Competitive Decomposition-Based Multiobjective Architecture Search for the Dendritic Neural Model

Junkai Ji, Jiajun Zhao, Qiuzhen Lin, *Member, IEEE*, and Kay Chen Tan, *Fellow, IEEE*

***Abstract*—The dendritic neural model (DNM) is computationally faster than other machine-learning techniques, because its architecture can be implemented by using logic circuits and its calculations can be performed entirely in binary form. To further improve the computational speed, a straightforward approach is to generate a more concise architecture for the DNM. Actually, the architecture search is a large-scale multiobjective optimization problem (LSMOP), where a large number of parameters need to be set with the aim of optimizing accuracy and structural complexity simultaneously. However, the issues of irregular Pareto front, objective discontinuity, and population degeneration strongly limit the performances of conventional multiobjective evolutionary algorithms (MOEAs) on the specific problem. Therefore, a novel competitive decomposition-based MOEA is proposed in this study, which decomposes the original problem into several constrained subproblems, with neighboring subproblems sharing overlapping regions in the objective space. The solutions in the overlapping regions participate in environmental selection for the neighboring subproblems and then propagate the selection pressure throughout the entire population. Experimental results demonstrate that the proposed algorithm can possess a more powerful optimization ability than the state-of-the-art MOEAs. Furthermore, both the DNM itself and its hardware implementation can achieve very competitive classification performances when trained by the proposed algorithm, compared with numerous widely used machine-learning approaches.**

***Index Terms*—Architecture search, dendrite, multiobjective optimization, neural network.**

I. Introduction

CURRENTLY, enormous amounts of data are continually being generated at an unprecedented rate. These data involve numerous domains, such as bioinformatics, finance, medicine, security, and engineering sciences [1]. Data with high volume, high dimensionality, high velocity, and high diversity are called "big data," and they require new processing paradigms for optimization, decision making and insight discovery [2]. In recent decades, machine-learning techniques have commonly been used to deal with these massive datasets. Deep learning is undoubtedly one of the most promising paradigms and is attracting increasing attention due to its state-of-the-art performance, particularly in the fields of computer vision, speech recognition, and language processing [3], [4].

Conventional machine-learning techniques need to be implemented in a von Neumann computer architecture. However, the imminent end of the applicability of Moore's law, the high power demand based on Dennard scaling, and the low bandwidth between memory and CPUs have been widely perceived as bottlenecks of von Neumann systems, which will strongly limit the performance of machine-learning techniques on large-scale datasets [5]. The recent concept of neuromorphic computing can be regarded as a potential complementary system for avoiding these issues [6]. Neuromorphic computing refers to brain-inspired devices, models, and computers, which aim to provide faster computation, a smaller footprint, lower power consumption and more powerful fault tolerance than previous systems [7]. Neuromorphic computing is also beneficial for the study of neuroscience since the simulation of realistic neural behaviors is infeasible on traditional computers in terms of the scale, speed, and amount of power consumed [8].

Various neural models have been implemented in neuromorphic systems, including the leaky integrate-and-fire model [9], the FitzHugh-Nagumo model [10], the Izhikevich model [11], and the Hodgkin–Huxley model [12]. Since their signal processes are based on the timing of individual action potentials, these models are referred to as spiking neural networks (SNNs). SNNs can be efficiently implemented on specialized neuromorphic hardware devices, such as memristors, to achieve high power efficiency, as a result of which biological neurons and phenomena can be modeled more accurately [13]. However, SNNs still have limited applicability to complex engineering problems due to the lack of specialized and effective algorithms for either supervised or unsupervised learning [14], [15].

In addition to SNNs, a biologically inspired yet nonspiking model, called the dendritic neural model (DNM), was proposed in our previous studies [16], [17]. Similar to biological neurons, the architecture of the DNM comprises four distinct layers, namely: 1) the synaptic layer; 2) dendritic

Manuscript received May 23, 2021; revised September 22, 2021 and January 16, 2022; accepted March 25, 2022. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant U21A20512, Grant 61876162, Grant 62106151, and Grant 61876110; in part by the Research Grants Council of the Hong Kong under Grant PolyU11202418, Grant PolyU11211521, and Grant PolyU11209219; in part by the Guangdong Basic and Applied Basic Research Fund Project under Grant 2019A1515111139; and in part by the Shenzhen Technology Plan under Grant JCYJ20190808164211203. This article was recommended by Associate Editor Y. S. Ong. *(Corresponding author: Qiuzhen Lin.)*

Junkai Ji, Jiajun Zhao, and Qiuzhen Lin are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: qiuzhlin@szu.edu.cn).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: kctan@polyu.edu.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2022.3165374.

Digital Object Identifier 10.1109/TCYB.2022.3165374

layer; 3) membrane layer; and 4) cell body. The DNM can employ an inherent pruning mechanism to discard redundant synapses and dendritic branches and produce a model architecture specific to each practical application. This model mimics the biological observation that during the early phase of neurogenesis, neurons selectively eliminate synapses, dendritic branches and axons without causing cell death [18]. Further research studies have found that the simplified DNM can be substituted with logic circuits in a neuromorphic system composed of comparators and logical NOT, AND, and OR gates without sacrificing model performance [17], [19]. This hardware implementation enables the DNM to perform calculations completely in binary form. Compared with conventional machine-learning techniques, the DNM has significant advantages in solving large-scale problems with high-speed data streams in terms of calculation speed and power consumption [20], [21]. Accordingly, the DNM has been effectively used in various practical applications, such as biological mechanism analysis [22], [23]; medical diagnosis [24], [25]; financial time-series prediction [26], [27]; and wind speed forecasting [28]. Our ongoing work explores how to further speed up the computation and reduce the power consumption of the DNM. A straightforward and simple approach is to prune a more concise architecture for the DNM. As a result, the neural architecture search of the DNM can be modeled as a large-scale multiobjective optimization problem (LSMOP). Specifically, the dimensions of the search space are determined by the number of encoding parameters of the DNM and always exceed 1000. The target of this problem is to optimize accuracy and structural complexity simultaneously.

Various multiobjective evolutionary algorithms (MOEAs) have been proposed to solve multiobjective optimization problems (MOPs) and can be divided into three categories: 1) Pareto dominance-based algorithms [29]; 2) decomposition-based algorithms [30]; and 3) indicator-based algorithms [31]. Considering that the search space increases exponentially with the number of dimensions, some MOEAs are specifically designed for LSMOPs, such as cooperative coevolving particle swarm optimization [32], the competitive divide-and-conquer algorithm [33], differential grouping2 [34], and other algorithms proposed in [35]–[37]. MOEAs have been widely used in practical applications due to their superior performance. For instance, in [38], the NSGA-III optimization algorithm was modified to solve multiobjective trajectory optimization problems effectively by incorporating a multiple-shooting discretization technique and three different constraint handling methods. In [39], a multiobjective particle swarm optimization algorithm combined with a local gradient search was introduced to solve the multiobjective automatic parking motion planning problem. A two-step strategy was introduced for the real-time trajectory planning of a hypersonic vehicle, where the MOEA was adopted to generate the optimal trajectory for training the deep neural network [40], [41].

However, three distinctive characteristics in the architecture search problem of the DNM significantly limit the performance of these traditional MOEAs. First, the objective of structural complexity is inherently discrete in the decision space. The solutions merely determine a fixed number of values on the basis of this discrete objective, which affects the diversity of the solutions in the population. Second, the Pareto-optimal front of the architecture search is irregular and strongly depends on the particular problem. An irregular Pareto front heavily impacts the effectiveness of decomposition-based MOEAs, as empirically demonstrated in [42]. Third, in the initial optimization phase, while solutions with high model complexity are easily dominated and eliminated, solutions with low complexity quickly take over the entire population. However, according to our empirical evidence, solutions with extremely low model complexity rarely achieve good model performance at the end of optimization. These can be regarded as “invalid” solutions. Population degeneration triggered by invalid solutions occurs for almost all MOEAs and severely degrades both their population diversity and search ability. In fact, these characteristics are found not only in multiobjective neural architecture search problems but also in other machine-learning problems, such as feature selection [43], [44].

To better solve the multiobjective neural architecture search problems in the DNM, a specific competitive decomposition-based MOEA (CDMOEA) is proposed. In the initialization phase of the CDMOEA, the original problem is decomposed into a number of constrained subproblems. Each subproblem corresponds to a distinct objective space. In addition, neighboring subproblems share overlapping regions in the objective space. The solutions in the regions of overlap can participate in the environmental selection of individual neighboring subproblems, propagating the selection pressure through the entire population. By integrating a specific initialization mechanism, repair mechanism and environmental selection mechanism, the CDMOEA can effectively deal with the issues caused by the above characteristics and efficiently find the approximate Pareto-optimal fronts for the DNM architecture search problem. To evaluate the multiobjective optimization performance of the CDMOEA, extensive comparisons are conducted on 20 benchmark classification problems in our experiments.

