Expensive Multi-Objective Evolutionary Optimization Assisted by Dominance Prediction

Yuan Yuan and Wolfgang Banzhaf

***Abstract*—We propose a new surrogate-assisted evolutionary algorithm for expensive multi-objective optimization. Two classification-based surrogate models are used, which can predict the Pareto dominance relation and θ-dominance relation between two solutions, respectively. To make such surrogates as accurate as possible, we formulate dominance prediction as an imbalanced classification problem and address this problem using deep learning techniques. Furthermore, to integrate the surrogates based on dominance prediction with multi-objective evolutionary optimization, we develop a two-stage preselection strategy. This strategy aims to select a promising solution to be evaluated among those produced by genetic operations, taking proper account of the balance between convergence and diversity. We conduct an empirical study on a number of well-known multi- and many-objective benchmark problems, over a relatively small number of function evaluations. Our experimental results demonstrate the superiority of the proposed algorithm compared with several representative surrogate-assisted algorithms.**

***Index Terms*—Surrogate-assisted evolutionary computation, expensive multi-objective optimization, many-objective optimization, deep neural networks, metamodeling.**

I. Introduction

MULTI-OBJECTIVE optimization problems (MOPs) arise naturally in most scientific and engineering disciplines. Evolutionary algorithms (EAs) are well suited for solving MOPs due to the population-based nature that allows the generation of the Pareto front (PF) approximation in a single run. Additionally, multi-objective EAs (MOEAs) can tackle MOPs with complex features (e.g., non-convex, mixed-integer and black-box) where traditional multi-objective optimization techniques are not easily applicable. MOEAs have been extensively studied [1], and recent research efforts [2], [3] mainly focus on making MOEAs scalable to MOPs with more than three objectives, often called many-objective optimization problems (MaOPs). In many practical MOPs, the evaluation of objective functions involves a computationally *expensive* simulation or experimental procedure [4] that can take up to hours or even days. This poses a serious obstacle to the use of multi-objective optimizers, particularly MOEAs, which usually require thousands or even tens of thousands of function evaluations to obtain satisfactory results. A common approach for addressing this issue is to develop cheap approximations to expensive objective functions. With the guidance of these approximations, known as *surrogates* or *metamodels* [4], [5], it is possible for an optimizer to achieve good performance with just very few number of function evaluations.

One popular class of surrogate-assisted algorithms for expensive MOPs is based on ideas about single-objective efficient global optimization (EGO) [6]. This optimization paradigm is also known as Bayesian optimization [7]. EGO uses a Kriging (also called Gaussian process regression) model to approximate the true objective function, and selects the next sample point for evaluation by optimizing an infill criterion called expected improvement (EI). Because EI is generally a non-convex function, its optimization usually resorts to an evolutionary algorithm. To extend EGO to MOPs, the key issue is how to convert multiple objective functions into a scalar function. A typical way to do this is to use a set of predefined scalarizing weight vectors (e.g., ParEGO [8] and MOEA/D-EGO [9]). However, a major problem with this approach is that the use of uniformly distributed weight vectors does not necessarily result in a good distribution of non-dominated solutions in the objective space [8], [10], so that the related algorithms may suffer from a loss of diversity. Another way to extend EGO for MOPs is to use a set-based performance indicator called hypervolume (HV) [11] (e.g., SMS-EGO [12] and SUR [13]). However, the computation of HV is costly and its time complexity grows exponentially with the number of objectives, limiting the applicability of such algorithms in MaOPs. For an overview and a taxonomy of EGO-based multi-objective optimizers, the reader is referred to [14], [15]. A recent study [16] discussed several alternative scalarization strategies for MOPs within the EGO framework.

Another prominent class of surrogate-assisted algorithms for expensive MOPs is based on existing MOEAs. These algorithms mostly build one surrogate model for each objective. A variety of machine learning models, such as Kriging, neural networks and support vector machine (SVM), have been used to build such surrogate models in MOEAs. There are mainly two strategies in terms of how to then use those surrogates: *fitness replacement* and *preselection*. In fitness replacement, a conventional MOEA is run as usual, but most solutions use the *predicted* objective values by the surrogate models, and only occasionally several solutions selected from the current population use true objective values provided by function evaluation. As a result, survival selection may be based on both true and predicted objective values. Typical surrogate-assisted MOEAs using fitness replacement can be found in [17]–[21]. On the other hand, in the alternative preselection

Manuscript received xxx; revised yyy.

Y. Yuan and W. Banzhaf are with the Department of Computer Science and Engineering and the BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824 USA (e-mail: yyuan@msu.edu; banzhafw@msu.edu).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.XXX.XXX

method, surrogate models serve as filters which select only a few of offspring solutions for function evaluation in each generation and discard the rest. Evaluated offsprings will then undergo survival selection in a MOEA. So unlike in fitness replacement, survival selection in preselection is always based on true objective values. Several notable surrogate-assisted MOEAs using preselection can be found in [22]–[24]. Note that no matter which of these methods is used, the newly evaluated solutions are usually employed to update the surrogates. One current important trend in the design of surrogate-assisted MOEAs is to leverage recent advances in evolutionary many-objective optimization so as to better balance convergence and diversity (e.g., K-RVEA [21] and HSMEA [24]). General reviews of existing surrogate-assisted MOEAs can be found in various sources [25]–[28].

Although building one surrogate model for each objective is a straightforward way to construct surrogate-assisted MOEAs, the cumulative approximation errors from each metamodel can become detrimental to the overall accuracy of the optimization process [29], [30]. In addition, building surrogates in this way will incur higher computational costs as the number of objectives increases. To overcome such limitations, another research direction on surrogate-assisted MOEAs that has been less explored is to form a single surrogate model combining all objectives. Loshchilov *et al.* [31] proposed a mono-surrogate combining ideas from SVM regression and one-class SVM, where all current non-dominated solutions are mapped onto a single value (up to some tolerance) and all dominated solutions are rendered to be on one side of this value. The same authors [29] later refined the mapping mechanism using rank-based SVM. Seah *et al.* [32] exploited the non-domination levels obtained by non-dominated sorting, thereby building an ordinal regression based surrogate that can predict the level of a new solution. Yu *et al.* [33] also proposed an ordinal regression based surrogate, where the ordinal level of a solution is assigned based on how far it is from the archived PF. Zhang *et al.* [34] introduced a binary classification based surrogate where the evaluated solutions form two classes, non-dominated and dominated solutions. Pan *et al.* [35] took a different approach to building a classification based surrogate: the evaluated solutions are divided into two classes using a set of reference solutions. Note that since mono-surrogate MOEAs do not directly model the objective functions, they are typically implemented with a preselection strategy. One potential downside of these algorithms is that the function to be modeled could be more complex due to the combination of all objectives, calling for more powerful computational models.

Because dominance comparisons constitute the core procedure in a lot of MOEAs, a machine learning model that can predict the dominance relation between two solutions would mesh well with the classification based mono-surrogate approach. However, surprisingly, *dominance prediction* has received little attention in the literature. The first attempt at Pareto dominance prediction by Guo *et al.* [36] used a Gaussian naive Bayes classifier. But this work was just exploratory and the adopted simple model required strong assumptions that are probably not adequate for learning nonlinear dominance relations. Bandaru *et al.* [37] conducted a more comprehensive study that investigated ten different classification algorithms for Pareto dominance prediction. Their work is highly relevant to our paper here since it involves a feedforward neural network (FNN) model. However, they adopted a still shallow model of a FNN with just one hidden layer, trained using traditional techniques, likely restricting the ability to model complex nonlinear relations. Given the remarkable progress in deep learning [38] over recent years, it is desirable to use deep models for this prediction purpose. In single-objective Bayesian optimization, deep FNNs have already been used as an alternative to Gaussian processes to model distributions over functions [39].

One very important limitation of [37] (just as the preliminary work by Guo *et al.* [36]) is that it does not study how to effectively integrate surrogate models based on dominance prediction with a MOEA for expensive multi-objective optimization. It might even be the case that Pareto dominance prediction alone is not enough to constitute a good surrogate-assisted MOEA. On the one hand, Pareto dominance comparisons can facilitate the convergence to the PF, but they cannot regulate the diversity of solutions. On the other hand, if the number of objectives is high, Pareto dominance may not provide enough selection pressure toward the PF [2], [40], leading to poor convergence. In light of the above limitations in current research on dominance prediction, our paper makes the following contributions:

1) We formulate dominance prediction as an imbalanced classification problem and address this problem in the context of modern deep learning methods, which is expected to better capture complex nonlinear dominance relations. An additional benefit of using deep neural networks as surrogates is that the model can be updated online with mini-batch gradient descent. This fits well with the expensive optimization scenario where the evaluated solutions become available sequentially.
2) We consider for the first time dominance prediction in terms of an additional dominance relation called θ-dominance [41], [42]. θ-dominance explicitly preserves diversity and is able to impose moderate selection pressure even in high-dimensional objective spaces, compensating for the limitations of Pareto dominance.
3) We propose a new surrogate-assisted MOEA called θ-DEA-DP for expensive MOPs, which combines θ-DEA [42] with the dominance prediction based surrogates. θ-DEA-DP is characterized by a two-stage preselection strategy assisted by Pareto dominance and θ-dominance prediction, in order to carefully maintain the balance between convergence and diversity in the objective space.

The rest of this paper is organized as follows. Section II introduces the background knowledge of this paper. Section III describes the proposed θ-DEA-DP in detail. Section IV provides experimental results and discussions. Finally, the conclusion is drawn in Section V.

II. Preliminaries and Background

This section presents the basic concepts and background information that are vital to the work in this paper.

A. Multi-Objective Optimization

A multi-objective optimization problem (MOP) can be formally stated as follows:[1]

$$\begin{aligned} \min \mathbf{f}(\mathbf{x}) &= (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^\top \\ \text{subject to } \mathbf{x} &\in \Omega \subseteq \mathbb{R}^n \end{aligned} \tag{1}$$

$\mathbf{x}$ is a n-dimensional decision vector in the decision space Ω, and $\mathbf{f} : \Omega \to \Theta \subseteq \mathbb{R}^m$, is an objective vector consisting of m objective functions, mapping Ω to the attainable objective space Θ. In the literature, a MOP is known as a many-objective optimization problem (MaOP) [2], [3] when $m > 3$.

Definition 1 (Pareto Dominance): A solution $\mathbf{u} \in \Omega$ is said to *Pareto dominate* another solution $\mathbf{v} \in \Omega$, denoted by $\mathbf{u} \prec \mathbf{v}$, iff $\forall i \in \{1, 2, \ldots, m\} : f_i(\mathbf{u}) \leq f_i(\mathbf{v})$ and $\exists j \in \{1, 2, \ldots, m\} : f_j(\mathbf{u}) < f_j(\mathbf{v})$.

Typically the objectives of a MOP are conflicting with each other, so there is no single solution that Pareto dominates all the others. Instead, there exists a set of equally good solutions in terms of Pareto dominance, called the *Pareto Set* (PS).

Definition 2 (Pareto Set): For a given MOP, the Pareto set is defined as $PS := \{\mathbf{x}^* \in \Omega \mid \nexists \mathbf{x} \in \Omega, \mathbf{x} \prec \mathbf{x}^*\}$.

The mapping of the PS into objective space is referred to as the Pareto front (PF), which is defined as follows:

Definition 3 (Pareto Front): For a given MOP, the Pareto front is defined as $PF := \{\mathbf{f}(\mathbf{x}^*) \mid \mathbf{x}^* \in PS\}$.

Generally, the goal of multi-objective optimization is to approximate the PF as much as possible. That is, the obtained objective vectors should be close to the PF (i.e., *convergence*), and also distributed as evenly as possible over the PF (i.e., *diversity*).

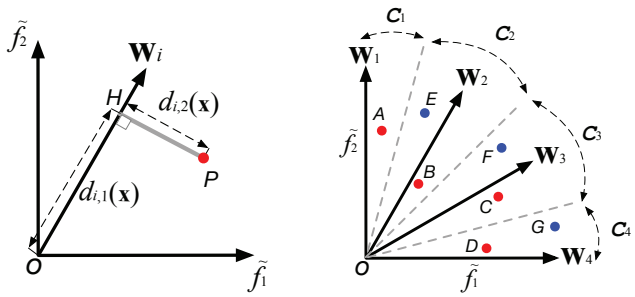
B. θ-Dominance

θ-dominance [41], [42] is a new dominance relationship that incorporates the decomposition idea. In θ-dominance, we need to predetermine a set of N uniformly distributed weight vectors (also called reference vectors) in the normalized objective space[2], denoted as $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N$. For any solution $\mathbf{x} \in \Omega$, suppose point P is its mapping in the normalized objective space and point H is the projection of P on the direction $\mathbf{w}_i$, then we can compute two distances corresponding to $\mathbf{w}_i$, that is, $d_{i,1}(\mathbf{x}) = |OH|$ and $d_{i,2}(\mathbf{x}) = |PH|$, where O is the origin. Further, the penalty boundary intersection (PBI) function [43] is defined as $\mathcal{F}_i(\mathbf{x}) = d_{i,1}(\mathbf{x}) + \theta d_{i,2}(\mathbf{x})$, where θ is a penalty parameter. To some extent, $\mathcal{F}_i(\mathbf{x})$ measures how close a solution $\mathbf{x}$ is to the PF along the direction $\mathbf{w}_i$. Fig. 1(a) illustrates $d_{i,1}$ and $d_{i,2}$ in two-dimensional objective space.

