{pop}$);

1: Initialize the network net. The elements of weights $W=\{W_1,W_2,W_3\}$ and biases $B=\{B_1,B_2,B_3\}$ are randomly generated within $[-0.5,0.5]$;
2: **for** $i=1$ to $iter_{train}$ **do**
3: A mini-batch of size $batchsize$ ($\mathbf{x}_s$ and $\mathbf{y}_s$) is created by randomly selecting from X_s and Y_s;
4: $(net,W,B)\leftarrow TrainNet(\mathbf{x}_s,\mathbf{y}_s,net,W,B,p_I,p_R,wd,lr)$;
5: **end for**
6: **for** $i=1$ to $iter_{test}$ **do**
7: $\hat{\mathbf{y}}_i \leftarrow TestNet(\mathbf{x}_{pop},net,W,B,p_I,p_R)$;
8: **end for**
9: Calculate $\hat{\mathbf{y}}_{pop}$ and $\hat{\mathbf{s}}_{pop}$ by equation (11).

The computational complexity of GP is $O(N^3)$ (N is the number of training data), since the inversion of an $N\times N$ correlation matrix is needed. N is often set to be linear to the search dimension in SAEAs. The computational complexity for constructing the EDN is $O(iter_{train}\cdot batchsize\cdot J\cdot K)$, where J and K are the number of neurons in the two hidden layers. A GP model is usually built for an expensive function and GP models must be rebuilt when newly sampled data are included in the training data, as mentioned in the Section I. Assume that the number of objectives is m and that the number of selection of solutions for the expensive FE is $iter_s$, then the total computational complexity of the GP model in GP-assisted MOEAs is $O(N^3\cdot m\cdot iter_s)$. While NNs are multiple-output regression models and the computational complexity for updating the EDN is $O(iter_r\cdot batchsize\cdot J\cdot K)$, where $iter_r$ is generally much smaller than $iter_{train}$, the total computational complexity of EDN in EDN-assisted MOEAs is still $O(iter_{train}\cdot batchsize\cdot J\cdot K)$. The computational complexity of AR-MOEA is $O(m\cdot P^3)$, where P is the size of the population. Since AR-MOEA is adopted as the underlying MOEA in our proposed EDN-ARMOEA, the computational complexity of EDN-ARMOEA is $O(iter_{train}\cdot batchsize\cdot J\cdot K+m\cdot P^3)$.

To summarize, the computational complexity of EDN-assisted evolutionary optimization is more scalable than GP-assisted evolutionary optimization for high-dimensional many-objective expensive problems for the following main reasons. First, the complexity of training a NN is more scalable to the increase in the number of training samples than that of training a GP. Second, updating the EDN during the optimization is computationally much more efficient than updating the GP. The same computational complexity is needed for updating the GP model every time when the training data are changed, assuming that the data size remains the same. By contrast, the computational complexity for updating the EDN becomes much smaller than that for building the initial EDN, so long as

the training data do not change completely. Finally, the EDN can be extended to approximate multiple objectives with little increase in computational complexity, while there will be a linear increase in complexity if the GP is used, in case a GP must be built for each objective.

IV. Empirical Results

In this section, we at first examine the effectiveness of the proposed model management strategy by comparing three variants (GP-ARMOEA, GPC-ARMOEA and GPD-ARMOEA) of the proposed EDN-ARMOEA algorithms. Specifically, GP-ARMOEA is a variant of EDN-ARMOEA where EDN is replaced with GPs and the proposed model management strategy is emplyed, similarly, GPC-ARMOMO and GPD-ARMOEA adopt GPs as the surrogates but use different model management strategies. GPC-ARMOEA only uses convergence as the selection criterion, while GPD-ARMOEA only uses diversity as the selection criterion. Note that both the selection criteria are defined in Section III.B. To further test the effectiveness of the proposed strategies, EDN-ARMOEA is compared with GP-ARMOEA, the heterogeneous ensemble (HeE)-assisted AR-MOEA [31]. Then, the performance of the proposed EDN model is tested by comparing with the HeE and GP. Finally, the performance of EDN-ARMOEA is compared with two state-of-the-art SAEAs, and the parameter sensitivity analysis is given. Moreover, in Section VII in the Supplementary material, EDN-ARMOEA is applied to the optimization of crude oil distillation units to verify the performance of the proposed algorithm on real-world problems. DTLZ [58] and WFG [59] test suites are adopted in the simulations. Note that it has been a common practice to test SAEAs on computationally efficient benchmark problems such as DTLZ and WFG, rather than directly on expensive real-world problems. One major difference from testing other EAs, however, is that only a very limited number of FEs using the objective functions is allowed so as to satisfy the assumption that the evaluations of the objectives in the benchmarks are computationally expensive. For each test problem, the largest number of decision variables and objectives are set to 100 and 20, respectively.

A. Parameter Settings

The following parameter settings are used in the empirical studies, where the number of decision variables and the number of objectives are denoted by d and m, respectively:

1) The number of independent runs is 20.
2) In Algorithm 2, 120 real function evaluations are used to test the performance of compared algorithms except the initial $11d-1$ training data, so the maximum number of function evaluations FE_{max} is $11d+119$; the parameter $delta$ is set to 0.08; the number of individuals to be evaluated by the real functions in each generation k is set to 5 (number of clusters in the k-means algorithm); based on our pilot study given in the Supplementary material, the population size P and the maximum number of generations $iter$ for AR-MOEA are 50 and 20, respectively. The sensitivity analysis for $delta$ and k is presented in Section IV-D.
3) In Algorithm 3, both the probabilities p_I and p_R are set to 0.9; as suggested in [42], the weight decay wd and the learning rate lr are set to 1×10^{-5} and 0.01, respectively; the number of neurons in both hidden layers J and K are 40; the size of mini-batch $batchsize$ is equal to the number of the decision variables; the iteration number for building the initial EDN $iter_{train}$ is set to 8×10^4; the number of calculations for approximating the output $iter_{test}$ is 100; and the iteration number for updating the EDN $iter_r$ is 8×10^3. Note again that the depth and size of the EDN are relatively small, mainly because the available number of training data is very limited. As J and K are fixed for all test problems considered in this work and the number of training data linearly increases with the number of decision variables, $iter_{train}$ and $iter_r$ are also fixed. The sensitivity analysis for p_I, p_R, J, K, $iter_{train}$ and $iter_r$ is given in Section IV-D.
4) The GP model is implemented by DACE, a Matlab Kriging toolbox. The zero order polynomial and the Gaussian kernel function are adopted for the regression model and the correlation model, respectively; all initial parameters of the correlation function are set to 5; the lower and upper bounds for the parameters of the correlation function are set to 1e-05 and 100, respectively.