The contributions of this study can be summarized as follows.

1) A novel CDMOEA is proposed to optimize the multiobjective architecture search problem of the DNM.
2) Three typical characteristics of the problem are considered in the algorithm design of the CDMOEA, namely: a) irregular Pareto fronts; b) the high discontinuity of one objective; and c) the population degeneration triggered by invalid solutions.
3) Extensive experiments are carried out on 20 benchmark datasets. The CDMOEA significantly improves the optimization performance compared with several state-of-the-art MOEAs.
4) After being trained by the CDMOEA, the DNM and its hardware implementation version can achieve very competitive classification accuracy and computational speed

compared with numerous widely used machine-learning approaches.

The remainder of this article is organized as follows. In Section II, the model background is briefly reviewed. In Section III, the equally constrained MOP (ECMOP) and the procedure of the proposed CDMOEA are defined. Next, the experimental designs and results are presented in Section IV, and an analysis is provided in Section V. Finally, conclusions and directions for future work are given in Section VI.

II. Preliminaries

A. Dendritic Neural Model

Inspired by observations of biological neurons, the DNM architecture is composed of four layers, namely: 1) a synaptic layer; 2) a dendritic layer; 3) a membrane layer; and 4) a cell body, as illustrated in Fig. 1. The mathematical definitions of each layer are as follows.

1) *Synaptic Layer:*

$$\mathcal{S}_{i,m} = \frac{1}{1 + e^{-k\left(w_{i,m}x_i - q_{i,m}\right)}}. \tag{1}$$

2) *Dendritic Layer:*

$$\mathcal{Z}_m = \prod_{i=1}^{I} \mathcal{S}_{i,m}. \tag{2}$$

3) *Membrane Layer:*

$$V = \sum_{m=1}^{M} \mathcal{Z}_m. \tag{3}$$

4) *Cell Body (Soma):*

$$O = \frac{1}{1 + e^{-k_{soma}(V - \theta_{soma})}} \tag{4}$$

where x_i represents the ith input signal, $\mathcal{S}_{i,m}$ represents the output of the ith synapse layer on the mth branch of dendrites, $\mathcal{Z}_m$ denotes the output of the mth branch of dendrites, and V and O denote the outputs of the membrane layer and the cell body, respectively. I represents the number of input features, and M represents the number of dendritic branches. k, k_{soma}, and θ_{soma} are positive constants. w and q are the parameters that need to be trained by the learning algorithms.

B. Connection Cases

As introduced above, w and q are the only parameters that determine the final model architecture of the DNM for each specific task. Different values of w and q create four different connection cases for the synapses: 1) a direct connection; 2) an inverse connection; 3) a constant-1 connection; and 4) a constant-0 connection. For the sake of clarity, the symbol of each connection case is presented in Fig. 1. Detailed descriptions of these connection cases are given as follows.

1) *Direct Connection:*

$$C_{\mathcal{D}} = \left\{(w, q) | 0 < q_{i,m} < w_{i,m}\right\}. \tag{5}$$

2) *Inverse Connection:*

$$C_{\mathcal{I}} = \left\{(w, q) | w_{i,m} < q_{i,m} < 0\right\}. \tag{6}$$

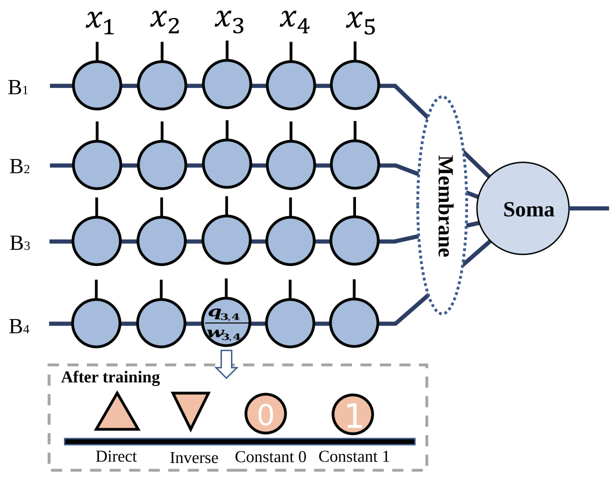


Fig. 1. Model morphology of the DNM and connection cases of the synapses.

3) *Constant-1 Connection:*

$$C_1 = \left\{(w, q) | q_{i,m} < 0 < w_{i,m} \text{ or } q_{i,m} < w_{i,m} < 0\right\}. \tag{7}$$

4) *Constant-0 Connection:*

$$C_0 = \left\{(w, q) | 0 < w_{i,m} < q_{i,m} \text{ or } w_{i,m} < 0 < q_{i,m}\right\}. \tag{8}$$

The different connection cases correspond to different kinds of synaptic outputs. Specifically, the synaptic layer in the constant-0 connection case consistently outputs 0, and the output of the synaptic layer in the constant-1 connection case remains at approximately 1, ignoring the values of the input signals x_i. If x_i exceeds the threshold, then the synapse layer in the direct connection case outputs a value of approximately 0; otherwise, it outputs a value of approximately 1. In contrast, the synapse layer in the inverse connection case outputs a value of approximately 0 if x_i is less than the threshold; otherwise, it outputs a value of approximately 1.

C. Hardware Implementation

The connection cases of the synapses play a crucial role in determining the DNM architecture. Since the multiplication function is employed in the dendritic layer, following the rule that "any value times one is equal to itself," the synapses in the constant-1 connection case do not contribute to the result of the dendritic branch; therefore, this kind of synapse should be deleted. According to the rule that "any value multiplied by 0 yields 0," the synapses in the constant-0 connection case make the output of the corresponding dendritic branches approximately 0; therefore, this kind of dendritic branch and all the synapses on it need to be discarded. As such, the model architecture of the DNM can be simplified by synaptic and dendritic pruning mechanisms.

Notably, only the synapses in the direct and inverse connection cases are left in the final DNM after synaptic and dendritic pruning. Then, the simplified DNM can be transformed by logic circuit classifiers (LCCs), which are composed of comparators and logical AND, OR, and NOT gates, as shown in Fig. 2. Specifically, the synapses in the direct connection case can be replaced with a comparator, while those in the

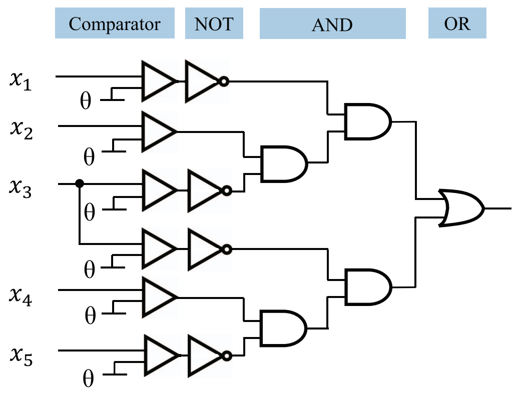


Fig. 2. Structure of an LCC containing comparators and logical AND, OR, and NOT gates.

inverse connection case can be substituted with the combination of a comparator and a logical NOT gate. Each dendritic branch can be replaced with a logical AND gate, and the membrane layer can be replaced with a logical OR gate. In an LCC, each comparator has a threshold $\theta_{i,m}$, which can be calculated by $\theta_{i,m} = q_{i,m}/w_{i,m}$. Unlike other artificial neural networks (ANNs), LCCs perform calculations completely in binary form without any floating-point calculations. Since LCCs are suitable for hardware implementation and large-scale parallel computing, they have an extremely high computational speed, enabling them to deal with big data and high-speed streams in the era of data explosion. For more details regarding the LCC transformation, readers can refer to [17] and [19].

D. Multiobjective Problem Formulation

The mean-squared error (MSE) is employed as the loss function f_{error} to measure how well the trained model fits the training samples, which can be mathematically described by

$$f_{\text{error}}^n = \frac{1}{J}\sum_{j=1}^{J}\left(O_j^n - T_j\right)^2 \tag{9}$$

where J is the sample size of the training dataset, T_j represents the desired target of the jth input signal, and O_j^n represents the jth practical output of the nth neural model. A smaller value of f_{error}^n indicates better training performance of the learning algorithm on the DNM.