In θ-dominance, the N weight vectors serve the purpose of dividing the objective space into N clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N$. Every solution is exclusively assigned to one cluster according to the perpendicular distance from the weight vectors (i.e., $d_{i,2}$). Formally, $\mathbf{x} \in \mathcal{C}_j$, iff $j = \operatorname{argmin}_{i=1}^{N} d_{i,2}(\mathbf{x})$. Fig. 1(b) illustrates the division of objective space with $N = 4$ weight vectors. As an example, point E belongs to $\mathcal{C}_2$ because it is closest to $\mathbf{w}_2$ in terms of the perpendicular distance.

[1]In this paper, we shall assume that the goal is to minimize objectives.

[2]Without loss of generality, we assume that all the normalized objective values are greater than or equal to 0.



(a) Distances $d_{i,1}$ and $d_{i,2}$ (b) Objective space division

Fig. 1. Illustration of θ-dominance in the normalized objective space.

With the introduction of the PBI function and the clustering operator, θ-dominance is defined as follows:

Definition 4 (θ-Dominance): A solution $\mathbf{u} \in \Omega$ is said to *θ-dominate* another solution $\mathbf{v} \in \Omega$, denoted by $\mathbf{u} \prec_\theta \mathbf{v}$, iff $\exists j \in \{1, 2, ..., N\}$: $\mathbf{u} \in \mathcal{C}_j$, $\mathbf{v} \in \mathcal{C}_j$ and $\mathcal{F}_j(\mathbf{u}) < \mathcal{F}_j(\mathbf{v})$.

In essence, θ-dominance only distinguishes the solutions in the same cluster. In Fig. 1(b), B and F are θ-non-dominated to each other since they belong to different clusters; B θ-dominates E because they are both in cluster $\mathcal{C}_2$ and B achieves a better PBI function value with respect to $\mathbf{w}_2$.

Similar to Pareto dominance, θ-dominance defines a strict partial order over solutions [42]. As for non-dominated sorting based on θ-dominance, we can first rank the solutions in each cluster $\mathcal{C}_j$ according to $\mathcal{F}_j$. Then the best solutions in each cluster constitute the first θ-non-domination level, the second best solutions constitute the second level, and so on. Note that, unlike Pareto dominance, θ-dominance stresses both convergence and diversity. So when conducting survival selection, the solutions in the last θ-non-domination level to be included can be randomly chosen instead of using a secondary criterion such as crowding distance [44]. In Fig. 1(b), A, B, C and D constitute the first θ-non-domination level, while E, F and G constitute the second θ-non-domination level.

Note that θ-dominance is not Pareto compliant, which supports our approach of predicting both Pareto and θ-dominance in this paper. Moreover, owing to using the PBI function, θ-dominance can usually maintain higher selection pressure toward the PF in many-objective optimization compared to Pareto dominance [42].

C. Deep Feedforward Neural Networks

Deep feedforward neural networks (FNNs) are among the most typical deep learning models. Given a set of input-output examples $\mathbb{T} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(M)}, \mathbf{y}^{(M)})\}$, called *training set*, the goal of the FNN is to approximate the function that maps input $\mathbf{x}$ to output $\mathbf{y}$. To this end, the FNN defines a parametric function with the architecture as shown in Fig. 2. In a FNN with depth D,[3] there is an input layer (i.e., the 0-th layer) that just receives the input data $\mathbf{x}$ and an output layer (i.e., the D-th layer) that gives the output $\mathbf{y}$. Between the input and output layers, there are $D-1$ hidden layers. In the l-th layer ($1 \leq l \leq D$), it accepts the output of the $(l-1)$-th layer

[3]Generally, a feedforward neural network can be seen as a "deep" model if the depth D is higher than 2.

$\mathbf{a}^{[l-1]}$ as input, and produces its output $\mathbf{a}^{[l]}$ via the nonlinear transformation: $\mathbf{a}^{[l]} = h(\boldsymbol{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]})$, where $\mathbf{a}^{[0]} = \mathbf{x}$; $\boldsymbol{W}^{[l]}$ and $\mathbf{b}^{[l]}$ are the weight and bias parameters for the l-th layer respectively; $h(\cdot)$ is a nonlinear activation function. The learning task is to determine the parameters $\boldsymbol{W}^{[l]}$ and $\mathbf{b}^{[l]}$, $l = 1, 2, \ldots, D$ on the basis of training set $\mathbb{T}$.

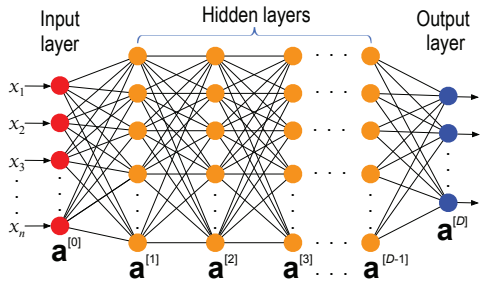


Fig. 2. The fully-connected deep feedforward neural network.

FNNs are commonly used for *multiclass classification*. In this case, the number of units in the output layer is equal to the number of classes K, and the softmax function is further employed to $\mathbf{a}^{[D]} = (a_1^{[D]}, a_2^{[D]}, \ldots, a_K^{[D]})^\top$ to obtain a discrete probability distribution for K classes: $p_k(\mathbf{x}) = \exp(a_k^{[D]}) / \sum_{k=1}^{K} \exp(a_k^{[D]})$, where $k = 1, 2, \ldots, K$. In multiclass classification, a target $\mathbf{y}^{(i)}$, $i = 1, 2, \ldots, M$ in training set $\mathbb{T}$ is usually expressed as a one-hot vector $\mathbf{y}^{(i)} = (y_1^{(i)}, y_2^{(i)}, \ldots, y_K^{(i)})$, where $y_k^{(i)} = 1$ iff $\mathbf{x}^{(i)}$ belongs to class k otherwise $y_k^{(i)} = 0$, $k = 1, 2, \ldots, K$. To learn the parameters $\boldsymbol{W} = (\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}, \ldots, \boldsymbol{W}^{[D]})$ and $\mathbf{b} = (\mathbf{b}^{[1]}, \mathbf{b}^{[2]}, \ldots, \mathbf{b}^{[D]})$, the following loss function for $\mathbb{T}$, called cross-entropy loss, is minimized with backpropagation [45].

$$J(\boldsymbol{W}, \mathbf{b}) = -\frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{K} y_k^{(i)} \log(p_k(\mathbf{x}^{(i)})) \tag{2}$$

With learned parameters, the predicted class at a new input $\mathbf{x}$ is given by $t(\mathbf{x}) = \operatorname{argmax}_{k=1}^{K} p_k(\mathbf{x})$. In addition, FNN can also indicate the probability (or confidence) that $\mathbf{x}$ belongs to the predicted class, which is given by $p(\mathbf{x}) = \max_{k=1}^{K} p_k(\mathbf{x})$.

In the deep learning era, a rectified linear unit (ReLU) is the most popular nonlinear activation function, which can usually make learning much faster compared to tanh or sigmoid function [38]. Furthermore, many more advanced techniques in parameter initialization (e.g., Kaiming initialization [46]), regularization (e.g., dropout [47]) and gradient-based optimization (e.g., Adam optimizer [48]) have been invented for efficient training of deep neural networks.

III. The Proposed Algorithm

A. Overview

The framework of the proposed θ-DEA-DP is described in Algorithm 1. First, we use the same approach as in θ-DEA [42] to generate a set of N structured weight vectors. In Step 2, we generate a set of initial solutions $\mathbb{P}$ using Latin hypercube sampling as described for ParEGO [8]. The number of solutions in $\mathbb{P}$ is set to $11n - 1$, where n is the number of decision variables. In Steps 3–9, all solutions in $\mathbb{P}$ are evaluated and added to an external archive $\mathbb{A}$, which is used to save all the *evaluated* solutions so far.

Algorithm 1 Framework of the Proposed θ-DEA-DP

1: $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\} \leftarrow$ InitializeWeightVectors(m)
2: $\mathbb{P} \leftarrow$ LatinHypercube(n)
3: $evals \leftarrow 0$
4: $\mathbb{A} \leftarrow \emptyset$
5: **for** $\mathbf{x} \in \mathbb{P}$ **do**
6: Evaluate$(\mathbf{x})$
7: $evals \leftarrow evals + 1$
8: $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbf{x}$
9: **end for**
10: p-net $\leftarrow$ Initiate-Pareto-Net$(\mathbb{A})$
11: θ-net $\leftarrow$ Initiate-θ-Net$(\mathbb{A})$
12: $\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_N^*\} \leftarrow$ Get-θ-Reps$(\mathbb{A})$
13: $\{\mathbf{y}_1^*, \mathbf{y}_2^*, \ldots, \mathbf{y}_N^*\} \leftarrow$ Get-Pareto-Reps$(\mathbb{A})$
14: $\mathbb{P} \leftarrow$ TruncatePopulation$(\mathbb{P}, N)$
15: **while** $evals <$ MaxEval **do**
16: $j \leftarrow$ ChooseTargetClusterIndex(N)
17: $\mathbb{Q} \leftarrow$ GenerateOffsprings$(\mathbb{P}, N^*)$
18: $\mathbf{z}^* \leftarrow$ TwoStagePreSelection$(\mathbb{Q}, \mathbf{x}_j^*, \mathbf{y}_j^*,$ p-net, θ-net$)$
19: Evaluate$(\mathbf{z}^*)$
20: $evals \leftarrow evals + 1$
21: $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbf{z}^*$
22: Update-Pareto-Net(p-net, $\mathbb{A}$)
23: Update-θ-Net(θ-net, $\mathbb{A}$)
24: Update-θ-Reps$(\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_N^*\}, \mathbb{A})$
25: Update-Pareto-Reps$(\{\mathbf{y}_1^*, \mathbf{y}_2^*, \ldots, \mathbf{y}_N^*\}, \mathbb{A})$
26: $\mathbb{P} \leftarrow$ TruncatePopulation$(\mathbb{P} \cup \mathbf{z}^*, N)$
27: **end while**

In Step 10, we train a neural network for Pareto dominance prediction using the solutions in $\mathbb{A}$, which we call Pareto-Net. Once Pareto-Net is trained, its functionality is that, when given any two solutions $\mathbf{u}$ and $\mathbf{v}$, it can output the predicted Pareto dominance relation between $\mathbf{u}$ and $\mathbf{v}$ (i.e., $\mathbf{u} \prec \mathbf{v}$, $\mathbf{v} \prec \mathbf{u}$ or $\mathbf{u} \simeq \mathbf{v}$) [4] along with a probability of belonging to the predicted dominance relation, without knowing their objective values. Similarly, in Step 11 we train a neural network called θ-Net for θ-dominance prediction.

As described in Section II-B, N weight vectors divide the objective space into N clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_N$. In Steps 12–13, we determine the θ-representative solution $\mathbf{x}_j^*$ and Pareto-representative solution $\mathbf{y}_j^*$ for each cluster $\mathcal{C}_j$, where $\mathbf{x}_j^*$ and $\mathbf{y}_j^*$ are both chosen from $\mathbb{A}$. The meaning of θ and Pareto representative solutions will be explained later.

In θ-DEA-DP, the population size is set to N, that is the same as the number of weight vectors. But the initial size of population $\mathbb{P}$ is $11n - 1$ which could be larger than N. So in Step 14, we use non-dominated sorting based on θ-dominance to select N elite solutions if $11n - 1 > N$.

Steps 15–27 are iterated until the maximum number of evaluations (MaxEval) is reached. In each iteration, we first select a cluster $\mathcal{C}_j$ to be considered (i.e., Step 16). To ensure

[4]For convenience, $\mathbf{u} \simeq \mathbf{v}$ (or $\mathbf{u} \simeq_\theta \mathbf{v}$) denotes $\mathbf{u}$ and $\mathbf{v}$ are Pareto (or θ) non-dominated to each other.

each cluster is approximately equally considered during optimization, all clusters are considered in rounds one by one, but their order is shuffled for each round.