In the WFG test suite, the number of position-related parameters k is set to $m-1$ when m is 3 or 5, while k is $2(m-1)$ when m is 10 or 20. The performance indicator adopted is the inverted generational distance (IGD) [60]. One thousand reference points are used to calculate IGD for $m = 3$. The Wilcoxon rank sum is used to conduct statistical tests at a significance level of 5%, where a symbol $+$ indicates that EDN-ARMOEA outperforms the compared algorithm, while a $-$ means that the compared algorithm outperforms EDN-ARMOEA, and a $\approx$ means that the results obtained by the two algorithms are statistically comparable. All experiments are conducted on a computer with an Intel Core i7, 3.4 GHz CPU, and the Microsoft Windows 7 Enterprize SP1 64-bit operating system. All compared algorithms are implemented in Matlab R2014a based on the PlatEMO toolbox [61].

B. Comparison with GPC-ARMOEA and GPD-ARMOEA

This set of comparative studies aims to verify the model management strategy proposed in III.B by comparing the GP-ARMORA with the GPC-ARMOEA and GPD-ARMOEA. Table I presents the statistical results in terms of IGD obtained by GP-ARMOEA, GPC-ARMOEA and GPD-ARMOEA on 20-D test problems. From these results, we can see that GP-ARMOEA achieves the best performance in terms of convergence and diversity on eight out of the sixteen test instances, while GPC-ARMOEA and GPD-ARMOEA perform best on DTLZ2 and WFG1, respectively. It is a general accepted notion that the key to achieve good performance on MOPs relies on whether the algorithm can achieve a good balance between convergence and diversity. Based on the observations in Table I, the proposed model management strategy helps the algorithm converge towards a set of well distributed solutions in the sense that both the convergence metric and diversity metric are taken into consideration.

TABLE I
MEAN (STANDARD DEVIATION) IGD VALUES OBTAINED BY GP-ARMOEA, GPC-ARMOEA AND GPD-ARMOEA ON 20-DIMENSIONAL 3-OBJECTIVE TEST INSTANCES

	GP-ARMOEA		GPC-ARMOEA		GPD-ARMOEA
DTLZ1	350.95 3.4e+01	≈	344.13 3.6e+01	≈	360.41 3.7e+01
DTLZ2	0.7224 7.9e-02	−	0.6898 9.8e-02	+	0.8278 5.7e-02
DTLZ3	1063.6 1.4e+02	≈	1078.7 1.5e+02	≈	1081.2 1.5e+02
DTLZ4	0.9692 8.0e-02	+	1.0276 1.2e-01	+	1.0590 9.9e-02
DTLZ5	0.5884 7.e-02	≈	0.5897 7.7e-02	+	0.7420 5.2e-02
DTLZ6	13.004 4.5e-01	+	14.043 4.6e-01	+	13.428 7.1e-01
DTLZ7	2.1828 3.8e-01	+	2.9850 6.5e-01	≈	2.2538 9.8e-02
WFG1	2.3241 1.1e-01	≈	2.3479 1.0e-01	−	2.2480 8.9e-02
WFG2	0.6802 3.1e-02	≈	0.6903 3.2e-02	+	0.7812 3.6e-02
WFG3	0.6836 2.2e-02	≈	0.6997 2.4e-02	≈	0.7184 1.3e-02
WFG4	0.5236 1.1e-02	≈	0.5329 3.7e-02	+	0.5488 1.6e-02
WFG5	0.6631 1.9e-02	+	0.7041 2.6e-02	+	0.6946 1.9e-02
WFG6	0.8698 1.4e-02	≈	0.8779 3.9e-02	≈	0.8777 1.3e-02
WFG7	0.6679 1.2e-02	≈	0.6698 2.1e-02	≈	0.6857 1.7e-02
WFG8	0.7271 2.1e-02	+	0.7971 3.1e-02	+	0.7648 2.0e-02
WFG9	0.8895 3.3e-02	≈	0.8795 5.1e-02	≈	0.8977 4.9e-02
+/ − / ≈	5/1/10		8/1/7		

C. Comparison with GP-ARMOEA and HeE-ARMOEA

To demonstrate the effectiveness of the EDN, we compare the performance of AR-MOEA assisted by the EDN, GP and HeE, respectively, on the DTLZ and WFG test suites. The results in terms of IGD and HV values on DTLZ1-7 are presented in Table II and III, respectively, while the results on WFG1-9 are presented in Table SIII and Table SIV in the Supplementary material.