The architectural complexity $f_{\text{complexity}}$ serves as the second objective to detect the degree to which the neural model is simplified by the pruning schemes. Mathematically, $f_{\text{complexity}}^n$ can be expressed as follows:

$$f_{\text{complexity}}^n = 1 - \frac{I \cdot \text{Num}_{dp}^n + \text{Num}_{sp}^n}{I * M} \tag{10}$$

where Num_{dp}^n and Num_{sp}^n record the numbers of times that synaptic and dendritic pruning are executed on the nth neural model, respectively.

Accordingly, the neural architecture search of the DNM can be formalized as the following multiobjective optimization algorithm:

$$\begin{aligned} &\text{Minimize} \quad F(x, T, w, q) = \left\{f_{\text{error}}, f_{\text{complexity}}\right\} \\ &\text{subject to} \quad \{w, q\} \in \Omega \end{aligned} \tag{11}$$

where Ω represents the search space of variables in the problem, which is set to $[-10, 10]^D$ in our experiment. The dimension number D of the multiobjective problem is equal to the parameter size of $\{w, q\}$, which can be calculated by $D = 2I * M$, where I represents the feature number and M represents the number of dendritic layers. According to our empirical evidence, setting M to a value larger than $1.5I$ can enable the satisfactory model performance of the DNM [17], [19]. Since N commonly exceeds 1000 and there exists a conflicting relationship between architecture simplicity and model capacity in the DNM [20], the architecture search of the DNM can be modeled as an LSMOP.

III. Methodology

A. Motivation

The primary motivation of this study is to propose an algorithm specifically designed for DNM architecture search problems to enable high computational speed and low power consumption of the hardware implementation in a neuromorphic system. Most neural architectures employed are developed manually by human experts, which is a time-consuming and error-prone process. As such, interest is growing in neural architecture search problems, especially given the remarkable progress in deep ANNs for image and speech recognition tasks [45]. However, the neural architecture search problem of the DNM is inherently different from those of deep learning techniques. In general, the neural architecture search comprises two main parts: 1) architecture optimization and 2) parameter optimization. The architecture optimization part optimizes the pattern of accurate model architectures, and the parameter optimization part learns the parameters in the standard layers. In deep ANNs, these two optimization parts are implemented by separate algorithms [46]–[48]. For instance, the evolutionary algorithm and reinforcement learning are responsible for architecture optimization, and stochastic gradient descent algorithms are applied to train the parameters by viewing each network architecture as a learnable parameter set. Different algorithms are used because optimizing the parameters in deep ANNs is far beyond the search ability of evolutionary algorithms. However, the DNM is aimed at fewer parameters, a simpler architecture and a higher computational speed. Accordingly, MOEAs can be used to optimize the neural architecture and model parameters simultaneously. This is the significant difference between the neural architecture problems of the DNM and deep ANNs.

According to the empirical evidence, Pareto-optimal fronts of the DNM architecture search problem are assumed, as represented by the gray dots in Fig. 3. One objective is continuous, but the other objective is discrete, and its unit length is $[1/(I * M)]$. The solutions with low model complexity are immature, implying that they cannot sufficiently capture the information from the training samples. The solutions with high model complexity are overfitted, suggesting that they can fit the training samples well but suffer from poor generalization performance. Therefore, the primary motivation of this study is to employ multiobjective optimization algorithms to find the most complete Pareto-optimal front and then make

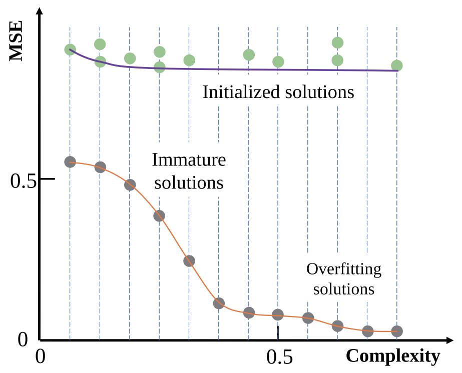


Fig. 3. Solution distribution of the DNM architecture search problem.

decisions to balance the tradeoff between model complexity and generalization performance in the final nondominated solutions.

The conventional MOEA based on decomposition (MOEA/D) with evenly distributed weight vectors does not perform well in solving these multiobjective problems because it is heavily impacted by the Pareto-front shape [42], [43]. Much effort has been made to overcome the disadvantages of the MOEA/D [49], [50]. Still, few studies consider high discontinuity of the second objective or extremely irregular Pareto fronts in the architecture search problems of the DNM. Another obstacle is that in the first stage, all initial solutions are generated randomly and thus have approximate values of f_{error}. Their distribution in the objective space is shown in Fig. 3, where the solutions are represented by light green dots. Solutions with high complexity are easily dominated and then eliminated, while solutions with low model complexity quickly take over the entire population during the optimization process, severely reducing the solution diversity and thus making it difficult for the algorithms to seek the entire Pareto-optimal front. This issue occurs for both Pareto- and decomposition-based MOEAs. More seriously, as described in our previous study [20], oversimplified architectures result in performance degeneration in the LCC. Conventional MOEAs are no longer applicable for solving DNM architecture search problems. As such, an MOEA with a newly designed decomposition mechanism needs to be proposed.

B. Algorithm Framework

To solve the DNM architecture search problem, a specific large-scale MOEA, called the CDMOEA, is introduced. In this section, the framework of the CDMOEA is elucidated, and then the main steps are described. To facilitate a better understanding, we not only detail each step but also provide the reasons for the given designs.

1) Algorithm Overview: The overall framework of the proposed algorithm is presented in Algorithm 1. To find the entire Pareto-optimal front, in line 1, the original problem is first decomposed into K subproblems according to the different ranges of the second objective in the decision space. In line 2, N nearest-neighbor subproblems are selected for each subproblem. Then, the initial population is generated randomly in line 3. Each subproblem is allocated Ps solutions, and the fitness values of all solutions are evaluated. Next, the algorithm enters an evolutionary loop, in which offspring are created for each subproblem. One parent is randomly selected from the kth subproblem S_k. The other two parents are randomly selected from their neighboring subproblems or the entire population, as determined by the hyperparameter δ. Clearly, δ influences the population diversity and balances the tradeoff between exploration and exploitation during the optimization process. In line 12, both differential and polynomial mutation operators are employed to produce the offspring. Then, the model complexity of the offspring is limited to the feasible interval $(0, \psi]$ by the repair mechanism. Finally, the environmental selection is executed to determine which solutions can survive in the next generation in line 15. These procedures are repeated until a termination criterion is met. In our algorithm, the criterion uses a maximum number of evaluations *MaxFES*, as illustrated in Algorithm 1.

Algorithm 1: Algorithm Framework

```
Input: Population size for each subproblem $Ps$, number of
       neighbours $N$, selection probability $\delta$, and maximum
       number of evaluations $MaxFES$;
Output: The final nondominated solution set.
1  /* Decomposition process */
   $S \leftarrow$ Decompose into $K$ subproblems by Algorithm 2;
2  /* Neighbouring subproblem allocation */
   $E \leftarrow$ Find the $N$ nearest neighbours for each subproblem;
3  /* Initialization process */
   $P, F \leftarrow$ Initialize the population by Algorithm 3;
4  while $FES < MaxFES$ do
5      for $k = 1$ to $K$ do
6          $X_1 \leftarrow$ Randomly select one solution from the $k^{th}$
           subproblem $S_k$;
7          if $rand < \delta$ then
8              $X_2, X_3 \leftarrow$ Randomly select two solutions from
               neighbouring subproblems in $E_k$;
9          else
10             $X_2, X_3 \leftarrow$ Randomly select two solutions from the
               entire population;
11         end
12         /* Mutation mechanism */
           $Y \leftarrow$ Employ differential and polynomial mutation
           operators on $X_1$, $X_2$ and $X_3$;
13         $F(Y) \leftarrow$ Evaluate the fitness of $Y$;
14         /* Repair mechanism */
           Repair the offspring solution $Y$ by Algorithm 4;
15         /* Selection mechanism */
           Execute environmental selection by Algorithm 5;
16         $FES \leftarrow FES + 1$;
17     end
18 end
19 return all nondominanted solutions in the final population;
```

2) Competitive Decomposition Mechanism: Compared with other MOPs, there are three unique characteristics of DNM architecture search problems: 1) irregular Pareto fronts; 2) the high discontinuity of one objective; and 3) population degeneration triggered by invalid solutions. A competitive decomposition mechanism that fully takes these characteristics into account is proposed in this section. As illustrated in Fig. 4, the competitive decomposition mechanism divides the decision space with the feasible interval $(0, \psi]$ into several regions.