In Step 17, we create an offspring population $\mathbb{Q}$ (with size N^*) from the current population $\mathbb{P}$ by using simulated binary crossover (SBX) and polynomial mutation [49]. As in θ-DEA, we adopt random mating selection. Note that $N^* \gg N$, so that good candidate solutions are very likely to be included in $\mathbb{Q}$.

Now we have a selected cluster $\mathcal{C}_j$ and a set of candidate solutions $\mathbb{Q}$. In Step 18, we use a two-stage preselection strategy to select a single solution $\mathbf{z}^*$ from $\mathbb{Q}$ for evaluation, assisted by the trained Pareto-Net and θ-Net. The solution $\mathbf{z}^*$ is expected to achieve considerable *improvement* over the θ-representative solution $\mathbf{x}_j^*$ and Pareto-representative solution $\mathbf{y}_j^*$.

Since we have a newly evaluated solution $\mathbf{z}^*$, we can create some new training examples for Pareto-Net and θ-Net. By combing previous and new training examples, we update the Pareto-Net and the θ-Net in Steps 22 and 23, respectively. Also because of $\mathbf{z}^*$, we need to update θ and Pareto representative solutions in Steps 24 and 25, respectively. In Step 26, we truncate the population size back to N using θ-non-dominated sorting.

Note that θ-DEA-DP does not incorporate a special normalization procedure. Currently, θ-DEA-DP uses the same method as ParEGO [8] to normalize the objective functions at the beginning of optimization.

In the following, Section III-B illustrates the concept of θ and Pareto representative solutions; Section III-C describes how to initiate and update Pareto-Net and θ-Net; Section III-D details the two-stage preselection strategy; Section III-E discusses several design principles of θ-DEA-DP.

B. Representative Solutions

For a cluster $\mathcal{C}_j$, its θ-representative solution $\mathbf{x}_j^*$ satisfies $\mathbf{x}_j^* \in \mathbb{A} \cap \mathcal{C}_j$ and $\nexists \mathbf{x} \in \mathbb{A}, \mathbf{x} \prec_\theta \mathbf{x}_j^*$. According to Definition 4, $\mathbf{x}_j^*$ is indeed the solution that achieves the minimum value of $\mathcal{F}_j$ among all solutions in $\mathbb{A} \cap \mathcal{C}_j$. Note that $\mathbf{x}_j^*$ can be null if there is no solution in $\mathbb{A}$ falling into $\mathcal{C}_j$ (i.e., $\mathbb{A} \cap \mathcal{C}_j = \emptyset$).

Suppose that we have obtained all θ-representative solutions $\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_N^*\}$. Among them (null solutions are ignored), we can find Pareto non-dominated solutions, and the set of their indices is denoted by $\mathbb{I} = \{i_1, i_2, \ldots, i_{N'}\}$. For a cluster $\mathcal{C}_j$, if $j \in \mathbb{I}$, its Pareto-representative solution $\mathbf{y}_j^*$ is the same with its θ-representative solution $\mathbf{x}_j^*$. If $j \notin \mathbb{I}$ and $\mathbf{x}_j^*$ is not null, we can certainly find an index $i \in \mathbb{I}$ satisfying $\mathbf{x}_i^* \prec \mathbf{x}_j^*$, and then $\mathbf{y}_j^*$ is set to $\mathbf{x}_i^*$. Note that if there is more than one such index i in $\mathbb{I}$, we pick the one that leads to the shortest Euclidean distance between $\mathbf{w}_i$ and $\mathbf{w}_j$. Lastly, if $\mathbf{x}_j^*$ is null, the Pareto-representative solution $\mathbf{y}_j^*$ is necessarily null.

Fig. 3 further illustrates representative solutions. In this figure, there is no representative solution for $\mathcal{C}_1$; A, B and C are θ-representative solutions for $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$, respectively. Because B and C are Pareto non-dominated solutions among the three θ-representative solutions, B and C are also Pareto-representative solutions for $\mathcal{C}_3$ and $\mathcal{C}_4$ respectively. As for $\mathcal{C}_2$, B is set to its Pareto-representative solution. This is because B Pareto dominates A, and $\mathbf{w}_2$ is closer to $\mathbf{w}_3$ than to $\mathbf{w}_4$.

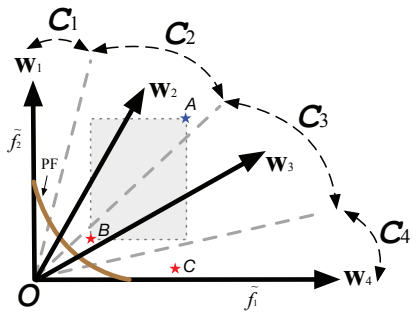


Fig. 3. Illustration of θ and Pareto representative solutions.

C. Surrogates Based on Dominance Prediction

Since the procedures presented in this subsection apply to Pareto-Net and θ-Net similarly, Pareto-Net is used as the example for illustration. We formulate dominance prediction as a three-class classification problem (A dominates B, B dominates A, or neither dominates the other).

1) Initiate the Surrogates: To initiate Pareto-Net in Step 10 of Algorithm 1, we need to first construct a training set $\mathbb{T}$ using the evaluated solutions in archive $\mathbb{A}$. Suppose $\mathbb{A} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_S\}$, where S is the current number of solutions in $\mathbb{A}$. The input of each training example in $\mathbb{T}$ is a vector that concatenates any two solutions $\mathbf{x}_i$ and $\mathbf{x}_j$ from $\mathbb{A}$ ($i \neq j$) denoted by $\mathbf{x}_{i,j} = [\mathbf{x}_i, \mathbf{x}_j]$, and its corresponding output $\mathbf{y}_{i,j}$ is a one-hot vector indicating the class $\mathbf{x}_{i,j}$ belongs to. In this paper, we say that $\mathbf{x}_{i,j}$ belongs to class 1 iff $\mathbf{x}_i \prec \mathbf{x}_j$, belongs to class 2 iff $\mathbf{x}_j \prec \mathbf{x}_i$, and belongs to class 3 iff $\mathbf{x}_i \simeq \mathbf{x}_j$. In this way, we create a training set $\mathbb{T}$ containing $S(S-1)$ input-output examples.

With this training set we train the classifier (i.e., Pareto-Net) which is a fully-connected FNN as shown in Fig. 2. During training, we use Kaiming initialization [46] to initialize the FNN parameters, use weight decay as a regularizer to reduce overfitting, and run the Adam optimizer [48] with a fixed minibatch size to optimize FNN parameters. These are standard techniques for training deep FNNs and have been implemented in PyTorch [50].

Our key observation for dominance prediction is that it is usually an *imbalanced* classification problem [51], which will be further discussed in Section III-E. To alleviate class imbalance, we adopt the common strategy to introduce a weighting factor for each class in cross-entropy loss. Corresponding to Eq. (2), the weighted cross-entropy loss is defined as:

$$J(\boldsymbol{W}, \mathbf{b}) = -\frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{K} \alpha_k y_k^{(i)} \log(p_k(\mathbf{x}^{(i)})) \qquad (3)$$

where α_k ($k = 1, 2, \ldots, K$) is the inverse class frequency of class k in the training set.

Given a concatenated vector of two solutions denoted by $[\mathbf{u}, \mathbf{v}]$, the trained Pareto-Net can predict the class of $[\mathbf{u}, \mathbf{v}]$ that is given by $t(\mathbf{u}, \mathbf{v})$, and also provide the estimated probability of belonging to the predicted class that is given by $p(\mathbf{u}, \mathbf{v})$. The drawback of this prediction function is that the predicted dominance relation between $\mathbf{u}$ and $\mathbf{v}$ may depend on whether the input vector is $[\mathbf{u}, \mathbf{v}]$ or $[\mathbf{v}, \mathbf{u}]$ when Pareto-Net does not

perfectly capture the characteristics of Pareto dominance. For example, Pareto-Net may output $t(\mathbf{u},\mathbf{v})=1$ indicating $\mathbf{u}\prec\mathbf{v}$ and output $t(\mathbf{v},\mathbf{u})=3$ indicating $\mathbf{u}\simeq\mathbf{v}$, which leads to inconsistency. To remedy this, we use the input vector that results in larger prediction probability/confidence to determine the dominance relation, so the predicted class of $[\mathbf{u},\mathbf{v}]$ and the corresponding prediction probability is revised to:

$$\hat{t}(\mathbf{u},\mathbf{v})=\begin{cases} t(\mathbf{u},\mathbf{v}), & \text{if } p(\mathbf{u},\mathbf{v})\geq p(\mathbf{v},\mathbf{u}) \\ 3-t(\mathbf{v},\mathbf{u}), & \text{if } p(\mathbf{u},\mathbf{v})<p(\mathbf{v},\mathbf{u}) \text{ and } t(\mathbf{v},\mathbf{u})\neq 3 \\ 3, & \text{if } p(\mathbf{u},\mathbf{v})<p(\mathbf{v},\mathbf{u}) \text{ and } t(\mathbf{v},\mathbf{u})=3 \end{cases} \quad (4)$$

and $\hat{p}(\mathbf{u},\mathbf{v})=\max\{p(\mathbf{u},\mathbf{v}),p(\mathbf{v},\mathbf{u})\}$. For convenience, we specify $\hat{t}(\mathbf{u},\mathbf{v})=\hat{p}(\mathbf{u},\mathbf{v})=0$ if either $\mathbf{u}$ or $\mathbf{v}$ is null.

Analogously, we have the prediction function $\hat{t}_\theta(\mathbf{u},\mathbf{v})$ along with the prediction probability $\hat{p}_\theta(\mathbf{u},\mathbf{v})$ for θ-Net. $\hat{t}_\theta(\mathbf{u},\mathbf{v})=1$, $\hat{t}_\theta(\mathbf{u},\mathbf{v})=2$ and $\hat{t}_\theta(\mathbf{u},\mathbf{v})=3$ imply the predicted θ-dominance relation $\mathbf{u}\prec_\theta\mathbf{v}$, $\mathbf{v}\prec_\theta\mathbf{u}$ and $\mathbf{u}\simeq_\theta\mathbf{v}$, respectively.

2) Update the Surrogates: In each iteration, we intend to update the Pareto-Net (i.e., Step 22 of Algorithm 1). If $|\mathbb{A}|<T_{\max}$, all solutions in $\mathbb{A}$ are used to update Pareto-Net, otherwise only the $T_{\max}$ most recently evaluated solutions in $\mathbb{A}$ are considered in order to reduce computational cost, where $T_{\max}$ is a predefined parameter.

Suppose $\mathbf{z}^*$ is a newly evaluated solution that has just been added to $\mathbb{A}$ (see Step 21 of Algorithm 1), so it will definitely be considered in the updating. The other solutions considered except $\mathbf{z}^*$ constitute the set $\mathbb{A}'=\{\mathbf{x}_{i_1},\mathbf{x}_{i_2},\ldots,\mathbf{x}_{i_{T'}}\}$, where $T'=\max\{|\mathbb{A}|,T_{\max}\}-1$. We can construct $2T'$ new examples that have not been seen by the Pareto-Net, with input of these examples as $[\mathbf{x}_{i_1},\mathbf{z}^*]$, $[\mathbf{x}_{i_2},\mathbf{z}^*]$, …, $[\mathbf{x}_{i_{T'}},\mathbf{z}^*]$, $[\mathbf{z}^*,\mathbf{x}_{i_1}]$, $[\mathbf{z}^*,\mathbf{x}_{i_2}]$, …, $[\mathbf{z}^*,\mathbf{x}_{i_{T'}}]$. To estimate the current performance of Pareto-Net, we test it on the $2T'$ examples. Note that due to class imbalance, the overall accuracy is usually not a meaningful indicator of performance. Hence we record the accuracy on each class, denoted by acc_1, acc_2 and acc_3 respectively, and obtain $acc_{\min}=\min_{k=1}^{3}acc_k$.

With $acc_{\min}$, we first determine whether to update or not. If $acc_{\min}>\gamma$ (γ is an accuracy threshold), Pareto-Net may currently work pretty well so we just keep it unchanged. Otherwise, we continue training Pareto-Net to update its model parameters. The targeted training set here is constructed by pairing any two solutions in $\mathbb{A}'\cup\mathbf{z}^*$. Due to the gradient-based training mechanism, we do not need to train Pareto-Net from scratch. Instead, we use the current parameters of Pareto-Net as starting point for the Adam optimizer.