First, DTLZ and WFG test suits with $d = 20, 40, 60, 100$ when $m = 3$ are used to test the impact of the dimensionality of the decision space. Table SIII in the Supplementary material and Table II summarizes the statistical results in terms of IGD values obtained by EDN-ARMOEA, GP-ARMOEA and HeE-ARMOEA for different decision variables. We can see from these results that EDN-ARMOEA achieves the best overall performance, followed by HeE-ARMOEA. EDN-ARMOEA outperforms GP-ARMOEA and HeE-ARMOEA on 60, and perform comparably with GP-ARMOEA and HeE-ARMOEA on 36, respectively, out of a total of 64 test instances. Note that although HeE-ARMOEA also shows promising performance, its performance degrades as the number of objectives increases. Second, the impact of the number of objectives is examined by setting $m = 3, 5, 10, 20$ when $d = 40$. Table SIV in the Supplementary material and Table III presents the

TABLE II
MEAN (STANDARD DEVIATION) OF THE IGD VALUES OBTAINED BY EDN-ARMOEA, GP-ARMOEA AND HEE-ARMOEA FOR DIFFERENT NUMBERS OF DECISION VARIABLES ON THE DTLZ TEST SUITE

Algorithm	d	EDN-ARMOEA	GP-ARMOEA	HeE-ARMOEA
DTLZ1	20	350.0 (3.9e+01)	350.9 (3.4e+01) ≈	345.6 (3.2e+01) ≈
	40	859.9 (5.8e+01)	884.9 (6.2e+01) ≈	862.3 (6.8e+01) ≈
	60	1425 (4.9e+01)	1445 (6.8e+01) ≈	1458 (5.5e+01) +
	100	2596 (7.0e+01)	2614 (6.8e+01) ≈	2605 (6.1e+01) ≈
DTLZ2	20	0.647 (6.6e-02)	0.723 (7.9e-02) +	0.311 (2.4e-02) −
	40	1.972 (1.2e-01)	2.007 (1.2e-01) ≈	0.712 (5.3e-02) −
	60	3.296 (1.3e-01)	3.316 (1.4e-01) ≈	1.437 (1.1e-01) −
	100	6.097 (1.5e-01)	6.104 (1.5e-01) ≈	3.329 (2.6e-01) −
DTLZ3	20	1064 (1.4e+02)	1064 (1.4e+02) ≈	1062 (1.2e+02) ≈
	40	2778 (1.6e+02)	2754 (1.5e+02) ≈	2781 (1.5e+02) ≈
	60	4548 (2.0e+02)	4507 (1.7e+02) ≈	4571 (1.9e+02) ≈
	100	8252 (2.1e+02)	8274 (2.1e+02) ≈	8253 (2.9e+02) ≈
DTLZ4	20	0.844 (1.8e-01)	0.969 (8.0e-02) +	0.974 (3.0e-02) +
	40	2.096 (1.7e-01)	2.305 (1.9e-01) +	1.242 (5.4e-02) −
	60	3.450 (2.0e-01)	3.666 (1.6e-01) +	1.725 (1.2e-01) −
	100	6.322 (2.2e-01)	6.332 (2.3e-01) ≈	3.250 (1.7e-01) −
DTLZ5	20	0.464 (7.8e-02)	0.588 (7.5e-02) +	0.157 (1.6e-02) −
	40	1.906 (1.2e-01)	1.949 (1.4e-01) ≈	0.558 (5.0e-02) −
	60	3.220 (1.6e-01)	3.235 (1.7e-01) ≈	1.306 (8.7e-02) −
	100	6.015 (1.8e-01)	6.029 (1.6e-01) ≈	3.280 (2.2e-01) −
DTLZ6	20	13.71 (7.9e-01)	13.00 (4.5e-01) −	14.18 (2.7e-01) +
	40	31.64 (4.3e-01)	32.21 (6.0e-01) +	32.13 (4.6e-01) +
	60	49.53 (7.1e-01)	49.77 (4.7e-01) ≈	49.73 (5.3e-01) ≈
	100	49.53 (7.1e-01)	85.35 (6.6e-01) ≈	85.53 (7.4e-01) +
DTLZ7	20	2.959 (7.0e-01)	2.183 (3.8e-01) −	4.440 (5.7e-01) +
	40	4.574 (9.5e-01)	8.581 (1.2e+00) +	6.393 (7.1e-01) +
	60	5.215 (5.8e-01)	9.322 (5.7e-01) +	6.837 (8.0e-01) +
	100	6.559 (5.0e-01)	9.924 (2.7e-01) +	7.505 (3.5e-01) +
+/ − / ≈			9/2/17	9/11/8

TABLE III
MEAN (STANDARD DEVIATION) OF THE IGD VALUES OBTAINED BY EDN-ARMOEA, GP-ARMOEA AND HEE-ARMOEA FOR DIFFERENT NUMBERS OF OBJECTIVES ON THE DTLZ TEST SUITE