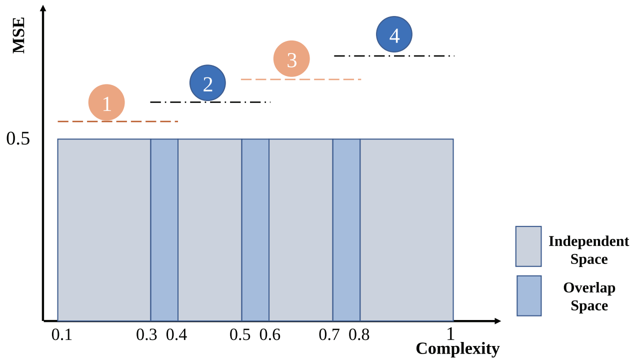


Fig. 4. Illustration of the competitive decomposition mechanism.

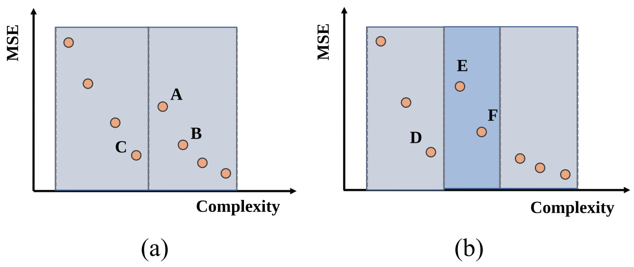


Fig. 5. Comparison of different decomposition mechanisms: (a) decomposition mechanism for a decision space with independent regions and (b) decomposition mechanism for a decision space with regions of overlap.

All the regions have various model architectural boundaries, which implies that each solution in the population is allocated to a different subproblem based on its results for the second objective function. Only the solutions in the same subproblem evolve and compete with each other. This decomposition mechanism has two distinct advantages: first, the search of the subproblem is restricted to a smaller space, which reduces the difficulty of finding the portion of the Pareto-optimal front in this space; second, since the initial solutions with extremely low model complexity are allocated to one subproblem, they cannot dominate other solutions and take over the entire population in a very small number of iterations. Solution diversity is guaranteed during the entire optimization procedure.

In addition, each subproblem has a region of overlap with the nearest neighboring subproblem. For instance, the intervals of the first and second subproblems are [0.1, 0.4] and [0.3, 0.6], respectively; their overlap region is the decision space within the interval [0.3, 0.4]. The intention of the overlap region is to maintain the competitive relationship among the solutions in neighboring subproblems. If there is no overlap region, then the Pareto front may be distributed, as illustrated in Fig. 5 (a). Solutions X_A and X_B in the second subproblem are obviously dominated by solution X_C in the first subproblem. However, if an overlap region exists, then dominated solutions X_E and X_F are eliminated by solution X_D in each iteration. As shown in Fig. 5(b), in the overlap region, only the solutions that are nondominated in both subproblems can survive to the next generation. As such, the solutions in the overlap regions participate in the environmental selection of neighboring subproblems, and the solutions in neighboring subproblems are compared in an indirect way. Then, the selection pressure propagates step by step until it covers the entire population.

Algorithm 2: Decomposition Mechanism

```
Input: Maximum feasible interval of the complexity $\psi$, the
       individual interval $I_{ind}$ and the overlap interval $I_{over}$;
Output: Subproblem number $K$, upper- and lower-bound sets
        $B_U$ and $B_L$.
1  /* Calculate the number of subproblems */
   $K \leftarrow \lceil \frac{\psi - I_{ind}}{I_{ind} - I_{over}} \rceil + 1$;
2  for $k = 1$ to $K$ do
3      /* Calculate the lower bound of the $k$-th subproblem */
       $B_L(k) \leftarrow (k-1)(I_{ind} - I_{over})$;
4      /* Calculate the upper bound of the $k$-th subproblem */
       $B_U(k) \leftarrow k(I_{ind} - I_{over}) + I_{over}$;
5  end
6  $B_U(K) \leftarrow \psi$;
```

Algorithm 3: Initialization Mechanism

```
Input: $K$, $B_U$, $B_L$, $Ps$ and $FES$;
Output: The population $P$ and the fitness $F$,
1  $P \leftarrow$ Initialize the population, where $|P| = K * Ps$;
2  $F \leftarrow$ Evaluate the population, where $F = \{f_e, f_c\}$;
3  $FES \leftarrow |P|$;
4  for $k = 1$ to $K$ do
5      /* Find the solutions of each subproblem */
       $S_k \leftarrow \{X | f_c(X) \in [B_L(k), B_U(k)]\}$;
6      if $|S_k| > Ps$ then
7          /* Discard the redundant solutions randomly */
           $\varnothing \leftarrow S_k(i)$, where $i \in \{1, 2, \cdots, |S_k|\}$;
8      end
9      if $|S_k| < Ps$ then
10         /* Regenerate the insufficient solutions */
           $\hat{X} \leftarrow$ Initialize the solutions, where $|\hat{X}| = Ps - |S_k|$
           and $f_c(\hat{X}) \in [B_L(k), B_U(k)]$;
11         $S_k \leftarrow S_k \cup \hat{X}$;
12         $FES \leftarrow FES + |\hat{X}|$;
13     end
14 end
```

However, all subproblems are optimized independently if there are no overlap regions, and the final Pareto-optimal front is faulty and incomplete. The pseudocode of the competitive decomposition mechanism used in our algorithm is presented in Algorithm 2.

3) Initialization Mechanism: The pseudocode of the initialization mechanism is shown in Algorithm 3. After decomposition into K subproblems, the population, which has a total size of $|P| = K * Ps$, is randomly initialized in line 1. Then, the fitness values of all the solutions F are evaluated in line 2. According to their fitness, each subproblem is guaranteed to possess the same number of solutions. If the subpopulation size exceeds Ps, in line 7, the redundant solutions are randomly selected and discarded; if the subpopulation size is less than Ps, in line 10, the subpopulation is filled up with insufficient solutions that are also randomly generated.

4) Repair Mechanism: In the proposed algorithm, the feasible interval of model complexity in the decision space is $(0, \psi]$. In each iteration, all newly generated solutions are limited to this interval; otherwise, they are repaired. Note that the repair mechanism should follow the principle that the repaired solutions are as close to their original coordinates as possible,

Algorithm 4: Repair Mechanism

```
Input: $Y$, $F(Y)$, $\psi$ and $FES$;
Output: $Y$,
1  while $f_c(Y) = 0$ do
2      if $\exists\{(w_d, q_d)|d = 1, 2, \ldots, I * M\} \in C_0$ then
3          /* Repair the weights of Constant-0 synapses */
           $m \leftarrow$ Index of the dendritic branch, which has the fewest synapses in $C_0$;
4          Regenerate the weights of the $m$-th dendritic branch until $\{w, q\} \in C_{\mathcal{D}} \cup C_{\mathcal{I}}$;
5      else
6          /* Repair the weights of Constant-1 synapses */
           Regenerate the weights of a random synapse in $C_1$ until $\{w, q\} \in C_{\mathcal{D}} \cup C_{\mathcal{I}}$;
7      end
8      $FES \leftarrow FES + 1$;
9  end
10 while $f_c(Y) > \psi$ do
11     /* Repair the weights of direct or inverse synapses */
       Regenerate the weights of a random synapse in $C_{\mathcal{D}} \cup C_{\mathcal{I}}$ until $\{w, q\} \in C_1$;
12     $FES \leftarrow FES + 1$;
13 end
```

Algorithm 5: Environmental Selection

```
Input: $P$, $Y$, $F$, $B_U$ and $B_L$;
Output: $P$.
1  $P \leftarrow P \cup Y$;
2  for $k = 1$ to $K$ do
3      if $f_c(Y) \in [B_L(k), B_U(k)]$ then
4          $S_k \leftarrow$ Solutions of the $k_{th}$ subproblem;
5          /* Select the worst solutions*/
           $X_{worse} \leftarrow$ Dominated or most crowded solutions from $S_k$;
6          /* Abandon the worst solutions*/
           if $\{\exists d | X_{worse} \in S_d \ and \ k \neq d\ \}$ then
7              Remove $X_{worse}$ from $S_k$;
8          else
9              Remove $X_{worse}$ from population $P$;
10         end
11     end
12 end
13 return $P$;
```

which is beneficial for guaranteeing the algorithm's convergence during the optimization process, especially at the end of optimization.