The last issue for the updating is to determine the number of training epochs[5] denoted by E_{upd}. Suppose that E_{init} is the number of training epochs used in initiating Pareto-Net, then we specify E_{upd} according to $acc_{\min}$ and E_{init} as follows:

$$E_{\text{upd}}=(1-\frac{acc_{\min}}{\gamma})E_{\text{init}} \quad (5)$$

This equation incorporates two considerations: 1) It is reasonable that E_{upd} is generally smaller than E_{init} since the Pareto-Net has already been trained for some time before the updating; 2) A higher estimated accuracy $acc_{\min}$ is likely to require smaller adjustments of model parameters.

[5]One epoch means one complete pass through all the training examples.

D. Two-Stage Preselection Strategy

In each iteration, we first choose a target cluster $\mathcal{C}_j$ and sample a set of candidate solutions $\mathbb{Q}$ (see Steps 16–17 of Algorithm 1). Our goal is to select a single solution from $\mathbb{Q}$ for evaluation with the aid of Pareto-Net and θ-Net. To achieve this, we employ a two-stage preselection strategy.

1) Comparison With Representative Solutions: The first stage is responsible for selecting a subset of $\mathbb{Q}$ in which solutions are likely to make improvements to the current representative solutions. According to whether a θ-representative solution $\mathbf{x}_j^*$ is null or not, there are two cases in the first stage that need to be handled separately.

If $\mathbf{x}_j^*$ is not null, we compare each solution in $\mathbb{Q}$ with $\mathbf{x}_j^*$ in terms of θ-dominance, and compare it with $\mathbf{y}_j^*$ in terms of Pareto dominance. More specifically, for every solution $\mathbf{z}$ in $\mathbb{Q}$, we obtain $\hat{t}_\theta(\mathbf{z},\mathbf{x}_j^*)$ and $\hat{t}(\mathbf{z},\mathbf{y}_j^*)$ using θ-Net and Pareto-Net, respectively. According to these values we can divide the solutions in $\mathbb{Q}$ into four categories:

1) $\mathbb{Q}_1=\{\mathbf{z}\in\mathbb{Q}\mid\hat{t}_\theta(\mathbf{z},\mathbf{x}_j^*)=1\wedge\hat{t}(\mathbf{z},\mathbf{y}_j^*)=1\}$
2) $\mathbb{Q}_2=\{\mathbf{z}\in\mathbb{Q}\mid\hat{t}_\theta(\mathbf{z},\mathbf{x}_j^*)=1\wedge\hat{t}(\mathbf{z},\mathbf{y}_j^*)=3\}$
3) $\mathbb{Q}_3=\{\mathbf{z}\in\mathbb{Q}\mid\hat{t}_\theta(\mathbf{z},\mathbf{x}_j^*)=3\wedge\hat{t}(\mathbf{z},\mathbf{y}_j^*)=1\}$
4) $\mathbb{Q}_4=\{\mathbf{z}\in\mathbb{Q}\mid\hat{t}_\theta(\mathbf{z},\mathbf{x}_j^*)=2\vee\hat{t}(\mathbf{z},\mathbf{y}_j^*)=2\}$

We shall ignore solutions in $\mathbb{Q}_4$ because all are predicted to be θ-dominated by $\mathbf{x}_j^*$ or Pareto dominated by $\mathbf{y}_j^*$. Among $\mathbb{Q}_1$, $\mathbb{Q}_2$ and $\mathbb{Q}_3$, we only select one category for further consideration and ignore solutions of the other categories. In the proposed algorithm, $\mathbb{Q}_1$ is selected with the highest priority, followed by $\mathbb{Q}_2$ and $\mathbb{Q}_3$. If $\mathbb{Q}_1\neq\emptyset$, we just select $\mathbb{Q}_1$. $\mathbb{Q}_2$ or $\mathbb{Q}_3$ are selected only if the categories with higher priority are empty.

If $\mathbf{x}_j^*$ is null, this implies that at least one cluster does not have evaluated solutions. In this case, we want to find a solution belonging to the unexplored cluster in order to enhance diversity. So we only consider solutions in $\mathbb{Q}$ that are θ-non-dominated to all current non-null θ-representative solutions. We designate this set as category $\mathbb{Q}_5$, formally defined as: $\mathbb{Q}_5=\{\mathbf{z}\in\mathbb{Q}\mid\forall i\in\{1,2,\ldots,N\},\hat{t}_\theta(\mathbf{z},\mathbf{x}_i^*)=3\}$.

Note that for both cases, we only keep a maximum of $Q_{\max}$ solutions if the number of solutions in the selected category is larger than parameter $Q_{\max}$. To do this, we compute the sum of prediction probabilities related to each solution $\mathbf{z}$ in the selected category, denoted by $p_{\text{sum}}(\mathbf{z})$. This reads for the first case $p_{\text{sum}}(\mathbf{z})=p_\theta(\mathbf{z},\mathbf{x}_j^*)+p(\mathbf{z},\mathbf{y}_j^*)$, while for the second case we have $p_{\text{sum}}(\mathbf{z})=\sum_{i=1}^{N}p_\theta(\mathbf{z},\mathbf{x}_i^*)$. Then we can select $Q_{\max}$ largest solutions in terms of $p_{\text{sum}}(\mathbf{z})$.[6]

2) Comparison Within the Selected Category of Solutions: Suppose that in the first stage, we have selected a category $\mathbb{Q}_k$, $k\in\{1,2,3,5\}$ for consideration, denoted as $\mathbb{Q}_k=\{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_Q\}$. The second preselection stage is now responsible for selecting one solution from $\mathbb{Q}_k$ that is likely to

[6]Although it rarely occurs, we may fail to select any solution if $\mathbb{Q}_1=\mathbb{Q}_2=\mathbb{Q}_3=\emptyset$ in the first case and $\mathbb{Q}_5=\emptyset$ in the second case. At this time, we just resample the set of candidate solutions $\mathbb{Q}$ using genetic operators.

yield the largest improvement over the representative solutions. To do this, we first use Pareto-Net (θ-Net) to predict the Pareto dominance (θ-dominance) relation between any two solutions in $\mathbb{Q}_k$. Then for each solution $\mathbf{z}_i$ in $\mathbb{Q}_k$, we define the following metric, which we call *expected* dominance number (EDN):

$$e(\mathbf{z}_i) = \sum_{j \neq i} I(\hat{t}(\mathbf{z}_i, \mathbf{z}_j) = 1)\hat{p}(\mathbf{z}_i, \mathbf{z}_j) \tag{6}$$

where $I(\cdot)$ is the indicator function. This metric basically counts the number of solutions in $\mathbb{Q}_k$ that are predicted to be Pareto dominated by $\mathbf{z}_i$, weighted by the probability with which $\mathbf{z}_i$ Pareto dominates. Similarly, we can define an EDN in terms of θ-dominance for $\mathbf{z}_i$ as follows:

$$e_\theta(\mathbf{z}_i) = \sum_{j \neq i} I(\hat{t}_\theta(\mathbf{z}_i, \mathbf{z}_j) = 1)\hat{p}_\theta(\mathbf{z}_i, \mathbf{z}_j) \tag{7}$$

Finally, we rank the solutions in $\mathbb{Q}_k$ according to the sum $e_{\text{sum}}(\mathbf{z}_i) = e(\mathbf{z}_i) + e_\theta(\mathbf{z}_i)$. The solution with maximal value $e_{\text{sum}}(\mathbf{z}_i)$ is selected for function evaluation.

E. Discussion

In θ-DEA-DP, we combine Pareto and θ-dominance prediction. As explained in Section I, it is undesirable to only use Pareto dominance prediction. On the other hand, it is also not a good practice to only use θ-dominance prediction, because this may lead to slow convergence speed. Take Fig. 3 for an illustration and suppose that we are searching for a solution that θ-dominates the current θ-representative solution A in cluster $\mathcal{C}_2$, assisted by θ-Net. The selected solution for evaluation will likely fall into the shaded area in Fig. 3 and will thereby be Pareto dominated by solution B in another cluster $\mathcal{C}_3$. As a result, many newly evaluated solutions may not help to improve the overall quality of the current solution set, hindering fast convergence towards the PF. For a similar reason, we distinguish Pareto and θ-representative solutions for a cluster. For example, if we use A as the Pareto representative solution for $\mathcal{C}_2$ instead of B, then many promising solutions recognized by Pareto-Net would be located in the shaded area, which indeed cannot contribute to an overall quality improvement. Another potential benefit of using both Pareto-Net and θ-Net is that the optimization process may still be guided well even if one of the two surrogates performs poorly, leading to better robustness of optimization.

In θ-DEA-DP, the number of offsprings N^* should be sufficiently large to sample more and better candidate solutions. However, if N^* is too large, the surrogates will be used too frequently for comparing solutions. This may introduce too much approximation noise into the selection process since surrogates are usually imperfect, thereby making it hard to pick out a very good solution. A large N^* may also incur unaffordable computational costs. For these reasons we have to set a maximum category size $Q_{\max}$ in two-stage preselection.

As mentioned in Section III-C1, it is important to view dominance prediction as an imbalanced classification problem. For Pareto dominance, it is well known that the overwhelming majority of solutions will become non-dominated to each other as the number of objectives increases. As for θ-dominance, an extreme case is that all solutions involved in the training set are in the same cluster. In this case, there will be no training example belonging to class 3. Conversely, if all solutions are uniformly distributed in N clusters and each cluster has L solutions, then we can deduce that there are $L(L-1)N/2$ training examples in both class 1 and class 2, and $N(N-1)L^2$ in class 3. We therefore know that the ratio between class 3 and class 1 (or class 2) is larger than $2(N-1):1$. Since the value of N usually ranges from tens to hundreds, the class distribution will be highly imbalanced.

In the training of Pareto-Net and θ-Net, we do not apply dropout [47]. Although dropout is commonly used as a regularization strategy in deep learning, we find that it usually worsens the final optimization performance in our experiments and a very modest weight decay without dropout is usually a better configuration. Interestingly, Snoek *et al.* [39] had a similar observation in the context of single-objective Bayesian optimization. An analysis of the underlying reasons will be undertaken in future work.

In two-stage preselection, it is intuitive that solutions in $\mathbb{Q}_1$ are considered with highest priority, since each solution in $\mathbb{Q}_1$ can improve both Pareto and θ-representative solutions according to the surrogates. We further prefer to promote convergence without disturbing the diversity preservation mechanism, so $\mathbb{Q}_2$ is given higher priority than $\mathbb{Q}_3$.

The main difference between θ-DEA-DP and θ-DEA lies in which offspring solutions are evaluated and then involved in the survival selection process. In θ-DEA, all offspring solutions are evaluated, whereas in θ-DEA-DP, only a single offspring solution is picked out for evaluation via a two-stage preselection assisted by dominance prediction.

IV. Experiments

In this section, we first provide the basic settings of our experiments. Then we evaluate and validate the performance of θ-DEA-DP on multi-objective and many-objective optimization problems, respectively. Lastly, we investigate the effects of different components of θ-DEA-DP.

A. Experimental Design

1) Test Problems: We select test problems from three widely used multi-objective benchmark suites.

The first benchmark suite, ZDT [52], contains six two-objective test problems, which introduce different problem difficulties for evolutionary optimization using a scheme suggested by Deb [53]. We choose four unconstrained problems here, referred to as ZDT1–ZDT4.

The second benchmark suite, DTLZ [54], defines a set of problems that are scalable to any number of objectives. Following the practice in [8], we consider four of these problems called DTLZ1, DTLZ2, DTLZ4 and DTLZ7. Moreover, for DTLZ1, we replace 20π in the cosine term with 2π to reduce the ruggedness of the function, as suggested in [8].

The third benchmark suite, WFG [55], is constructed via a series of transformations that are sequentially applied to decision variables. Each transformation can introduce a desirable feature (e.g., non-separability) into the problem. Similar to

DTLZ, each WFG problem is also scalable in the number of objectives. Here we choose two problems WFG6 and WFG7 for investigation, as in [40].

In the supplementary material, we report on additional experiments that have been conducted on another four WFG problems (i.e., WFG4, WFG5, WFG8 and WFG9) and four real-world problems including a pressure vessel design problem, a two-bar truss design problem, a welded beam design problem and a conceptual marine design problem.

TABLE I
MULTI-OBJECTIVE TEST PROBLEMS USED IN THIS STUDY

Problem	No. of Objectives (m)	No. of Variables (n)	Features
ZDT1	2	10	Convex
ZDT2	2	10	Concave
ZDT3	2	10	Convex, Disconnected
ZDT4	2	10	Concave, Multi-modal
DTLZ1	2	6	Linear, Multi-modal
DTLZ2	3	8	Concave
DTLZ4	3	8	Concave, Biased
DTLZ7	3	8	Mixed, Disconnected
WFG6	3	10	Concave, Non-separable
WFG7	3	10	Concave, Biased

In Table I, we summarize the MOPs (with two or three objectives) used in our experiments, which have a variety of features. Moreover, for DTLZ1, DTLZ2, DTLZ4, DTLZ7, WFG6 and WFG7, we also consider their five and eight-objective versions, in order to evaluate the performance on MaOPs. The number of decision variables is set to 10 for all five and eight-objective problems.