Algorithm	m	EDN-ARMOEA	GP-ARMOEA	HeE-ARMOEA
DTLZ1	3	859.9 (5.8e+01)	884.9 (6.2e+01) ≈	862.3 (6.8e+01) ≈
	5	677.6 (4.0e+01)	687.1 (5.2e+01) +	681.1 (6.9e+01) ≈
	10	490.6 (3.2e+01)	532.4 (5.0e+01) +	490.7 (4.9e+01) ≈
	20	11.90 (5.5e-15)	11.90 (5.5e-15) ≈	11.90 (5.5e-15) ≈
DTLZ2	3	1.972 (1.2e-01)	2.007 (1.2e-01) ≈	0.712 (5.3e-02) −
	5	2.641 (7.1e-02)	2.676 (6.1e-02) ≈	3.150 (2.4e-01) +
	10	2.725 (1.3e-02)	2.877 (5.1e-02) +	3.340 (8.5e-02) +
	20	11.68 (2.0e-01)	11.87 (4.3e-02) ≈	11.85 (6.4e-02) ≈
DTLZ3	3	2778 (1.6e+02)	2754 (1.5e+02) ≈	2781 (1.5e+02) ≈
	5	2531 (1.4e+02)	2130 (1.0e+02)≈	2588 (1.6e+02) ≈
	10	2308 (1.5e+02)	2154 (1.e+02) ≈	2130 (1.0e+02)≈
	20	9.355(9.0e-01)	9.818 (6.4e-01) +	10.27 (9.7e-01) +
DTLZ4	3	2.096 (1.7e-01)	2.304 (1.9e-01) +	1.242 (5.4e-02) −
	5	3.529 (9.1e-02)	3.652 (1.8e-01) +	3.821 (2.3e-01) +
	10	8.889 (1.1e-01)	9.202 (1.0e-01) +	9.242 (1.0e-01) +
	20	9.611 (1.0e-01)	9.699 (8.3e-02) ≈	9.768 (8.6e-02) +
DTLZ5	3	1.906 (1.2e-01)	1.949 (1.4e-01) ≈	0.558 (5.0e-02) −
	5	2.740 (7.0e-02)	2.788 (7.9e-02) ≈	3.348 (1.9e-01) +
	10	8.786 (6.0e-02)	9.012 (8.1e-02) +	9.248 (1.2e-01) +
	20	11.90 (5.3e-05)	11.90 (6.4e-05) ≈	11.90 (6.1e-05) ≈
DTLZ6	3	31.64 (4.3e-01)	32.21 (6.0e-01) +	32.13 (4.6e-01) +
	5	25.92 (2.1e-01)	26.12 (2.5e-01) +	26.26 (1.5e-01) +
	10	12.28 (7.1e-01)	20.68(9.1e-01) +	27.33 (7.9e-01) +
	20	11.89 (9.8e-04)	11.90 (5.2e-04) ≈	11.90 (9.8e-04) ≈
DTLZ7	3	4.574 (9.5e-01)	8.581 (1.2e+00) +	6.393 (7.1e-01) +
	5	11.70 (6.0e-01)	19.382 (1.8e+00) +	14.92 (1.3e+00) +
	10	23.59 (1.0e+00)	40.58 (4.1e+00) +	31.89 (2.9e+00) +
	20	10.53 (4.1e-03)	10.68 (4.9e-03) +	10.59 (1.0e-03) +
+/ − / ≈			15/0/13	15/3/10

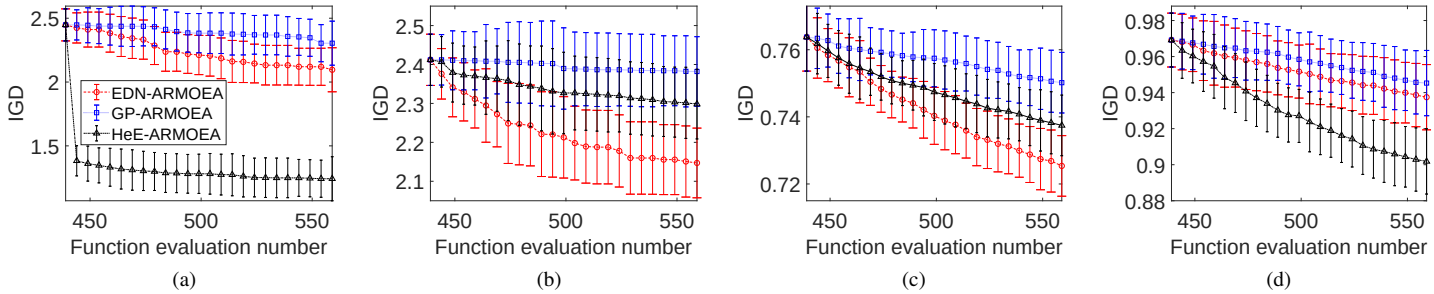


Fig. 2. The error bars of the IGD values versus the number of real function evaluations ($d = 40$, $m = 3$). (a) DTLZ4; (b) WFG1; (c) WFG5; (d) WFG9.