The pseudocode of the repair mechanism is presented in Algorithm 4. The solutions that need to be repaired are divided into two classes. The first class contains the solutions whose model complexity $f_c(Y)$ is equal to 0. These solutions are regarded as invalid candidates and have a high probability of occurring in one generation. For these solutions, if there exists at least one dendritic branch that possesses synapses of the constant-0 connection case, then the branch with the fewest constant-0 synapses is selected, and the weights of these synapses are regenerated until they become direct or inverse connection cases in line 4. Otherwise, all synapses are of the constant-1 connection case, and the weights of a randomly selected synapse are regenerated until they meet the conditions of $C_{\mathcal{D}} \cup C_{\mathcal{I}}$ in line 6. The second class is composed of the solutions whose model complexity $f_c(Y)$ is larger than ψ. For this kind of solution, one synapse of the direct or inverse connection case is chosen randomly, and then its weights are regenerated to meet the condition of C_1 in line 11. The process is repeated until the model complexity $f_c(Y)$ is no more than ψ. Through the above approaches, the weights of all invalid solutions are fine-tuned to satisfy the condition of a feasible interval.

5) Environmental Selection: Environmental selection determines which solutions can survive into the next generation. A greedy strategy is used in the proposed algorithm. The pseudocode of environmental selection is shown in Algorithm 5. The offspring solution Y is merged into the original population, and the subpopulation of each subproblem is updated in line 3. Then, for each subproblem, if solution Y belongs to it, then the worst solution X_{worse} is selected from the subpopulation according to nondominated sorting in line 7. Once all solutions are nondominated, X_{worse} is identified as the most crowded solution by calculating the Euclidean distances between the subpopulations. Next, if the selected solution X_{worse} belongs to another subproblem, then it is merely removed from the solutions in line 9. Otherwise, it is removed from the population directly in line 11. This step guarantees that the solutions in the overlap regions can maintain a competitive relationship with other solutions in the closest subproblems. Finally, the updated population is returned to the main procedure of the algorithm.

6) Computational Complexity Analysis: The computational complexity of the entire algorithm for the architecture search in the DNM is analyzed as follows: the decomposition process costs $\mathcal{O}(K)$, the neighboring subproblem allocation costs $\mathcal{O}(KN)$, and the cost of initializing the population is no higher than $\mathcal{O}(K^2 P_s D)$, where K and P_s represent the subproblem number and population size for each subproblem, respectively. N denotes the number of neighbors, and D is the number of decision variables. Next, the computational complexity of each evolution iteration is presented in detail. For each subproblem, randomly selecting one solution and its neighbors costs $\mathcal{O}(3)$. The costs of the mutation procedure and fitness evaluation are $\mathcal{O}(D)$ and $\mathcal{O}(1)$, respectively. The repair mechanism costs $\mathcal{O}(D)$, and environmental selection costs no larger than $\mathcal{O}(2KP_s^2 + 2K)$ in the worst case. Thus, the computational complexity of each iteration is $\mathcal{O}(K^2 P_s^2 + KD)$. Let P represent the size of the entire population, that is, $P = K \cdot Ps$. The computational complexity is no larger than $\mathcal{O}(P^2 + PD)$, which is acceptable and not time-consuming.

IV. Experiment

A. Experimental Setup

Twenty real-world UCI benchmark datasets from various fields, such as physics, finance, and image analysis, are employed in the experiments [51]. Table I describes all the datasets, namely, the numbers of instances, features and classes. These datasets are available online.[1] The experiments are executed independently 30 times on each dataset. For each run, the samples are randomly split into two parts: 1) a training

[1]https://archive.ics.uci.edu/ml/index.php

TABLE I
DESCRIPTION OF THE BENCHMARK CLASSIFICATION PROBLEMS

Problem	Number of instances	Number of features	Number of classes
Liver	345	6	2
Cancer	683	9	2
Australian	690	14	2
Japanese	690	15	2
Vertebral	310	6	2
Website	1353	9	2
Ionosphere	351	34	2
Blood	748	4	2
WBC	683	9	2
Haberman	306	3	2
Pima	768	8	2
Iris	150	4	3
Wine	178	13	3
Balance	625	4	3
Vehicle	846	18	4
Newthyroid	215	5	3
Postoperative	90	8	3
Ecoli	336	7	8
Zoo	101	16	7
Yeast	1484	8	10

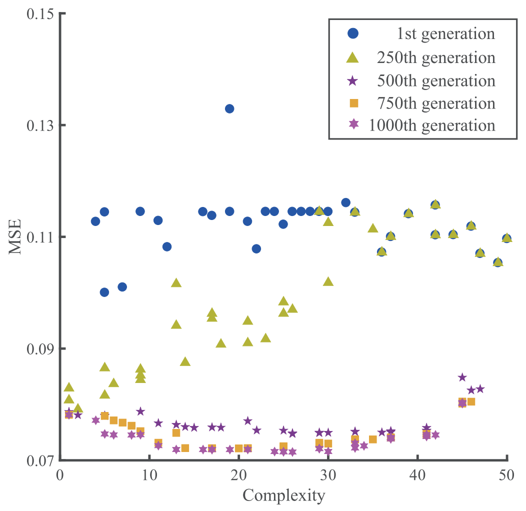


Fig. 6. Solution distributions in the different generations of the CDMOEA on the Blood problem.

dataset (70% of the samples) and 2) a test dataset (the remaining 30% of the samples). The average (*Mean*) and standard deviation (*Std*) of the results are recorded and compared. The code of our proposed algorithm has been made available on the website.[2] All experiments are conducted in MATLAB R2016b on a PC with a 3.20-GHz Intel Core i5-6500 CPU and 8-GB of RAM.

B. Performance Indicators

To investigate the optimization performances of all the MOEAs, two widely used performance indicators, that is: 1) the hypervolume (HV) and 2) the inverted generational distance (IGD), are employed in the experiments [52], [53]. Specifically, the HV indicator measures the volume enclosed by a reference point and the nondominated solutions produced by the algorithm. A larger HV value implies better algorithm performance. Accordingly, a reference point is necessary, and its coordinates are set to $(\psi, 1)$ in our experiments. The IGD indicator calculates the distances between the true Pareto front and nondominated solutions. However, the true Pareto front is usually unknown for DNM architecture search problems in the absence of *a priori* information. Thus, the true Pareto front is approximated by the nondominated solutions obtained from the union of the solutions of all the algorithms. Note that these algorithms need to use the same training dataset in each run. Otherwise, the approximation will be unreasonable. In addition, the accuracy of the test dataset is used as one of the evaluation indicators because it can reflect the generalizability of different classifiers well.

C. Comparisons of Multiobjective Evolutionary Algorithms

In this section, to comprehensively evaluate the proposed algorithm, six state-of-the-art MOEAs, namely: 1) the nondominated sorting genetic algorithm II (NSGA-II) [29]; 2) the MOEA/D [54]; 3) the weighted optimization framework (WOF) [36]; 4) the modified multiobjective particle swarm optimization (MMOPSO) [55]; 5) the large-scale competitive swarm optimizer (LMOCSO) [56]; and 6) the large-scale multiobjective optimization framework (LSMOF) [37], are used as the competitors. Note that NSGA-II acts as an elementary algorithm for the WOF and LSMOF, called WOF-NSGA-II and LSMOF-NSGA-II in the experiments, respectively. For a fair comparison, the maximum fitness evaluation time (*MaxFES*) is used as the termination condition and is set to 5.005E+4, and the population size is set to 50 for all the compared MOEAs unless otherwise specified.

The parameters of all the compared MOEAs are set as in the published papers. We present some hyperparameters of these algorithms here and invite the reader to refer to [29], [36], [37], [55], and [57] for further details. For NSGA-II, WOF-NSGA-II, MMOPSO, and LSMOF-NSGA-II, the crossover probability is set to $p_c = 1.0$, and the mutation probability is set to $p_m = 1/D$, where D is the number of decision variables. The distribution indexes are set to 20. The scale factor F and crossover rate CR are set to 0.5 and 1 in the MOEA/D. In the LMOCSO, the penalty parameter is set to $\alpha = 2$. The control parameters c_1 and c_2 are chosen randomly from [0.1, 0.5], and the inertial weight η is generated randomly from [0.1, 0.5], similar to the case for MMOPSO [55]. In addition, the hyperparameters of the CDMOEA are set as follows: $F = 0.9$, $CR = 0.7$, maximum feasible interval of complexity $\psi = 0.6$, subpopulation size $Ps = 3$, individual interval $I_{ind} = 0.15$ and overlap interval $I_{\text{over}} = 0.075$.