Note that the position-related parameter needs to be specified for WFG6 and WFG7. In the experiments, this parameter is always set to $m-1$ according to [56] and [42], where m is the number of objectives.

2) Performance Indicator: We use the inverted generational distance (IGD) to evaluate the performance of an algorithm. Let $\mathbb{P}^*$ be a set of uniformly distributed points on the PF and $\mathbb{S}$ be the set of points obtained by an algorithm in the objective space. Then IGD can be computed as follows:

$$\text{IGD}(\mathbb{S},\mathbb{P}^*)=\sum_{\mathbf{v}\in\mathbb{P}^*}\frac{d(\mathbf{v},\mathbb{S})}{|\mathbb{P}^*|} \tag{8}$$

where $d(\mathbf{v},\mathbb{S})$ is the minimum Euclidean distance between $\mathbf{v}$ and the points in $\mathbb{P}^*$. IGD can provide combined information about convergence and diversity of a solution set, and smaller IGD value means better performance. To calculate IGD reliably, $|\mathbb{P}^*|$ should be large enough to represent the PF very well. In our experiments, $|\mathbb{P}^*|$ is roughly 500, 10^3, 10^5 and 10^6 for 2, 3, 5 and 8-objective problems, respectively.

Note that hypervolume (HV) [11] is another popular indicator of performance used in the literature. HV provides a well-established alternative to IGD when the true PF is not known a prior. So in the supplementary material, we use HV for the performance evaluation on four real-world problems.

3) Algorithms in Comparison: We compare the proposed θ-DEA-DP with the following related algorithms:

- **θ-DEA** [42]: This algorithm combines Pareto and θ-non-dominated sorting to rank solutions in survival selection. Although θ-DEA is not a surrogate-assisted MOEA per se, we include it as a baseline for comparison considering its close relationship with θ-DEA-DP.
- **ParEGO** [8]: This algorithm aggregates multiple objective values of a solution into a single function value via a parameterized weight vector. Then, similar to EGO [6], a solution is selected for evaluation by maximizing the expected improvement (EI) criterion with respect to the current aggregation function. By choosing a different weight vector in each iteration ParEGO is expected to implicitly maintain the diversity of evaluated solutions.
- **DomRank** [16]: This algorithm is also an extension of EGO for MOPs like ParEGO. But it uses a different scalarization scheme based on Pareto dominance. In DomRank, the aggregation function value of a solution is proportional to the number of evaluated solutions that dominate it.
- **MOEA/D-EGO** [9]: This algorithm employs a similar scalarization scheme as ParEGO. But unlike ParEGO, MOEA/D-EGO considers all aggregation functions rather than a single one in each iteration and maximizes their corresponding EI values simultaneously using MOEA/D-DE [57] in order to generate several points for function evaluation.
- **CSEA** [35]: This algorithm selects a set of reference solutions from the evaluated solutions to construct the classification boundary. Based on this, a classifier is built to divide the candidate solutions into good and bad so as to guide the selection of promising solutions for function evaluation.

Similar to ParEGO and MOEA/D-EGO, θ-DEA-DP uses a number of weight vectors to aggregate objectives, which is reflected in θ-dominance. In the second stage of preselection, θ-DEA-DP ranks the candidate solutions by dominance comparisons, which is somewhat similar to DomRank. Like CSEA, θ-DEA-DP is also based on classification-based surrogates.

We implement θ-DEA-DP and θ-DEA in Python. In θ-DEA-DP, the deep learning models are built using PyTorch [50]. For the sake of reproducible research, the source code of θ-DEA-DP has been made available online.[7] For ParEGO and DomRank, we use the Python implementation[8] by Rahat *et al.* [16]. As for MOEA/D-EGO and CSEA, we use the Matlab implementation[9] available in the PlatEMO [58] platform.

We run each algorithm 21 times independently on each test problem. For each run, we use the Pareto non-dominated set of all evaluated solutions to calculate IGD. To test for statistical significance, we carry out the Wilcoxon rank sum test at a 5% significance level on IGD results obtained from two competing algorithms. Moreover, the Holm-Bonferroni method is used to counteract the problem of multiple comparisons.

4) Parameter Settings: In all algorithms compared, Latin hypercube sampling is used to generate $11n-1$ initial points for function evaluation, where n is the number of decision variables. To ensure fair comparison, all of them use the same

[7]https://github.com/yyxhdy/tdeadp
[8]https://bitbucket.org/arahat/gecco-2017
[9]https://github.com/BIMK/PlatEMO

termination criterion per run. That is, the maximum number of function evaluations is set to 250 for two and three-objective problems [8], 300 for five-objective problems, and 400 for eight-objective problems.

TABLE II
WEIGHT VECTOR SETTINGS

No. of Objectives (m)	Divisions (H)	No. of Weight Vectors (N)
2	10	11
3	4	15
5	2, 2	30
8	2, 1	44

In θ-DEA-DP, θ-DEA, ParEGO and MOEA/D-EGO, a set of predefined weight vectors is required. We use the same method as in [42] to produce structured weight vectors. Table II lists weight vector settings for the problem with different number of objectives, where H is a parameter that controls weight vector generation. To avoid only generating boundary weight vectors, two-layered weight vectors [40] are used for problems with five and eight objectives.

For θ-DEA-DP and θ-DEA, the penalty parameter θ in the PBI function is set to 5, and population size is set to the number of weight vectors. CSEA adopts the same population size.

TABLE III
PARAMETER SETTING FOR θ-DEA-DP

Module	Parameter	Value
Evolution	Crossover Probability (p_c)	1.0
	Mutation Probability (p_m)	$1/n$
	Distribution Index for Crossover (η_c)	30
	Distribution Index for Mutation (η_m)	20
	Number of Sampled Offsprings (N^*)	7000
Surrogate	Depth of FNN (D)	3
	No. of Units in Each Hidden Layer (U)	200
	Weight Decay Coefficient (λ)	0.00001
	Epochs for Initiating FNN (E_{init})	20
	Batch Size (B)	32
	Learning Rate (ϵ)	0.001
	Maximum Size for Updating ($T_{\max}$)	$11n + 24$
	Accuracy Threshold (γ)	0.9
	Maximum Category Size ($Q_{\max}$)	300

For θ-DEA-DP, other parameter values are shown in Table III grouped by different modules. Pareto-Net and θ-Net share the same hyperparameter setting. Note that these parameter values are just set to make θ-DEA-DP perform reasonably well but are probably not the best. The careful parametric study will be left for future research.

As for the other parameters in θ-DEA, ParEGO, DomRank, MOEA/D-EGO and CSEA, we simply follow parameter setting recommendations of the original studies [8], [9], [16], [35], [42].

B. Performance on Multi-Objective Optimization Problems

Table IV shows the results on the multi-objective problems depicted in Table I, where the best, median, and worst IGD values are reported. For each problem, the $+$, $-$ and $\approx$ symbols indicate that, compared to θ-DEA-DP, the corresponding algorithm performs significantly worse, significantly better or comparable, respectively. From Table IV, θ-DEA-DP significantly outperforms all the other algorithms on 8 out of 10 problems. For the remaining two problems, the next-best algorithm is barely statistically comparable to θ-DEA-DP (i.e., CSEA on DTLZ4 and MOEA/D-EGO on DTLZ7). θ-DEA generally performs much worse than θ-DEA-DP, demonstrating the effectiveness of the dominance prediction based surrogates. Moreover, it is worth highlighting that θ-DEA-DP usually outperforms the other compared algorithms by a large margin on ZDT1–ZDT3, DTLZ1 and DTLZ7. Particularly on ZDT2 and DTLZ1, the median IGD values achieved by θ-DEA-DP are about one order of magnitude lower than those by the best-performing counterparts (i.e., MOEA/D-EGO on ZDT2 and ParEGO on DTLZ1).

TABLE IV
BEST, MEDIAN AND WORST IGD VALUES ON TWO AND THREE-OBJECTIVE TEST PROBLEMS. BEST PERFORMANCE IS SHOWN IN BOLD

Problem	θ-DEA -DP	θ-DEA	ParEGO	DomRank	MOEA/D -EGO	CSEA
ZDT1	**1.16E-02**	3.46E-01	1.00E-01	1.18E-01	3.52E-02	1.61E-01
	1.61E-02	5.94E-01	1.37E-01	2.47E-01	6.30E-02	3.86E-01
	2.28E-02	9.74E-01	2.65E-01	4.37E-01	2.50E-01	7.91E-01
		+	+	+	+	+
ZDT2	**1.49E-02**	7.25E-01	1.61E-01	1.30E-01	3.02E-02	4.09E-01
	1.85E-02	1.30E+00	2.71E-01	1.80E-01	1.36E-01	1.06E+00
	2.84E-02	2.04E+00	4.32E-01	2.81E-01	3.74E-01	1.70E+00
		+	+	+	+	+
ZDT3	**3.82E-02**	2.77E-01	7.04E-02	4.21E-02	1.57E-01	1.66E-01
	6.05E-02	4.38E-01	1.03E-01	1.04E-01	2.85E-01	4.12E-01
	1.35E-01	5.91E-01	1.75E-01	1.81E-01	5.85E-01	7.43E-01
		+	+	+	+	+
ZDT4	**1.05E+01**	2.41E+01	3.40E+01	4.38E+01	6.16E+01	2.34E+01
	2.80E+01	3.53E+01	5.70E+01	6.80E+01	8.46E+01	4.78E+01
	3.68E+01	4.97E+01	7.54E+01	8.51E+01	9.49E+01	6.66E+01
		+	+	+	+	+
DTLZ1	**2.15E-02**	4.74E-01	1.93E-01	1.05E+00	1.55E+00	2.64E+00
	5.38E-02	6.26E+00	4.12E-01	4.61E+00	4.55E+00	1.22E+01
	1.14E-01	1.99E+01	1.75E+00	1.66E+01	1.14E+01	4.39E+01
		+	+	+	+	+
DTLZ2	**1.10E-01**	1.46E-01	1.47E-01	1.59E-01	2.24E-01	1.69E-01
	1.23E-01	1.77E-01	1.58E-01	1.82E-01	2.58E-01	2.20E-01
	1.44E-01	2.11E-01	1.70E-01	2.43E-01	2.83E-01	2.85E-01
		+	+	+	+	+
DTLZ4	**1.21E-01**	1.99E-01	3.99E-01	3.92E-01	4.54E-01	1.45E-01
	2.12E-01	3.23E-01	4.78E-01	4.80E-01	5.32E-01	2.25E-01
	3.13E-01	6.01E-01	5.44E-01	5.68E-01	6.31E-01	6.03E-01
		+	+	+	+	≈
DTLZ7	**7.26E-02**	5.93E-01	2.30E-01	1.93E-01	8.09E-02	2.59E-01
	9.43E-02	8.80E-01	2.95E-01	5.02E-01	1.07E-01	8.77E-01
	1.72E-01	1.59E+00	4.42E-01	1.01E+00	**1.33E-01**	1.51E+00
		+	+	+	≈	+
WFG6	**1.51E-01**	1.91E-01	2.31E-01	2.00E-01	2.07E-01	2.01E-01
	2.04E-01	2.19E-01	2.41E-01	2.26E-01	2.22E-01	2.10E-01
	2.30E-01	2.46E-01	2.65E-01	2.57E-01	2.37E-01	2.33E-01
		+	+	+	+	+
WFG7	**1.28E-01**	1.46E-01	1.82E-01	1.64E-01	1.73E-01	1.55E-01
	1.46E-01	1.61E-01	1.94E-01	1.77E-01	1.94E-01	1.77E-01
	1.70E-01	1.86E-01	2.09E-01	1.89E-01	2.01E-01	1.85E-01
		+	+	+	+	+
$+/-/\approx$		10/0/0	10/0/0	10/0/0	9/0/1	9/0/1

To illustrate the distributions of solutions in the objective space, Fig. 4 shows final non-dominated solutions obtained in the run with median IGD value for each surrogate-assisted algorithm for ZDT2, ZDT3, DTLZ1 and DTLZ2. From Fig.