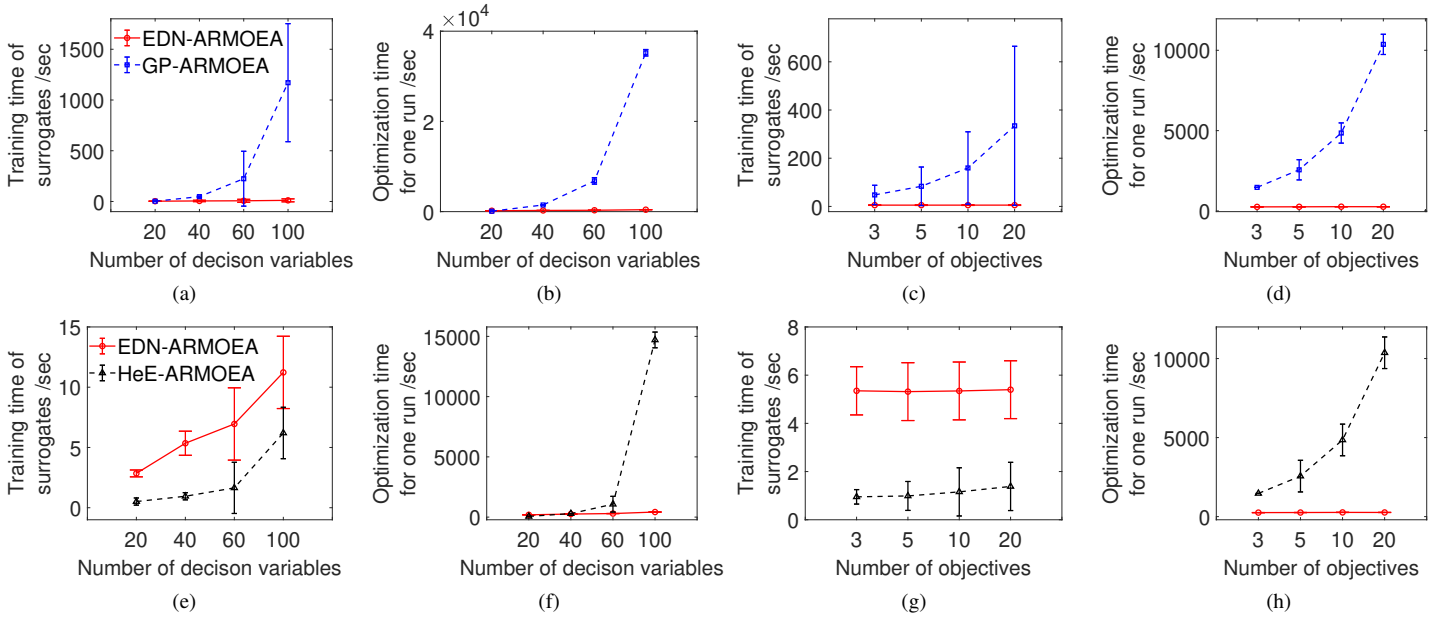


Fig. 3. The error bars of the training and optimization time for EDN-ARMOEA, GP-ARMOEA and HeE-ARMOEA on DTLZ7. (a) Training time of the surrogates versus the number of decision variables d in EDN-ARMOEA and GP-ARMOEA. (b) Optimization time for one run versus d in EDN-ARMOEA and GP-ARMOEA. (c) Training time of the surrogates versus the number of objectives m in EDN-ARMOEA and GP-ARMOEA. (d) Optimization time for one run versus m in EDN-ARMOEA and GP-ARMOEA. (e) Training time of the surrogates versus d in EDN-ARMOEA and HeE-ARMOEA. (f) Optimization time for one run versus d in EDN-ARMOEA and HeE-ARMOEA. (g) Training time of the surrogates versus m in EDN-ARMOEA and HeE-ARMOEA. (h) Optimization time for one run versus m in EDN-ARMOEA and HeE-ARMOEA.

statistical results in terms of IGD values obtained by the three compared algorithms. It is clear that the best overall performance on all the test instances is achieved by EDN-ARMOEA, confirming the scalability of EDN-ARMOEA. In particular, EDN-ARMOEA shows significantly better performance than GP-ARMOEA on a majority of the test instances (41 out of 64), which further demonstrates the advantage of the proposed EDN over GP for addressing high-dimensional MaOPs. This might be attributed to two reasons. First, the proposed model management strategy is capable of achieving a better balance between the convergence and diversity. Second, EDN-ARMOEA benefits from the proposed EDN model in that the EDN model can provide more accurate fitness predictions and useful uncertainty information for high-dimensional problems. To sum up, we conclude that the proposed EDN-ARMOEA is well suited for high-dimensional problems, especially when the number of objectives is large.

Taking the test instance DTLZ7 as an example, Fig. 3 plots the error bars of the computation time used for training and the optimization as the number of decision variables and the number of objectives change. It can be observed that the proposed EDN is much more scalable than GP and HeE to the increase in the number of training samples and the number of objectives, which confirms our theoretical analysis of the computational complexity of the EDN. Since feature selection in HeE is performed off-line only once based on the initial training data, the time consumption for feature selection is not included in the training time of HeE-ARMOEA. That is why the training time for EDN-ARMOEA is more than that for HeE-ARMOEA, while the optimization time of EDN-ARMOEA is less than that of HeE-ARMOEA. Fig. 2 plots the change of the average IGD values over the number of real FEs when d is 40 and m is 3. The IGD values of EDN-ARMOEA decrease quickly on WFG1 and WFG5 as the evolution proceeds, while the IGD values of HeE-ARMOEA decrease quickly on the other two test problems.

D. Comparisons in Fitness and Uncertainty Estimation

To further understand the advantages of the proposed EDN over HeE and GP models in terms of the fitness approximation

and uncertainty estimation, we examine the mean absolute error (MAE) of the fitness estimated by the models under comparison. Meanwhile, we calculate the mean standard deviation (STD) of the estimated variance ($\hat{s}^2$) to assess the ability of the compared models to provide uncertainty information. More specifically, in each run, EDN, HeE and GP models are trained with the same training data at first. Subsequently, 50 random samples are generated as the test data. Finally, each model estimates the fitness and the uncertainty on each test data. Besides, each model also estimates the fitness on each training data. Twenty independent runs are performed to obtain the mean MAE and mean STD of the estimated fitness and the uncertainty, respectively. Here, our hypothesis is that the smaller the mean MAE, the better the accuracy of the model, whereas the larger the mean STD, the better ability of estimating the uncertainty. The results are shown in Tables SV-SVII in the Supplementary material. Regarding to the mean MAE of the estimated fitness, on the one hand, the smallest absolute error of the predictions for the test data is achieved by the proposed EDN, indicating that the estimated objective values obtained by EDN is the closest to the real objective values. On the other hand, the comparison between the EDN and HeE in Table SVI demonstrates that the EDN model also can give better predictions on the training data. It is evident according to Table SVII that the mean STD of $\hat{s}$ obtained by GP is the smallest, which agrees with the observations obtained in [31], [62], indicating that GP fails to properly predict the uncertainty for high-dimensional problems when the number of training data is small. By the contrast, the proposed EDN is still able to provide the uncertainty information for high-dimensional problems.

To take a further look at how the approximated objective values and estimated uncertainties have contributed to the optimization, we perform comparative studies to examine how efficiently EDN-ARMORA, HeE-ARMOEA and GP-ARMOEA can identify solutions that can contribute to performance improvement. Fig. S1 in the Supplementary material plots the number of solutions that have contributed to the improvement of IGD over the number of real FEs during the evolution on four test problems. From these results, we can find that both EDN-ARMOEA and HeE-ARMOEA can find more contributing solutions than GP-ARMORA. EDN-ARMOEA finds the most contributing solutions on DTLZ4 and WFG9, while HeE-ARMOEA is most efficient in identifying contributing solutions on WFG1 and WFG5. These results are consistent with the observed differences in the performance of the three compared algorithms.