The distributions of the solutions in the different generations of the CDMOEA are illustrated in Fig. 6. The MSEs of the solutions decrease with the optimization phase, and all the solutions can cover the entire search space of the complexity objective evenly in all the generations. In particular, based on the solutions in the 1000th generation, we find that the solutions with either too low or too high model complexity cannot achieve good generalization performance in terms of the MSE, as solutions with too low model complexity imply that the model structure is too small to approximate

[2]http://www.dnm.net.cn/

TABLE II
AVERAGE AND STANDARD DEVIATION OF THE HV OBTAINED BY THE SEVEN COMPARED ALGORITHMS ON 20 BENCHMARK PROBLEMS

	LMOCSO Mean ± Std	WOF-NSGA-II Mean ± Std	LSMOF-NSGA-II Mean ± Std	MMOPSO Mean ± Std	NSGA-II Mean ± Std	MOEA/D Mean ± Std	CDMOEA Mean ± Std
Liver	0.7725 ± 0.0202 (↓)	0.7738 ± 0.0229 (↓)	0.6875 ± 0.2339 (↓)	0.5817 ± 0.3271 (↓)	0.7717 ± 0.0275 (↓)	0.6042 ± 0.2775 (↓)	**0.7842 ± 0.0208**
Cancer	0.9240 ± 0.0221 (↓)	0.9387 ± 0.0207 (↓)	0.9326 ± 0.0217 (↓)	0.5671 ± 0.4397 (↓)	0.9499 ± 0.0178 (↓)	0.4479 ± 0.4278 (↓)	**0.9672 ± 0.0085**
Australian	0.8396 ± 0.1630 (↓)	0.8798 ± 0.0261 (↓)	0.8285 ± 0.1609 (↓)	0.4278 ± 0.4363 (↓)	0.8604 ± 0.1637 (↓)	0.5409 ± 0.4204 (↓)	**0.8949 ± 0.0119**
Japanese	0.8477 ± 0.0661 (↓)	0.7852 ± 0.2679 (↓)	0.7978 ± 0.2211 (↓)	0.3583 ± 0.4190 (↓)	0.8841 ± 0.0277 (↓)	0.4446 ± 0.4246 (↓)	**0.8964 ± 0.0101**
Vertebral	0.8731 ± 0.0191 (↓)	0.8803 ± 0.0229 (↓)	0.8713 ± 0.0249 (↓)	0.7267 ± 0.3315 (↓)	0.8854 ± 0.0154 (○)	0.6541 ± 0.3357 (↓)	**0.8885 ± 0.0175**
Website	0.9020 ± 0.0214 (↓)	0.9039 ± 0.0116 (↓)	0.8959 ± 0.0210 (↓)	0.7925 ± 0.2706 (↓)	0.8789 ± 0.1663 (↓)	0.6442 ± 0.3959 (↓)	**0.9141 ± 0.0087**
Ionosphere	0.2323 ± 0.1221 (↓)	0.7396 ± 0.2565 (↓)	0.3569 ± 0.3198 (↓)	0.4516 ± 0.3215 (↓)	0.2634 ± 0.3829 (↓)	0.7077 ± 0.2899 (↓)	**0.9243 ± 0.0185**
Blood	0.8433 ± 0.0137 (○)	0.8155 ± 0.1546 (○)	0.8421 ± 0.0145 (↓)	0.8085 ± 0.1539 (↓)	0.8440 ± 0.0122 (○)	0.6074 ± 0.3433 (↓)	**0.8447 ± 0.0147**
WBC	0.9317 ± 0.0277 (↓)	0.9454 ± 0.0198 (↓)	0.9433 ± 0.0215 (↓)	0.7316 ± 0.3727 (↓)	0.9552 ± 0.0141 (↓)	0.3781 ± 0.4404 (↓)	**0.9699 ± 0.0093**
Haberman	0.7950 ± 0.0294 (↓)	0.8066 ± 0.0269 (○)	**0.8096 ± 0.0271 (○)**	0.8062 ± 0.0258 (○)	0.8043 ± 0.0264 (○)	0.7118 ± 0.1960 (↓)	0.8038 ± 0.0252
Pima	0.8168 ± 0.0155 (↓)	0.8205 ± 0.0216 (↓)	0.8157 ± 0.0223 (↓)	0.6671 ± 0.3047 (↓)	0.8192 ± 0.0207 (↓)	0.5877 ± 0.3314 (↓)	**0.8323 ± 0.0137**
Iris	0.9945 ± 0.0051 (↓)	0.9955 ± 0.0037 (↓)	0.9962 ± 0.0004 (↓)	0.8923 ± 0.3026 (↓)	0.9962 ± 0.0004 (↓)	0.0000 ± 0.0000 (↓)	**0.9962 ± 0.0004**
Wine	0.8939 ± 0.1720 (↓)	0.9252 ± 0.1784 (↓)	0.9398 ± 0.0404 (↓)	0.2993 ± 0.4326 (↓)	0.9578 ± 0.0353 (↓)	0.1686 ± 0.3445 (↓)	**0.9811 ± 0.0106**
Balance	0.6927 ± 0.3524 (↓)	0.4339 ± 0.4414 (↓)	0.7831 ± 0.2657 (↓)	0.1453 ± 0.3305 (↓)	0.4635 ± 0.4411 (↓)	0.4008 ± 0.4361 (↓)	**0.9301 ± 0.0106**
Vehicle	0.4493 ± 0.4004 (↓)	0.6923 ± 0.2781 (↓)	0.7502 ± 0.2052 (↓)	0.3432 ± 0.3993 (↓)	0.7802 ± 0.1494 (↓)	0.1675 ± 0.3096 (↓)	**0.8376 ± 0.0130**
Newthyroid	0.9031 ± 0.0286 (↓)	0.9231 ± 0.0243 (↓)	0.8946 ± 0.1702 (↓)	0.8355 ± 0.2299 (↓)	0.9253 ± 0.0215 (↓)	0.5670 ± 0.4093 (↓)	**0.9537 ± 0.0156**
Postoperative	0.6888 ± 0.1385 (↓)	0.7266 ± 0.0420 (↓)	0.7027 ± 0.1411 (↓)	0.4443 ± 0.3720 (↓)	0.7374 ± 0.0578 (↓)	0.5493 ± 0.2573 (↓)	**0.7610 ± 0.0413**
Ecoli	0.8982 ± 0.1710 (↓)	0.9486 ± 0.0182 (↓)	0.9457 ± 0.0144 (↓)	0.6360 ± 0.4244 (↓)	0.9473 ± 0.0162 (↓)	0.6346 ± 0.3914 (↓)	**0.9637 ± 0.0125**
Zoo	0.8694 ± 0.2965 (↓)	0.9826 ± 0.0183 (↓)	0.9654 ± 0.0310 (↓)	0.5080 ± 0.4838 (↓)	0.9815 ± 0.0192 (↓)	0.5075 ± 0.4836 (↓)	**0.9960 ± 0.0020**
Yeast	0.7956 ± 0.2176 (↓)	0.8405 ± 0.1606 (↓)	0.8487 ± 0.1625 (↓)	0.5986 ± 0.3994 (↓)	0.8431 ± 0.1609 (↓)	0.5100 ± 0.3961 (↓)	**0.9023 ± 0.0092**
↑ / ↓ / ○	0 / 19 / 1	0 / 18 / 2	0 / 19 / 1	0 / 19 / 1	0 / 17 / 3	0 / 20 / 0	-