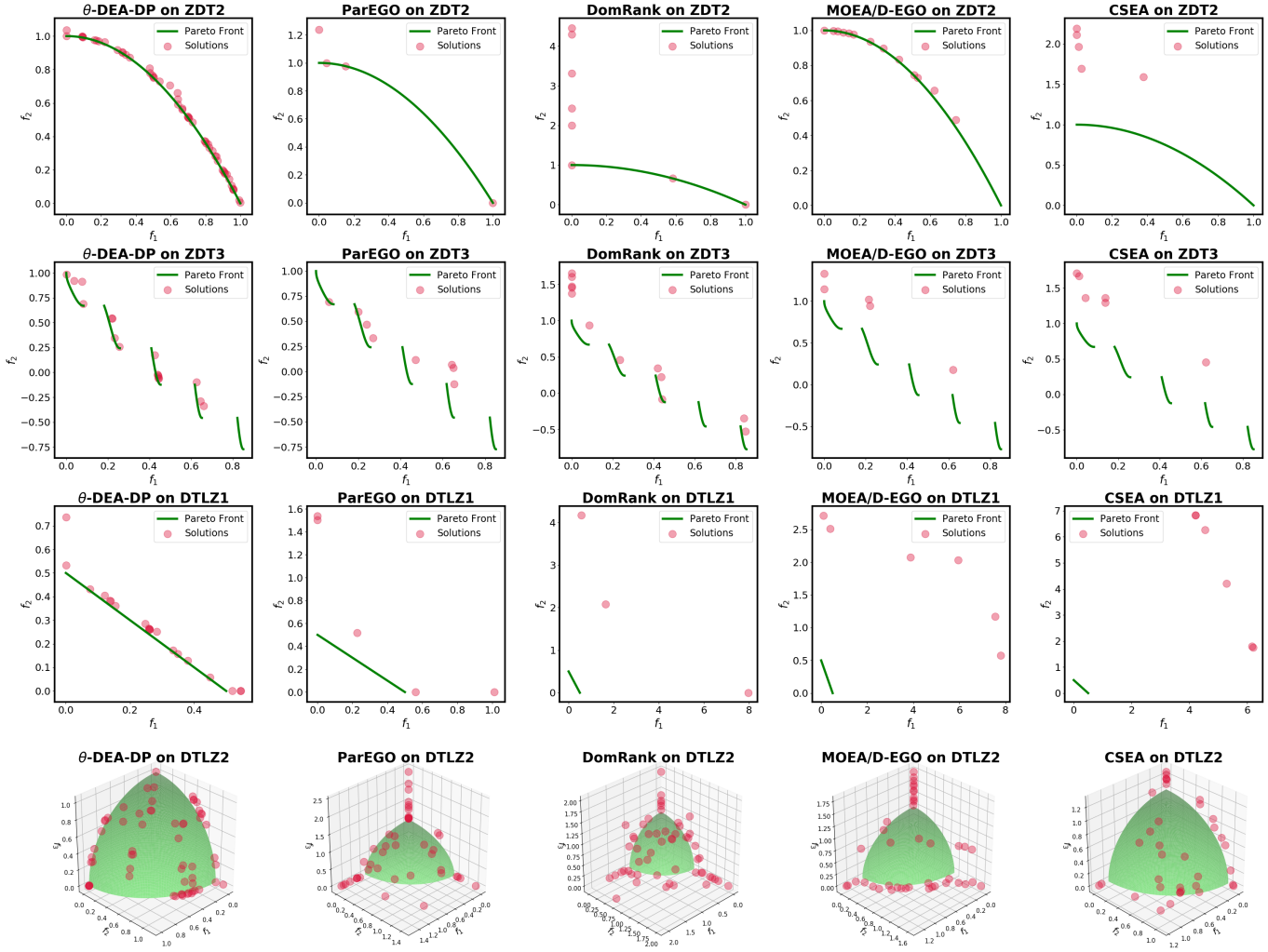


Fig. 4. Final solution sets with the median IGD values in the objective space on ZDT2, ZDT3, DTLZ1 and DTLZ2.

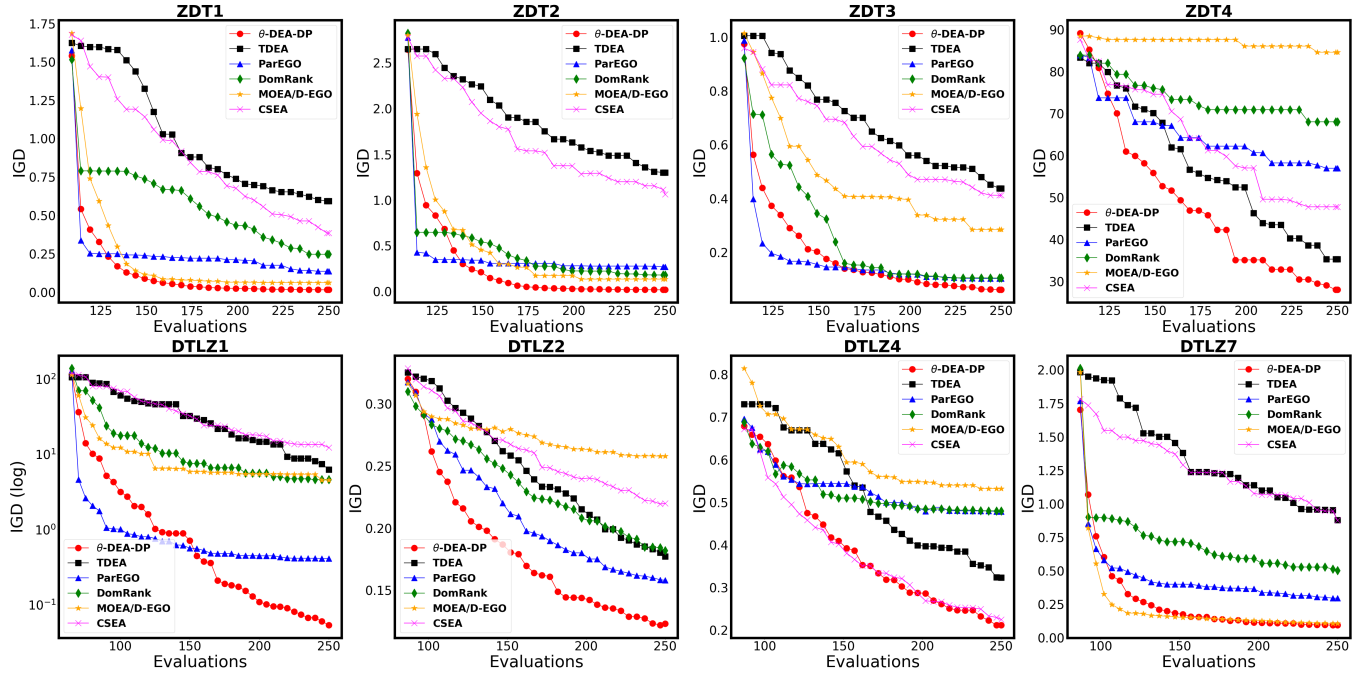


Fig. 5. Evolution of the median of IGD values (over 21 runs) versus the number of function evaluations on ZDT and DTLZ problems.

4, we can derive the following observations:

1) On ZDT2, θ-DEA-DP achieves a very good approximation of the PF; ParEGO, DomRank and MOEA/D-EGO miss a large part of the PF, suffering from the diversity issue; CSEA struggles to converge toward the PF.
2) The PF of ZDT3 has five disconnected parts. All algorithms considered have difficulty in covering them all, but θ-DEA-DP can obtain the best approximation in terms of both convergence and diversity.
3) DTLZ1 is constructed using a multi-modal g function [8], [54], which poses a great difficulty for convergence to the PF. Surprisingly, θ-DEA-DP can still converge to the PF of DTLZ1 very well while maintaining a good spread of solutions. ParEGO can only obtain very few solutions near to the PF. As for DomRank, MOEA/D-EGO and CSEA, their solutions are far away from the PF. Note that ZDT4 is another problem with a multi-modal feature, but its fitness landscape is too rugged, containing 21^9 local PFs. So on ZDT4, although θ-DEA-DP obtains a solution quality higher than all the other algorithms, it indeed fails to approach the PF as indicated by the large IGD values in Table IV.
4) On DTLZ2, all solutions obtained by θ-DEA-DP are close to the PF, whereas some solutions of the other four algorithms are far away from the PF.

TABLE V
BEST, MEDIAN AND WORST IGD VALUES ON FIVE-OBJECTIVE TEST PROBLEMS. BEST PERFORMANCE IS SHOWN IN BOLD

Problem	θ-DEA -DP	θ-DEA	ParEGO	DomRank	MOEA/D -EGO	CSEA
DTLZ1	**5.38E-01**	1.30E+01	2.57E+00	3.57E+00	1.01E+01	3.01E+00
	1.29E+00	3.78E+01	7.51E+00	4.02E+01	5.43E+01	1.54E+01
	6.09E+00	8.02E+01	1.85E+01	1.11E+02	9.93E+01	3.34E+01
		+	+	+	+	+
DTLZ2	**2.46E-01**	3.05E-01	3.59E-01	3.62E-01	3.89E-01	3.45E-01
	2.65E-01	3.41E-01	3.77E-01	4.07E-01	4.41E-01	4.10E-01
	3.01E-01	3.96E-01	4.07E-01	4.48E-01	5.07E-01	4.73E-01
		+	+	+	+	+
DTLZ4	**3.42E-01**	4.43E-01	5.68E-01	6.10E-01	6.32E-01	3.72E-01
	4.45E-01	6.30E-01	6.23E-01	6.64E-01	7.12E-01	**4.10E-01**
	5.57E-01	8.82E-01	7.01E-01	7.62E-01	7.70E-01	**5.48E-01**
		+	+	+	+	≈
DTLZ7	**3.52E-01**	1.05E+00	4.81E-01	5.63E-01	5.19E-01	1.03E+00
	6.88E-01	1.18E+00	**5.37E-01**	8.36E-01	6.47E-01	1.44E+00
	1.01E+00	1.76E+00	**5.73E-01**	1.06E+00	7.65E-01	1.83E+00
		+	−	≈	≈	+
WFG6	**2.63E-01**	2.94E-01	3.13E-01	3.33E-01	2.96E-01	2.83E-01
	2.91E-01	3.11E-01	3.28E-01	3.60E-01	3.13E-01	3.18E-01
	3.15E-01	3.46E-01	3.49E-01	3.80E-01	3.23E-01	3.61E-01
		+	+	+	+	+
WFG7	**2.56E-01**	2.64E-01	2.95E-01	2.82E-01	2.93E-01	2.67E-01
	2.69E-01	2.82E-01	3.37E-01	2.97E-01	3.13E-01	2.96E-01
	3.01E-01	3.17E-01	3.54E-01	3.11E-01	3.57E-01	3.47E-01
		+	+	+	+	+
+/−/≈		6/0/0	5/1/0	5/0/1	5/0/1	5/0/1

Fig. 5 plots the evolutionary trajectories of median IGD values (over 21 runs) with the number of function evaluations for each algorithm on ZDT and DTLZ problems. From Fig. 5 we can see that θ-DEA-DP can always reach a much lower IGD value at a much faster convergence rate than θ-DEA, suggesting that the surrogates in θ-DEA-DP are very powerful. Moreover, as observed from these trajectories, θ-DEA-DP also achieves very competitive performance at any time during the optimization process among all the algorithms considered. This implies that θ-DEA-DP can be successfully applied to many different optimization scenarios where the allowed number of function evaluations varies. Another interesting finding is that ParEGO often converges very fast during earlier stages of optimization (sometimes even faster than θ-DEA-DP), but then almost stagnates. This phenomenon is particularly clear on ZDT2, ZDT3 and DTLZ1, where ParEGO outperforms θ-DEA-DP in the early period but is surpassed by θ-DEA-DP later. We conjecture that ParEGO may be more suitable for optimization scenarios with a much smaller number of function evaluations (e.g., 120).

C. Performance on Many-Objective Optimization Problems

Table V shows the best, median and worst IGD results on five-objective test problems. As can be seen from Table V, θ-DEA-DP still shows overwhelming advantage over all other algorithms compared.