E. Comparison with Some State-of-the-art SAEAs

In this subsection, the performance of the proposed EDN-ARMOEA on high-dimensional MOPs/MaOPs is compared with two state-of-the-art SAEAs, namely GP-assisted RVEA (K-RVEA) [54] and GP-assisted MOEA/D (MOEA/D-EGO) [33]. More precisely, two sets of experiments are conducted on the DTLZ test suite, investigating the impact of the dimensionality of the search space and the number of objectives, respectively, on the performance of the three algorithms. The corresponding statistical results in terms of IGD values of the solution sets obtained by the three algorithms are presented in Tables SVIII-SIX in the Supplementary material, respectively.

It is evident according to Table SVIII that the proposed algorithm shows better performance in terms of convergence and diversity than K-RVEA and MOEA/D-EGO, revealing its superiority in handling high-dimensional MOPs. EDN-ARMOEA significantly outperforms K-RVEA and MOEA/D-EGO on nine and eleven, respectively, out of 21 problems in total, while K-RVEA and MOEA/D-EGO achieve the best results on two and three instances with $d = 20$, respectively. It confirms that the EDN with the proposed model management strategy can benefit the EDN-ARMOEA for solving high-dimensional problems. Similar conclusion regarding the number of objectives can be drawn from Table SIX.

F. Parameter Sensitivity Analysis

The influence of the number of the initial samples has been discussed in [31], [51] and the results demonstrate that the differences between the performances of different initial samples are small, so the sensitivity to the initial samples is not included here. Similarly, the size of the population and the maximum number of generation are set based on our pilot study. The sensitivity to the five important parameters in EDN-ARMOEA are analyzed in the following and all results are averaged over 20 independent runs.

1) Sensitivity to the Number of Training Iterations ($iter_{train}, iter_r$): $iter_{train}$ and $iter_r$ are the number of learning iterations in constructing and updating the EDN, respectively. The number of iterations is a key parameter determining the quality and efficiency of the proposed EDN. If $iter_{train}$ and $iter_r$ are too small, the EDN is not able to approximate the fitness values adequately well. If $iter_{train}$ and $iter_r$ are too large, training the EDN as well as approximating the fitness using EDN will be very time-consuming.

We examine the performance of EDN-ARMOEA on four WFG instances for different $iter_{train}$ and $iter_r$ values. The results are plotted in Figs. S2-S3 in the Supplementary material. EDN-ARMOEA performs well on WFG1 as $iter_{train}$ increases, while the IGD values on WFG2, WFG3 and WFG4 are not very sensitive to $iter_{train}$. EDN-ARMOEA performs well on WFG1 and WFG4 as $iter_r$ increases, whereas the IGD values on WFG2 and WFG3 are not very sensitive to $iter_r$. Since the EDN has 353 parameters in total, we recommend that $iter_{train}$ and $iter_r$ should be about 30 times and 3 times of the number of the parameters in EDN, respectively.

2) Sensitivity to the Number of Hidden Neurons (J, K): We assume that J is equal to K for simplicity. More hidden neurons mean that more data and more time are needed in the training. The performance of EDN-ARMOEA is examined under four different settings based on the sensitivity analysis on $iter_{train}$ and $iter_r$. It can be seen from Fig. S4 in the Supplementary material that the IGD on WFG4 hardly changes when the number of hidden neurons is more than 20, while the IGD values on the other three problems decrease as J and K increase except on WFG1 when J and K are 50 and on WFG2 when J and K are 40. Thus, we recommend that J and K are 40 in consideration of the computational time.

3) Sensitivity to the Dropout Probability ($1-p_I, 1-p_R$): Srivastava et al. recommended that the dropout probability is 0.2 for the input layer and that the dropout probability is 0.5 for hidden layers [41], while other dropout probability has also been suggested [63]. There is no theoretic guidance for setting the dropout probability, meaning that this parameter needs to be properly tuned for different problems. The error bars of the mean IGD of EDN-ARMOEA over $1-p_I$ and $1-p_R$ is plotted in Fig. S5 in the Supplementary material, indicating that a dropout probability between 0.1 and 0.2 for both input and hidden layers should work well for most problems. We set both dropout probabilities to 0.1 in this work.

4) Sensitivity to the Number of Solutions Evaluated by the Real Fitness Function in Each Generation (k): The results of sensitivity analysis for k are plotted in Fig. S6 in the Supplementary material. The best IGD values are achieved on WFG1 and WFG2 when k is set to 3, while the best IGD values are achieved on WFG3 and WFG4 when k equals to 5. The performance change is relatively small except on WFG1. In this work, we set k to 5 as a compromise.

5) Sensitivity to the Parameter $delta$: Parameter $delta$ is used to enhance the diversity of the population when the proportion of the valid reference points in the current generation is smaller than that in the previous generation. A larger $delta$ indicates that fewer valid reference points are needed in the current generation, so an increase in $delta$ decreases the frequency of selecting the uncertain solutions for the real FE. Fig. S7 in the Supplementary material shows the influence of $delta$ on the mean frequency of enhancing diversity. The frequency decreases quickly as $delta$ increases, and the effect of $delta$ on the frequency is different for different problems. The frequency of enhancing diversity determines the trade-off between the convergence and diversity of the achieved optimal solution set, and in this work, $delta$ is set to 0.08.

V. Conclusion

GPs are not well suited for high-dimensional, multi- and many-objective problems because of their prohibitive computational complexity when the number of training data becomes large. This work proposes an efficient dropout neural network (EDN) to replace GPs for surrogate-assisted evolutionary optimization of computationally expensive problems with a large number of decision variables or many objectives. The main motivation comes from the fact that neural networks are more scalable to the number of training samples and the number of outputs than GPs in computational complexity and that dropout neural networks can also reason about the uncertainty of their outputs. The proposed EDN is more computationally efficient than the conventional dropout neural networks so that it can be used as a surrogate in SAEAs.