TABLE III
AVERAGE AND STANDARD DEVIATION OF THE IGD OBTAINED BY THE SEVEN COMPARED ALGORITHMS ON 20 BENCHMARK PROBLEMS

	LMOCSO Mean ± Std	WOF-NSGA-II Mean ± Std	LSMOF-NSGA-II Mean ± Std	MMOPSO Mean ± Std	NSGA-II Mean ± Std	MOEA/D Mean ± Std	CDMOEA Mean ± Std
Liver	0.0469 ± 0.0493 (○)	0.0444 ± 0.0505 (○)	0.1883 ± 0.4189 (↓)	0.3742 ± 0.5853 (↓)	0.0446 ± 0.0485 (○)	0.3018 ± 0.5089 (↓)	**0.0279 ± 0.0175**
Cancer	0.0614 ± 0.0589 (↓)	0.0461 ± 0.0619 (↓)	0.0367 ± 0.0524 (○)	0.5518 ± 0.6677 (↓)	0.0332 ± 0.0611 (○)	0.7212 ± 0.6613 (↓)	**0.0230 ± 0.0202**
Australian	0.1298 ± 0.2639 (↓)	0.0712 ± 0.1020 (↓)	0.1352 ± 0.2627 (↓)	0.7420 ± 0.6854 (↓)	0.1068 ± 0.2690 (↓)	0.5540 ± 0.6679 (↓)	**0.0119 ± 0.0124**
Japanese	0.1165 ± 0.1020 (↓)	0.2172 ± 0.4191 (↓)	0.1890 ± 0.3478 (↓)	0.8585 ± 0.6510 (↓)	0.0805 ± 0.1092 (↓)	0.6963 ± 0.6841 (↓)	**0.0072 ± 0.0092**
Vertebral	0.0609 ± 0.0586 (↓)	0.0455 ± 0.0505 (↓)	0.0572 ± 0.0561 (↓)	0.2750 ± 0.5196 (↓)	0.0384 ± 0.0545 (○)	0.3407 ± 0.5477 (↓)	**0.0210 ± 0.0126**
Website	0.0650 ± 0.0935 (↓)	0.0610 ± 0.0908 (↓)	0.0642 ± 0.0911 (↓)	0.2184 ± 0.4154 (↓)	0.0983 ± 0.2639 (○)	0.4191 ± 0.6134 (↓)	**0.0146 ± 0.0123**
Ionosphere	1.4142 ± 0.0000 (↓)	0.2320 ± 0.4052 (↓)	0.7375 ± 0.6039 (↓)	1.4142 ± 0.0000 (↓)	0.9876 ± 0.6158 (↓)	0.2629 ± 0.4631 (↓)	**0.0045 ± 0.0085**
Blood	0.0972 ± 0.1420 (↓)	0.1320 ± 0.2773 (↓)	0.0904 ± 0.1375 (↓)	0.1381 ± 0.2785 (↓)	0.0867 ± 0.1408 (↓)	0.4038 ± 0.5778 (↓)	**0.0122 ± 0.0091**
WBC	0.0463 ± 0.0269 (↓)	0.0273 ± 0.0216 (○)	0.0269 ± 0.0208 (○)	0.3256 ± 0.5540 (↓)	**0.0234 ± 0.0195 (↑)**	0.8367 ± 0.6719 (↓)	0.0289 ± 0.0200
Haberman	0.0302 ± 0.0222 (○)	0.0221 ± 0.0167 (○)	**0.0189 ± 0.0207 (↑)**	0.0215 ± 0.0197 (○)	0.0203 ± 0.0186 (↑)	0.1415 ± 0.3470 (↓)	0.0250 ± 0.0206
Pima	0.0621 ± 0.0769 (↓)	0.0519 ± 0.0786 (↓)	0.0555 ± 0.0711 (↓)	0.2813 ± 0.5177 (↓)	0.0436 ± 0.0647 (↓)	0.3918 ± 0.5777 (↓)	**0.0152 ± 0.0131**
Iris	0.0885 ± 0.0493 (↓)	0.0475 ± 0.0481 (↓)	0.0494 ± 0.0454 (↓)	0.2016 ± 0.4136 (↓)	0.0493 ± 0.0467 (↓)	1.4142 ± 0.0000 (↓)	**0.0118 ± 0.0158**
Wine	0.1571 ± 0.2516 (↓)	0.1251 ± 0.2584 (↓)	0.1014 ± 0.0883 (↓)	0.9770 ± 0.6311 (↓)	0.0708 ± 0.0940 (↓)	1.1566 ± 0.5247 (↓)	**0.0199 ± 0.0227**
Balance	0.3407 ± 0.5481 (↓)	0.7316 ± 0.6945 (↓)	0.1989 ± 0.4150 (↓)	1.1887 ± 0.5131 (↓)	0.7016 ± 0.6796 (↓)	0.7848 ± 0.6847 (↓)	**0.0046 ± 0.0093**
Vehicle	0.6539 ± 0.6775 (↓)	0.2483 ± 0.4679 (↓)	0.1476 ± 0.3482 (↓)	0.8311 ± 0.6794 (↓)	0.1103 ± 0.2523 (↓)	1.1133 ± 0.5549 (↓)	**0.0132 ± 0.0100**
Newthyroid	0.1007 ± 0.0763 (↓)	0.0749 ± 0.0719 (↓)	0.1258 ± 0.2558 (↓)	0.1862 ± 0.3421 (↓)	0.0805 ± 0.0800 (↓)	0.5404 ± 0.6312 (↓)	**0.0216 ± 0.0177**
Postoperative	0.1523 ± 0.2660 (↓)	0.1014 ± 0.1235 (↓)	0.1437 ± 0.2706 (↓)	0.5985 ± 0.6802 (↓)	0.0845 ± 0.1312 (○)	0.3576 ± 0.4934 (↓)	**0.0339 ± 0.0268**
Ecoli	0.1011 ± 0.2530 (↓)	0.0399 ± 0.0480 (↓)	0.0390 ± 0.0490 (↓)	0.4602 ± 0.6358 (↓)	0.0377 ± 0.0454 (↓)	0.4385 ± 0.6000 (↓)	**0.0173 ± 0.0148**
Zoo	0.2088 ± 0.4136 (↓)	0.0577 ± 0.0803 (↓)	0.0668 ± 0.0878 (↓)	0.6914 ± 0.6882 (↓)	0.0638 ± 0.0925 (↓)	0.7023 ± 0.6798 (↓)	**0.0052 ± 0.0098**
Yeast	0.1463 ± 0.3468 (↓)	0.0891 ± 0.2533 (↓)	0.0875 ± 0.2535 (↓)	0.4608 ± 0.6361 (↓)	0.0868 ± 0.2544 (↓)	0.5725 ± 0.6527 (↓)	**0.0168 ± 0.0183**
↑ / ↓ / ○	0 / 18 / 2	0 / 17 / 3	1 / 17 / 2	0 / 19 / 1	2 / 13 / 5	0 / 20 / 0	-

the complex nonlinear relationships between the inputs and targets, while solutions with too high model complexity indicate that the structure is too large and results in overfitting. The CDMOEA can provide a set of solutions that balance the accuracy and model complexity well.

The experimental results of the algorithms regarding the HV and IGD are compared in Tables II and III, respectively. For the HV, the CDMOEA obtains the best performance on 19 (of 20) benchmark problems. The exception is that LSMOF-NSGA-II performs best on the Haberman dataset. For the HV, the CDMOEA is superior to all the other algorithms on 18 of 20 problems. The exceptions are the WBC and Haberman datasets, on which LSMOF-NSGA-II and NSGA-II, respectively, perform the best. In addition, a Wilcoxon rank-sum test [58] is employed to detect significant differences between the CDMOEA and its competitors. The significance level is set to 0.1, and the symbols "↑," "↓," and "○" denote that the competitor is significantly better than, significantly worse than, and statistically equivalent to the proposed CDMOEA. The CDMOEA clearly performs significantly better than the six state-of-the-art MOEAs on most benchmark datasets in

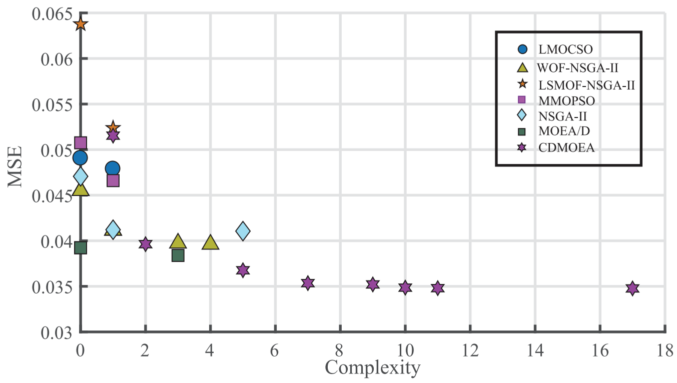


Fig. 7. Pareto fronts of the compared MOEAs on the website problem.

terms of both the HV and IGD. This finding suggests that the Pareto fronts optimized by the CDMOEA are more diverse than those optimized by the other MOEAs. The obtained Pareto fronts of all the algorithms on the Website dataset are illustrated in Fig. 7. Clearly, solutions with relatively high architectural complexity are missing for the other MOEAs;

TABLE IV
AVERAGE AND STANDARD DEVIATION OF THE DNM ACCURACY OBTAINED BY THE SEVEN COMPARED ALGORITHMS ON 20 BENCHMARK PROBLEMS