TABLE VI
BEST, MEDIAN AND WORST IGD VALUES ON EIGHT-OBJECTIVE TEST PROBLEMS. BEST PERFORMANCE IS SHOWN IN BOLD

Problem	θ-DEA -DP	θ-DEA	ParEGO	DomRank	MOEA/D -EGO	CSEA
DTLZ1	**3.21E-01**	5.67E-01	6.89E-01	5.95E-01	8.23E-01	4.32E-01
	4.99E-01	2.38E+00	1.37E+00	1.31E+00	6.89E+00	7.52E-01
	1.28E+00	7.43E+00	3.46E+00	3.79E+00	2.56E+01	2.22E+00
		+	+	+	+	+
DTLZ2	**3.90E-01**	4.09E-01	5.15E-01	4.75E-01	4.75E-01	5.20E-01
	4.41E-01	4.51E-01	5.48E-01	5.13E-01	5.12E-01	5.82E-01
	4.91E-01	4.97E-01	5.85E-01	5.52E-01	5.70E-01	6.49E-01
		+	+	+	+	+
DTLZ4	**4.95E-01**	5.80E-01	6.14E-01	6.38E-01	6.47E-01	4.98E-01
	5.35E-01	6.23E-01	6.30E-01	6.55E-01	6.65E-01	5.63E-01
	6.08E-01	6.98E-01	6.57E-01	6.85E-01	7.12E-01	6.59E-01
		+	+	+	+	+
DTLZ7	1.00E+00	9.64E-01	**6.79E-01**	7.14E-01	7.27E-01	1.15E+00
	1.24E+00	1.10E+00	**7.18E-01**	8.77E-01	8.01E-01	1.33E+00
	1.38E+00	1.39E+00	**7.67E-01**	1.23E+00	9.00E-01	1.51E+00
		≈	−	−	−	≈
WFG6	**3.71E-01**	4.05E-01	4.60E-01	4.55E-01	4.22E-01	4.56E-01
	4.14E-01	4.46E-01	4.97E-01	4.83E-01	4.37E-01	5.00E-01
	4.75E-01	4.78E-01	5.30E-01	5.29E-01	4.77E-01	5.41E-01
		+	+	+	+	+
WFG7	**4.47E-01**	4.50E-01	5.31E-01	4.50E-01	4.57E-01	4.72E-01
	4.81E-01	**4.71E-01**	5.56E-01	4.82E-01	5.19E-01	5.54E-01
	5.44E-01	5.21E-01	5.84E-01	**5.16E-01**	5.76E-01	6.18E-01
		≈	+	≈	+	+
+/−/≈		4/0/2	5/1/0	4/1/1	5/1/0	5/0/1

On the five-objective DTLZ4, CSEA obtains a smaller median and worse IGD values than θ-DEA-DP, although there is no statistical difference between the results. Recalling that CSEA also performs comparably to θ-DEA-DP on three-objective DTLZ4, CSEA may be good at handling a biased search space.

On five-objective DTLZ7, θ-DEA-DP is significantly outperformed by ParEGO and is statistically comparable to DomRank and MOEA/D-EGO. The unsatisfactory performance of θ-DEA-DP here may be attributed to the fact that five-objective DTLZ7 has up to 16 disconnected PF regions. So the diversity

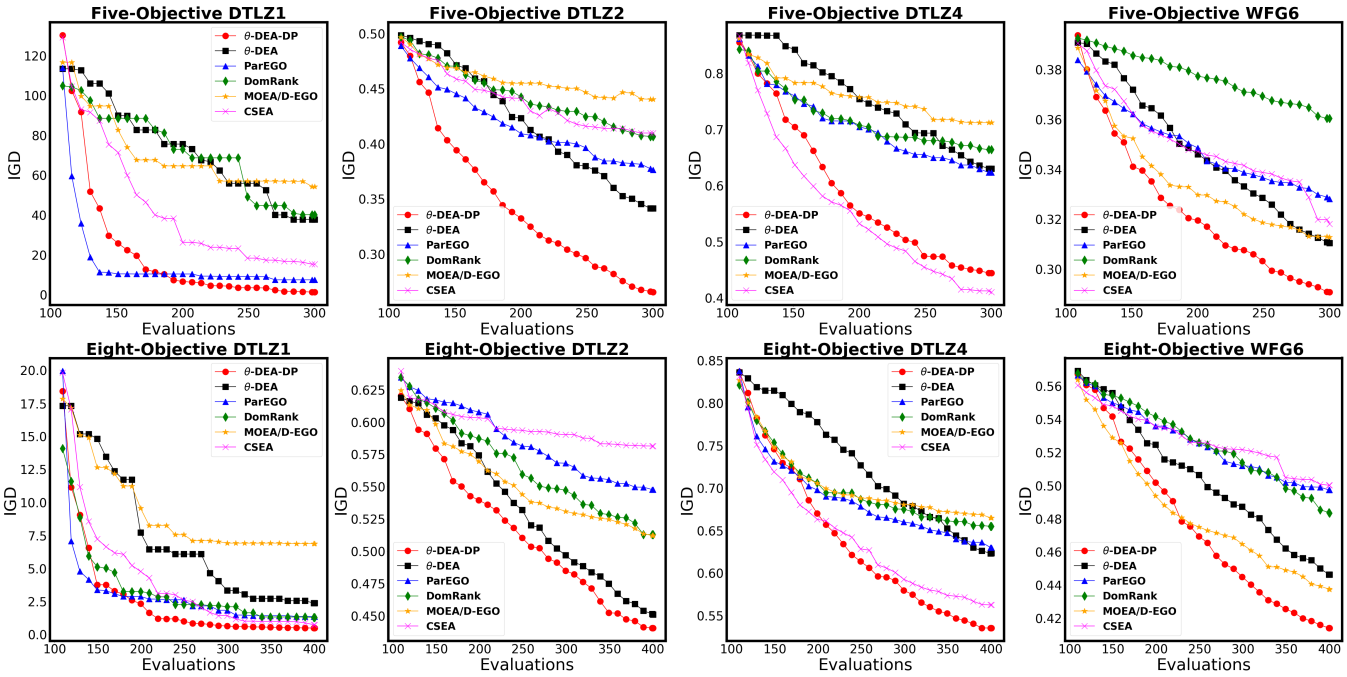


Fig. 6. Evolution of the median of IGD values (over 21 runs) versus the number of function evaluations on five- and eight-objective test problems.

preservation mechanism in θ-DEA-DP, which assumes the PF shape is regular, may not work well. One possible remedy is to adjust the distribution of weight vectors dynamically [59].

Table VI reports the IGD results on eight-objective test problems. Overall, θ-DEA-DP is compared favorably with any of the other algorithms on these problems. Several more detailed observations are as follows:

1) θ-DEA becomes competitive to all surrogate-assisted algorithms including θ-DEA-DP in the eight-objective case. Compared to θ-DEA-DP, θ-DEA performs comparably on DTLZ7 and WFG7. A possible reason is that the higher number of objectives makes it much more difficult to build surrogates with good accuracy, so that the guidance provided by surrogates may become very limited.
2) On eight-objective DTLZ7, θ-DEA-DP becomes less competitive than on five-objective DTLZ7, which is significantly outperformed by three other algorithms. This is not surprising considering that the number of disconnected PF regions of DTLZ7 increases exponentially with the number of objectives.
3) The number of objectives can dramatically influence the competitive relationship between algorithms. For example, θ-DEA-DP is significantly better than CSEA on eight-objective DTLZ4, but is just statistically comparable to CSEA on the three- and five-objective instances. Another more obvious example is the comparison with CSEA on DTLZ1, where it performs worst among all compared algorithms on the two-objective instance but is second only to θ-DEA-DP on the eight-objective instance.

Fig. 6 shows the convergence curves of the median IGD obtained with the number of function evaluations on some five- and eight-objective problems, in order to have a better understanding of the optimization process of each algorithm. It is clear that θ-DEA-DP can usually reduce the IGD values more quickly than the other algorithms.

TABLE VII
THE AVERAGE CPU TIME CONSUMED IN MANAGING SURROGATES

	θ-DEA-DP	ParEGO	DomRank
CPU Time (min.)	37.74	31.36	41.45

In our experiments, θ-DEA-DP, ParEGO and DomRank are all implemented in Python and run in the same computing environment. So we can fairly compare average CPU times over all runs consumed by these three algorithms in managing the surrogates. From Table VII, θ-DEA-DP costs reasonable time for surrogates compared to ParEGO and DomRank. Note that this metric is not very important in practice since function evaluations usually dominate the overall computation time in expensive optimization.

D. Investigations into the Components of θ-DEA-DP

First, we would like to investigate how well deep learning performs on dominance prediction compared to traditional machine learning approaches. To do this, we need to have training sets and test sets. For a multi-objective problem, we use Latin hypercube sampling to generate $11n-1$ solutions, and construct a training set using these solutions as described in Section III-C1. Then we randomly generate 1,000 examples for each class, constituting a test set with size 3,000. We choose support vector machine (SVM) [60], random forest (RF) [61] and complement naive Bayes (CNB) [62] for comparison. CNB is specially designed for imbalanced classifica-

TABLE VIII
COMPARISON OF DIFFERENT MACHINE LEARNING MODELS ON THE DOMINANCE PREDICTION PROBLEM. BEST PERFORMANCE IS SHOWN IN BOLD. STANDARD DEVIATION IS SHOWN IN THE BRACKET

Problem	m	n	Pareto Dominance Prediction Accuracy (%)					θ-Dominance Prediction Accuracy (%)				
			Pareto-Net	Pareto-Net*	SVM	RF	CNB	θ-Net	θ-Net*	SVM	RF	CNB
ZDT1	2	10	**97.37**(0.54)	96.20(0.72)$^+$	81.40(0.63)$^+$	38.33(1.37)$^+$	63.80(2.58)$^+$	**67.77**(1.38)	65.70(1.73)$^+$	66.50(1.65)$^+$	34.23(1.48)$^+$	49.10(2.04)$^+$
ZDT2	2	10	**97.73**(0.37)	97.67(0.70)$^\approx$	85.30(0.87)$^+$	59.73(3.83)$^+$	67.03(1.22)$^+$	79.60(2.23)	**79.90**(2.31)$^\approx$	63.63(1.27)$^+$	64.93(3.99)$^+$	51.43(1.53)$^+$
ZDT3	2	10	**79.93**(1.57)	76.23(1.94)$^+$	73.27(1.23)$^+$	40.63(2.28)$^+$	61.53(1.92)$^+$	**61.53**(1.73)	57.87(1.44)$^+$	59.73(1.26)$^+$	33.37(0.06)$^+$	44.50(1.08)$^+$
DTLZ1	2	6	**76.33**(2.65)	70.90(3.57)$^+$	71.23(2.78)$^+$	38.50(2.36)$^+$	34.57(1.61)$^+$	**70.23**(1.57)	55.60(2.30)$^+$	56.50(1.83)$^+$	33.33(0.00)$^+$	33.50(1.65)$^+$
DTLZ2	3	8	**65.47**(3.88)	44.37(2.87)$^+$	56.80(3.64)$^+$	33.43(0.42)$^+$	43.87(1.49)$^+$	**58.03**(1.40)	46.03(1.34)$^+$	50.13(1.65)$^+$	33.33(0.02)$^+$	36.80(0.81)$^+$
DTLZ4	3	8	**78.87**(2.42)	68.10(2.82)$^+$	75.63(1.67)$^+$	41.83(2.64)$^+$	58.77(2.85)$^+$	53.67(2.29)	54.33(2.06)$^\approx$	54.53(2.40)$^-$	**80.87**(4.33)$^-$	37.30(1.99)$^+$
DTLZ7	3	8	**94.00**(0.78)	89.87(1.79)$^+$	85.30(0.80)$^+$	40.10(2.52)$^+$	63.53(3.88)$^+$	**63.23**(1.54)	55.03(2.30)$^+$	58.87(1.81)$^+$	33.60(0.61)$^+$	46.23(0.72)$^+$
WFG6	3	10	**49.10**(3.06)	37.93(1.39)$^+$	40.87(1.54)$^+$	33.33(0.03)$^+$	46.27(1.56)$^+$	**52.17**(1.62)	46.93(1.53)$^+$	48.20(1.04)$^+$	34.77(3.28)$^+$	40.90(1.58)$^+$
WFG7	3	10	**62.43**(2.75)	55.53(2.62)$^+$	59.00(3.29)$^+$	34.00(0.56)$^+$	51.53(1.70)$^+$	**53.30**(2.06)	48.93(1.98)$^+$	50.13(1.75)$^+$	34.23(0.51)$^+$	46.90(0.92)$^+$

$+$, $-$ and $\approx$ indicate that the result is significantly worse, better or comparable compared to that of Pareto-Net (θ-Net), respectively, according to the Wilcoxon rank sum test at a 5% significance level.

tion. In SVM and RF, we use the same method as in Pareto-Net and θ-Net for alleviating class imbalances. To illustrate the necessity of handling imbalance, we also consider variants of Pareto-Net and θ-Net, denoted respectively as Pareto-Net* and θ-Net*, which use the conventional cross-entropy loss instead of the weighted version. For all the compared learning algorithms, we do not conduct hyperparameter optimization, which is itself an expensive procedure and is usually not desirable for surrogates in practice. So for deep learning algorithms, we just use the hyperparameter setting given in Table III. As for the other learning algorithms, we use the default hyperparameter values provided in scikit-learn [63].