EDN-ARMOEA, which employs EDN to assist AR-MOEA, a recently proposed robust MOEA for solving multi- and many-objective optimization problems, is verified on test problems with up to 100 decision variables and up to 20 objectives in comparison with state-of-the-art SAEAs. The results show that EDN-ARMOEA not only outperforms the compared algorithms on the majority of the test instances studied in this work, but also is computationally much more efficient. The difference in computational efficiency becomes more significant as the number of decision variables and the number of objective increase. Finally, EDN-ARMOEA is compared with AEMOEA, GP-assisted AR-MOEA, and heterogeneous ensemble assisted AR-MOEA on operational optimization in crude oil distillation units. The results confirm the advantage of the proposed EDN-ARMOEA over the three compared algorithms, demonstrating its capability in solving real-world optimization problems.

Although the proposed EDN outperforms the GP in both computational efficiency and optimization performance on most high-dimensional or MOPs/MaOPs, the computational complexity of the EDN can be further reduced and the performance of the EDN-assisted optimization can be further enhanced. Compared with the ensemble-based approach, the EDN has better computational scalability, but slightly worse performance in optimization of high-dimensional problems. Therefore, our future work will be dedicated to the development of more efficient dropout technique to enhance its accuracy in fitness approximation and uncertainty estimation without increasing its computational complexity. More efficient model management techniques that do not rely on sensitive user-defined parameters will also be investigated.

References

[1] Y. Tian, C. He, R. Cheng, and X. Zhang, "A multistage evolutionary algorithm for better diversity preservation in multiobjective optimization," *Trans. Syst. Man Cybern. Syst.*, 2019.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[3] Y. Jin and B. Sendhoff, "Fitness approximation in evolutionary computation-a survey," in *Proc. Genetic Evol. Comput. Conf*, 2002, pp. 1105–1112.

[4] ——, "A systems approach to evolutionary multiobjective structural optimization and beyond," *IEEE Comput. Intell. Mag.*, vol. 4, no. 3, pp. 62–76, Aug. 2009.

[5] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA J.*, vol. 41, no. 4, pp. 687–696, 2003.

[6] D. E. Ighravwe, S. A. Oke, and K. A. Adebiyi, "A surrogate model for optimal maintenance workforce cost determination in a process industry," *Eng. Appl. Sci. Res.*, vol. 44, no. 4, pp. 202–207, 2017.

[7] L. M. Ochoa-Estopiera, V. M. Enríquez-Gutierreza, J. M. Lu Chena, L. H.-S. Fernández-Ortizb, and M. Jobsonc, "Industrial application of surrogate models to optimize crude oil distillation units," *Chem. Eng. Trans.*, vol. 69, 2018.

[8] D. Guo, T. Chai, J. Ding, and Y. Jin, "Small data driven evolutionary multi-objective optimization of fused magnesium furnaces," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Athens, Greece, December 2016.

[9] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 442–458, 2019.

[10] D. Horn, T. Wagner, D. Biermann, C. Weihs, and B. Bischl, "Model-based multi-objective optimization: taxonomy, multi-point proposal, toolbox and benchmark," in *Proc. Int. Conf. Evol. Multi Criterion Optim.*, Guimarães, Portugal, 2015, pp. 64–78.

[11] A. Díaz-Manríquez, G. Toscano, J. H. Barron-Zambrano, and E. Tello-Leal, "A review of surrogate assisted multiobjective evolutionary algorithms," *Comput. Intell. Neurosci.*, vol. 2016, 2016.

[12] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 61–70, 2011.

[13] R. Allmendinger, M. T. M. Emmerich, J. Hakanen, Y. Jin, and E. Rigoni, "Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case," *J. Multi-Criteria Decis. Anal.*, vol. 14, no. 1/2, pp. 5–25, 2017.

[14] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1–35, 2015.

[15] H. Chen, R. Cheng, W. Pedrycz, and Y. Jin, "Solving many-objective optimization problems via multistage evolutionary search," *Trans. Syst. Man Cybern. Syst.*, 2019, DOI: 10.1109/TSMC.2019.2930737.

[16] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 263–282, 2002.

[17] X. Zou, Y. Chen, M. Liu, and L. Kang, "A new evolutionary algorithm for solving many-objective optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1402–1412, 2008.

[18] F. Di Pierro, S.-T. Khu, and D. A. Savic, "An investigation on preference order ranking scheme for multiobjective evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 11, no. 1, pp. 17–45, 2007.

[19] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.

[20] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, 2013.

[21] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.

[22] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Int. Conf. Parallel Probl. Solving Nat.* Springer, 2004, pp. 832–842.

[23] A. Díaz-Manríquez, G. Toscano-Pulido, C. A. C. Coello, and R. Landa-Becerra, "A ranking method based on the R2 indicator for many-objective optimization," in *2013 Proc. IEEE Congr. Evol. Comput.* IEEE, 2013, pp. 1523–1530.

[24] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 97–112, 2018.

[25] Y. T. X. Z. R. C. Y. J. Cheng He, Lianghao Li and X. Yao, "Accelerating large-scale multi-objective optimization via problem reformulation," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 949–961.

[26] L. M. Antonio and C. A. C. Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *2013 Proc. IEEE Congr. Evol. Comput.* IEEE, 2013, pp. 2758–2765.

[27] A. Song, Q. Yang, W.-N. Chen, and J. Zhang, "A random-based dynamic grouping strategy for large scale multi-objective optimization," in *2016 Proc. IEEE Congr. Evol. Comput.* IEEE, 2016, pp. 468–475.

[28] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 129–142, 2016.

[29] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 74–88, 2018.

[30] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 644–660, Aug. 2017.

[31] D. Guo, Y. Jin, J. Ding, and T. Chai, "Heterogeneous ensemble-based infill criterion for evolutionary multiobjective optimization of expensive problems," *IEEE Trans. Cybern*, vol. 49, no. 3, pp. 1012–1025, 2018.

[32] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, 2006.

[33] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 456–474, 2009.

[34] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *Trans Neural Netw. Learn. Syst.*, 2020.

[35] D. Büche, N. N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with Gaussian process fitness function models," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 183–194, May 2005.