Table VIII shows the median prediction accuracy (over 21 runs) and the standard deviation of different algorithms. Note that in each run the training is conducted over a set of different Latin hypercube samples. As can be seen, Pareto-Net (θ-Net) generally achieves much higher accuracy than SVM, RF and CNB on Pareto (θ) dominance prediction, demonstrating the superiority of deep learning algorithms. Compared to Pareto-Net* (θ-Net*), Pareto-Net (θ-Net) usually performs much better, suggesting that it is necessary to consider the imbalance issue in dominance prediction. Moreover, all the algorithms usually obtain lower accuracy on θ-dominance prediction than on Pareto dominance prediction, indicating that θ-dominance relations may be much harder to learn.

Although the accuracy of surrogates is not the only factor that impacts the performance of surrogate-assisted MOEAs [5], we find some interesting correlations. For example, Pareto-Net and θ-Net have very high accuracy on ZDT2, which can partly explain why θ-DEA-DP achieves a very small IGD value on this problem.

To study the impact of different parameters in θ-DEA-DP, Fig. 7 plots the evolutionary trajectories of median IGD for two-objective DTLZ1 under various parameter settings. Fig. 7 offers the following insights:

1) U, N^* and $Q_{\max}$ need to be sufficiently large. On the other hand, larger values may not always be helpful as it can sometimes harm final performance.
2) Larger D and γ can be beneficial to final performance. But a comprise should be made since computation time for surrogates will be increased thereby.
3) Smaller $T_{\max}$ is usually preferred, which implies that we should focus more on the most recently evaluated solutions when updating the surrogates.
4) The neural networks cannot be too complex, otherwise it may lead to overfitting due to the limitation of data.

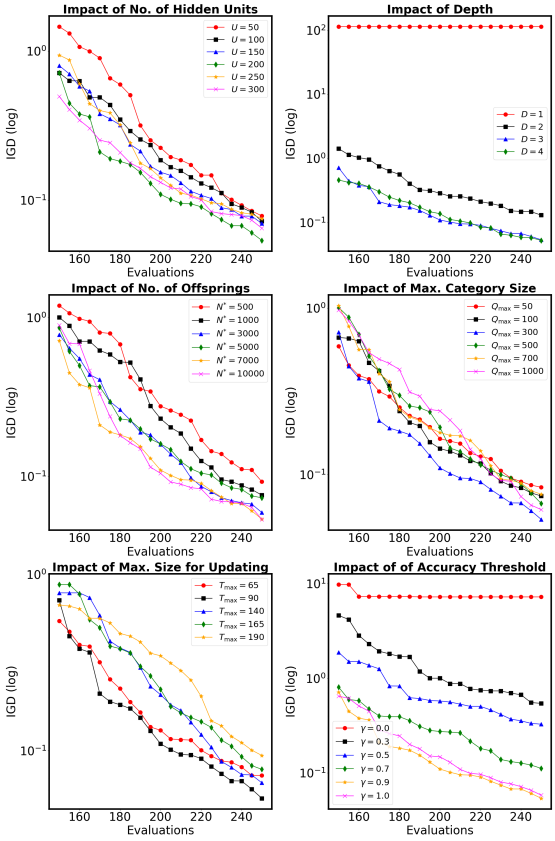


Fig. 7. Evolution of the median of IGD values (over 21 runs) on two-objective DTLZ1 in the last 100 function evaluations, under various parameter settings.

V. CONCLUSION

In this paper, we have described θ-DEA-DP, a surrogate-assisted MOEA for expensive multi-objective optimization. θ-DEA-DP maintains two deep neural networks as surrogates, one for Pareto dominance prediction (i.e., Pareto-Net) and another for θ-dominance prediction (i.e., θ-Net). The two surrogates interact with the evolutionary optimization process of

θ-DEA via a two-stage preselection strategy. More specifically, at each iteration every candidate solution is compared with Pareto and θ-representative solutions for the target cluster, then the surviving candidates are further compared with each other, giving rise to a winner for function evaluation. All these comparisons between solutions are in terms of dominance relations predicted by Pareto-Net or θ-Net, taking into account the confidence of these predictions. With the newly evaluated solution, Pareto-Net and θ-Net are updated in an ad-hoc way according to their current estimated accuracy. Owing to the characteristics of θ-dominance, θ-DEA-DP is expected to provide enough selection pressure even in a high-dimensional objective space, while preserving the desired diversity explicitly through uniformly distributed weight vectors.

Experimental results on a number of multi/many-objective benchmark problems have shown that θ-DEA-DP performs considerably better than several typical surrogate-assisted algorithms. It is noteworthy that θ-DEA-DP has a great advantage in addressing multi-modal multi-objective problems (e.g., DTLZ1). In addition, we demonstrated that deep learning is usually superior to traditional learning algorithms on dominance prediction, and investigated the impact of several key parameters in θ-DEA-DP. Currently, the performance of θ-DEA-DP is not satisfactory on problems with very high numbers of objectives (e.g., 10 or more), which needs to be improved by the further study. In the future, it is also worth exploring the applications of θ-DEA-DP in expensive real-world engineering problems [64], [65].

ACKNOWLEDGEMENT

The authors acknowledge funding from the John R. Koza Chair endowment fund to Michigan State University.

REFERENCES

[1] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, ser. Genetic and Evolutionary Computation Series. Springer, 2007, vol. 5.

[2] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review." in *Proceedings of IEEE Congress on Evolutionary Computation*, 2008, pp. 2419–2426.

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

[4] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: A practical guide*. John Wiley & Sons, 2008.

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

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

[7] 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.

[8] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[9] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2009.

[10] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing convergence and diversity in decomposition-based many-objective optimizers," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 180–198, 2016.

[11] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithmsa comparative case study," in *Proceedings of International conference on Parallel Problem Solving from Nature*, 1998, pp. 292–301.

[12] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, 2008, pp. 784–794.

[13] V. Picheny, "Multiobjective optimization using Gaussian process emulators via stepwise uncertainty reduction," *Statistics and Computing*, vol. 25, no. 6, pp. 1265–1280, 2015.

[14] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser, "On expected-improvement criteria for model-based multi-objective optimization," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, 2010, pp. 718–727.

[15] D. Horn, T. Wagner, D. Biermann, C. Weihs, and B. Bischl, "Model-based multi-objective optimization: taxonomy, multi-point proposal, toolbox and benchmark," in *Proceedings of International Conference on Evolutionary Multi-Criterion Optimization*, 2015, pp. 64–78.

[16] A. A. Rahat, R. M. Everson, and J. E. Fieldsend, "Alternative infill strategies for expensive multi-objective optimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 873–880.

[17] M. Farina, "A neural network based generalized response surface multiobjective evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, 2002, pp. 956–961.

[18] Y. Yun, M. Yoon, and H. Nakayama, "Multi-objective optimization based on meta-modeling by using support vector regression," *Optimization and Engineering*, vol. 10, no. 2, pp. 167–181, 2009.

[19] I. Voutchkov and A. Keane, "Multi-objective optimization using surrogates," in *Computational Intelligence in Optimization*. Springer, 2010, pp. 155–175.

[20] T. Akhtar and C. A. Shoemaker, "Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection," *Journal of Global Optimization*, vol. 64, no. 1, pp. 17–32, 2016.

[21] 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 Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 129–142, 2018.

[22] M. T. Emmerich, K. C. Giannakoglou, and B. Naujoks, "Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 421–439, 2006.

[23] M. K. Karakasis and K. C. Giannakoglou, "On the use of metamodel-assisted, multi-objective evolutionary algorithms," *Engineering Optimization*, vol. 38, no. 8, pp. 941–957, 2006.

[24] A. Habib, H. K. Singh, T. Chugh, T. Ray, and K. Miettinen, "A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive multi/many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 1000–1014, 2019.

[25] J. Knowles and H. Nakayama, "Meta-modeling in multiobjective optimization," in *Multiobjective Optimization*. Springer, 2008, pp. 245–284.

[26] L. V. Santana-Quintero, A. A. Montano, and C. A. C. Coello, "A review of techniques for handling expensive functions in evolutionary multi-objective optimization," in *Computational Intelligence in Expensive Optimization Problems*. Springer, 2010, pp. 29–59.

[27] A. Díaz-Manríquez, G. Toscano, J. H. Barron-Zambrano, and E. Tello-Leal, "A review of surrogate assisted multiobjective evolutionary algorithms," *Computational Intelligence and Neuroscience*, vol. 2016, 2016.

[28] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, vol. 23, no. 9, pp. 3137–3166, 2019.

[29] I. Loshchilov, M. Schoenauer, and M. Sebag, "Dominance-based pareto-surrogate for multi-objective optimization," in *Proceedings of Asia-Pacific Conference on Simulated Evolution and Learning*, 2010, pp. 230–239.

[30] K. Deb, R. Hussein, P. C. Roy, and G. Toscano-Pulido, "A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 104–116, 2019.

[31] I. Loshchilov, M. Schoenauer, and M. Sebag, "A mono surrogate for multiobjective optimization," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 2010, pp. 471–478.

[32] C.-W. Seah, Y.-S. Ong, I. W. Tsang, and S. Jiang, "Pareto rank learning in multi-objective evolutionary algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[33] X. Yu, X. Yao, Y. Wang, L. Zhu, and D. Filev, "Domination-based ordinal regression for expensive multi-objective optimization," in *IEEE Symposium Series on Computational Intelligence*. IEEE, 2019, pp. 2058–2065.

[34] J. Zhang, A. Zhou, K. Tang, and G. Zhang, "Preselection via classification: A case study on evolutionary multiobjective optimization," *Information Sciences*, vol. 465, pp. 388–403, 2018.

[35] 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 Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 74–88, 2019.

[36] G. Guo, W. Li, B. Yang, W. Li, and C. Yin, "Predicting Pareto dominance in multi-objective optimization using pattern recognition," in *Proceedings of the International Conference on Intelligent System Design and Engineering Application*, 2012, pp. 456–459.

[37] S. Bandaru, A. H. Ng, and K. Deb, "On the performance of classification algorithms for learning Pareto-dominance relations," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2014, pp. 1139–1146.

[38] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[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 *Proceedings of International Conference on Machine Learning*, 2015, pp. 2171–2180.

[40] 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 Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[41] Y. Yuan, H. Xu, and B. Wang, "An improved NSGA-III procedure for evolutionary many-objective optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 661–668.

[42] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

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

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

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

[49] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115–148, 1995.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proceedings of Advances in Neural Information Processing Systems*, 2019, pp. 8026–8037.

[51] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[52] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[53] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evolutionary Computation*, vol. 7, no. 3, pp. 205–230, 1999.

[54] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*. Springer, 2005, pp. 105–145.

[55] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[56] R. H. Gómez and C. A. C. Coello, "MOMBI: A new metaheuristic for many-objective optimization based on the r2 indicator," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 2488–2495.

[57] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2008.

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

[59] X. Ma, Y. Yu, X. Li, Y. Qi, and Z. Zhu, "A survey of weight vector adjustment methods for decomposition based multi-objective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 634–649, 2020.

[60] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[61] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[62] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive Bayes text classifiers," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 616–623.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[64] S. J. Daniels, A. A. Rahat, R. M. Everson, G. R. Tabor, and J. E. Fieldsend, "A suite of computationally expensive shape optimisation problems using computational fluid dynamics," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 296–307.

[65] V. Volz, B. Naujoks, P. Kerschke, and T. Tušar, "Single-and multi-objective game-benchmark for evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 647–655.

Yuan Yuan is currently a Postdoctoral Fellow with the Department of Computer Science and Engineering and a member of the BEACON Center for the Study of Evolution in Action at Michigan State University, USA. He received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 2015. From 2014 to 2015, he was a visiting Ph.D. student with the Centre of Excellence for Research in Computational Intelligence and Applications, University of Birmingham, U.K. He worked as a Research Fellow at the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from 2015 to 2016. His research interests include evolutionary computation, machine learning, and search-based software engineering.

Wolfgang Banzhaf is the John R. Koza Chair for Genetic Programming in the Department of Computer Science and Engineering and a member of the BEACON Center for the Study of Evolution in Action at Michigan State University, USA. Previously, he was University Research Professor in the Department of Computer Science of Memorial University of Newfoundland, Canada, where he served as head of department 2003–2009 and 2012–2016. His research interests are in the field of bio-inspired computing, notably evolutionary computation and complex adaptive systems. Studies of self-organization and the field of Artificial Life are also of very much interest to him. Recently he has become more involved with network research as it applies to natural and man-made systems.