[36] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[37] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 456–474, Jun. 2010.

[38] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 329–355, Jun. 2010.

[39] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable Bayesian optimization using deep neural networks," in *ICML*, 2015, pp. 2171–2180.

[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Suskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv:1207.0580, 2012.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[42] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Appendix," [Online]. Available: https://arxiv.org/abs/1506.02157.

[43] ——, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016, pp. 1050–1059.

[44] H. Wang, Y. Jin, and J. Doherty, "Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2664–2677, Sep. 2017.

[45] N. Azzouz, S. Bechikh, and L. Ben Said, "Steady state IBEA assisted by MLP neural networks for expensive multi-objective optimization problems," in *Genet. Evol. Comput. Conf.* ACM, 2014, pp. 581–588.

[46] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 609–622, 2018.

[47] Y. Tian, X. Zhang, R. Cheng, and Y. Jin, "A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric," in *2016 Proc. IEEE Congr. Evol. Comput.* IEEE, 2016, pp. 5222–5229.

[48] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, 2015.

[49] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," in *Advances in Neural Information Processing Systems*, 2013, pp. 351–359.

[50] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[51] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Glob. Optim.*, vol. 13, no. 4, pp. 455–492, 1998.

[52] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, 2006.

[53] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[54] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 129–142.

[55] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocess Microsyst.*, vol. 73, p. 102989, 2020.

[56] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[58] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, Honolulu, HI, USA, 2002, pp. 825–830.

[59] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, 2006.

[60] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.

[61] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 73–87, 2017.

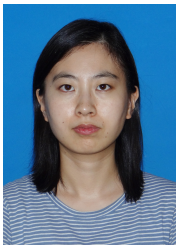
[62] J. Tian, Y. Tan, J. Zeng, C. Sun, and Y. Jin, "Multi-objective infill criterion driven Gaussian process assisted particle swarm optimization

of high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 459–472, 2019.

[63] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino, "Curriculum dropout," in *Proceedings of the IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3544–3552.

Dan Guo received the B.Sc. and Ph.D. degrees from Northeastern University, Shenyang, China, in 2012 and 2019, respectively. She currently works at the Science and Technology on Electromagnetic Scattering Laboratory, Beijing. Her current research interests include surrogate-assisted multiobjective evolutionary optimization and machine learning.

Xilu Wang received B.Sc. and M.Sc. from Harbin Institute of Technology in 2016 and Xidian University in 2018, respectively. She is currently a PhD student at the University of Surrey under the supervision of Prof. Yaochu Jin.



Kailai Gao received the B.Sc. degree from Northeast Petroleum University, Daqing, China, in 2016, and the M.Sc. degree from Northeastern University, Shenyang, China, in 2019, where she is currently pursuing the Ph.D. degree in control theory and control engineering with the State Key Laboratory of Synthetical Automation for Process Industry. Her current research interests include multitasking evolutionary optimization, and their application.

Yaochu Jin (M'98-SM'02-F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Germany, in 2001. He is currently a Distinguished Chair Professor in Computational Intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He was a Finland Distinguished Professor with the University of Jyväskylä, Finland, and Changjiang Distinguished Visiting Professor with the Northeastern University, China. His main research interests include evolutionary computation, multiobjective machine learning, secure machine learning, and self-organizing collective systems.

Dr. Jin was a recipient of the 2014, 2016, and 2019 IEEE Computational Intelligence Magazine Outstanding Paper Award, the 2018 and 2020 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, and the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He was named 2019-2020 Highly Cited Researchers by the Web of Science Group. Dr Jin is presently the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and Complex & Intelligent Systems. He was an IEEE Distinguished Lecturer for the period from 2013 to 2015 and 2017 to 2019. He is a Fellow of IEEE.

Jinliang Ding (M'09-SM'14) received the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2012. He is a Professor with the State Key Laboratory of Synthetical Automation for Process Industry, Northeastern University. He has authored or co-authored over 100 refereed journal papers and refereed papers at international conferences. He is also the inventor or co-inventor of 17 patents. His current research interests include modeling, plant-wide control and optimization for the complex industrial systems, stochastic distribution control, and multiobjective evolutionary algorithms and its application.

Dr. Ding was a recipient of the Young Scholars Science and Technology Award of China in 2016, the National Science Fund for Distinguished Young Scholars in 2015, the National Technological Invention Award in 2013, two First-Prize of Science and Technology Award of the Ministry of Education in 2006 and 2012, respectively, and the IFAC Control Engineering Practice 2011-2013 Paper Prize.

Tianyou Chai (M'90-SM'97-F'08) received the Ph.D. degree in control theory and engineering from Northeastern University, Shenyang, China, in 1985.

He has been with the Research Center of Automation, Northeastern University, Shenyang, China, since 1985, where he became a Professor in 1988 and a Chair Professor in 2004. He is the founder and Director of the Center of Automation, which became a National Engineering and Technology Research Center in 1997. He has made a number of important contributions in control technologies and applications. He has authored and coauthored two monographs, 84 peer reviewed international journal papers and around 219 international conference papers. He has been invited to deliver more than 20 plenary speeches in international conferences of IFAC and IEEE. His current research interests include adaptive control, intelligent decoupling control, integrated plant control and systems, and the development of control technologies with applications to various industrial processes.

Prof. Chai is a member of the Chinese Academy of Engineering, an academician of International Eurasian Academy of Sciences, and IFAC Fellow. He is a distinguished visiting fellow of The Royal Academy of Engineering (UK) and an Invitation Fellow of Japan Society for the Promotion of Science (JSPS). For his contributions, he has won three prestigious awards of National Science and Technology Progress, the 2002 Technological Science Progress Award from the Ho Leung Ho Lee Foundation, the 2007 Industry Award for Excellence in Transitional Control Research from the IEEE Control Systems Society, and the 2010 Yang Jia-Chi Science and Technology Award from the Chinese Association of Automation.