	LMOCSO	WOF-NSGA-II	LSMOF-NSGA-II	MMOPSO	NSGA-II	MOEA/D	CDMOEA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
Liver	0.6763 ± 0.0583 (○)	0.6487 ± 0.0426 (○)	0.6397 ± 0.0564 (↓)	0.6288 ± 0.0619 (↓)	0.6686 ± 0.0673 (↓)	0.6891 ± 0.0538 (○)	**0.6913** ± **0.0498**
Cancer	0.9557 ± 0.0149 (○)	0.9586 ± 0.0121 (↓)	0.9579 ± 0.0117 (○)	0.9576 ± 0.0143 (○)	0.9525 ± 0.0135 (○)	**0.9592** ± **0.0109** (○)	0.9567 ± 0.0141
Australian	0.8229 ± 0.0701 (○)	0.8337 ± 0.0488 (○)	0.8018 ± 0.0925 (↓)	0.7657 ± 0.1033 (↓)	0.8478 ± 0.0525 (○)	**0.8528** ± **0.0309** (↑)	0.8444 ± 0.0245
Japanese	0.7839 ± 0.1201 (↓)	0.8303 ± 0.0565 (○)	0.7874 ± 0.1049 (↓)	0.7717 ± 0.1051 (↓)	0.8296 ± 0.0511 (↓)	0.8438 ± 0.0313 (↓)	**0.8536** ± **0.0238**
Vertebral	0.8190 ± 0.0367 (○)	0.8251 ± 0.0414 (↓)	0.8050 ± 0.0405 (↓)	0.8043 ± 0.0322 (↓)	0.8276 ± 0.0385 (○)	0.8276 ± 0.0400 (○)	**0.8341** ± **0.0349**
Website	0.8709 ± 0.0285 (↓)	0.8733 ± 0.0338 (↓)	0.8457 ± 0.0598 (↓)	0.8392 ± 0.0497 (↓)	0.8768 ± 0.0197 (↓)	0.8781 ± 0.0216 (↓)	**0.8874** ± **0.0159**
Ionosphere	0.3502 ± 0.0373 (↓)	0.8048 ± 0.0467 (↓)	0.7486 ± 0.0398 (↓)	0.3546 ± 0.0566 (↓)	0.4784 ± 0.2147 (↓)	0.7927 ± 0.1986 (↓)	**0.9041** ± **0.0292**
Blood	**0.7874** ± **0.0220** (↑)	0.7784 ± 0.0204 (↓)	0.7699 ± 0.0222 (○)	0.7757 ± 0.0249 (○)	0.7704 ± 0.0293 (○)	0.7833 ± 0.0236 (○)	0.7760 ± 0.0265
WBC	0.9569 ± 0.0172 (○)	0.9587 ± 0.0128 (○)	**0.9618** ± **0.0117** (○)	0.9566 ± 0.0110 (○)	0.9603 ± 0.0137 (○)	0.9600 ± 0.0120 (○)	0.9589 ± 0.0160
Haberman	**0.7489** ± **0.0425** (↑)	0.7196 ± 0.0328 (○)	0.7293 ± 0.0336 (○)	0.7409 ± 0.0437 (↑)	0.7348 ± 0.0351 (○)	0.7315 ± 0.0420 (○)	0.7239 ± 0.0367
Pima	0.7307 ± 0.0353 (↓)	0.7432 ± 0.0332 (○)	0.7426 ± 0.0230 (○)	0.7228 ± 0.0371 (↓)	**0.7551** ± **0.0279** (○)	0.7486 ± 0.0309 (○)	0.7451 ± 0.0349
Iris	**1.0000** ± **0.0000** (○)	0.9993 ± 0.0041 (○)	**1.0000** ± **0.0000** (○)	0.9993 ± 0.0041 (○)	**1.0000** ± **0.0000** (○)	0.9993 ± 0.0041 (○)	**1.0000** ± **0.0000**
Wine	0.9384 ± 0.1056 (↓)	0.9723 ± 0.0220 (○)	0.9447 ± 0.0411 (↓)	0.9503 ± 0.0518 (↓)	**0.9774** ± **0.0245** (○)	0.9761 ± 0.0171 (○)	0.9730 ± 0.0251
Balance	0.9085 ± 0.0189 (○)	**0.9199** ± **0.0165** (↑)	0.9135 ± 0.0188 (○)	0.9147 ± 0.0191 (○)	0.9099 ± 0.0193 (○)	0.9126 ± 0.0194 (○)	0.9106 ± 0.0150
Vehicle	0.7518 ± 0.0179 (↓)	0.7556 ± 0.0329 (○)	0.7575 ± 0.0257 (○)	0.7533 ± 0.0265 (↓)	0.7571 ± 0.0243 (↓)	0.7612 ± 0.0212 (○)	**0.7667** ± **0.0222**
Newthyroid	0.9246 ± 0.0355 (○)	0.9210 ± 0.0258 (↓)	0.9123 ± 0.0277 (↓)	0.9077 ± 0.0361 (↓)	0.9287 ± 0.0301 (○)	0.9303 ± 0.0302 (○)	**0.9323** ± **0.0423**
Postoperative	0.6526 ± 0.1062 (○)	0.6641 ± 0.0886 (○)	**0.6949** ± **0.0953** (↑)	0.6923 ± 0.0700 (↑)	0.6397 ± 0.0917 (○)	0.6346 ± 0.0908 (○)	0.6423 ± 0.0736
Ecoli	0.9482 ± 0.0250 (↓)	0.9607 ± 0.0177 (○)	0.9601 ± 0.0212 (○)	0.9383 ± 0.0423 (↓)	0.9594 ± 0.0186 (○)	0.9574 ± 0.0218 (○)	**0.9630** ± **0.0180**
Zoo	0.9300 ± 0.1320 (↓)	0.9811 ± 0.0324 (○)	0.9722 ± 0.0392 (↓)	0.8744 ± 0.1653 (↓)	0.9889 ± 0.0220 (↓)	0.9789 ± 0.0297 (↓)	**0.9978** ± **0.0085**
Yeast	0.8709 ± 0.0211 (↓)	0.8757 ± 0.0192 (↓)	0.8731 ± 0.0221 (○)	0.8641 ± 0.0226 (↓)	0.8749 ± 0.0144 (↓)	**0.8809** ± **0.0163** (○)	0.8796 ± 0.0150
↑/↓/○	2/9/9	1/7/12	1/9/10	2/13/5	0/7/13	1/4/15	-

TABLE V
AVERAGE AND STANDARD DEVIATION OF THE LCC ACCURACY OBTAINED BY THE SEVEN COMPARED ALGORITHMS ON 20 BENCHMARK PROBLEMS

	LMOCSO	WOF-NSGA-II	LSMOF-NSGA-II	MMOPSO	NSGA-II	MOEA/D	CDMOEA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
Liver	0.5840 ± 0.0584 (↓)	0.5994 ± 0.0500 (↓)	0.5910 ± 0.0375 (↓)	0.6045 ± 0.0471 (↓)	0.6247 ± 0.0761 (○)	0.5712 ± 0.0397 (↓)	**0.6455** ± **0.0525**
Cancer	0.8203 ± 0.2102 (↓)	0.9398 ± 0.0453 (↓)	0.9457 ± 0.0322 (○)	0.7703 ± 0.2823 (↓)	0.9448 ± 0.0215 (↓)	0.5968 ± 0.2887 (↓)	**0.9546** ± **0.0157**
Australian	0.7660 ± 0.1464 (↓)	0.7971 ± 0.1191 (○)	0.7673 ± 0.1304 (↓)	0.6527 ± 0.1251 (↓)	0.8195 ± 0.1123 (○)	0.6697 ± 0.1377 (↓)	**0.8409** ± **0.0280**
Japanese	0.7251 ± 0.1678 (↓)	0.8098 ± 0.0958 (↓)	0.7626 ± 0.1395 (↓)	0.6498 ± 0.1279 (↓)	0.8150 ± 0.0979 (↓)	0.6646 ± 0.1433 (↓)	**0.8541** ± **0.0208**
Vertebral	0.7498 ± 0.0810 (↓)	0.7867 ± 0.0410 (↓)	0.7685 ± 0.0687 (↓)	0.6925 ± 0.1957 (↓)	0.7950 ± 0.0396 (○)	0.6047 ± 0.1980 (↓)	**0.8057** ± **0.0384**
Website	0.8030 ± 0.0799 (↓)	0.8407 ± 0.0642 (↓)	0.8198 ± 0.0880 (↓)	0.7992 ± 0.1050 (↓)	0.8584 ± 0.0166 (↓)	0.6927 ± 0.1388 (↓)	**0.8674** ± **0.0145**
Ionosphere	0.3502 ± 0.0373 (↓)	0.7368 ± 0.1216 (↓)	0.5857 ± 0.1393 (↓)	0.3467 ± 0.0429 (↓)	0.4695 ± 0.2009 (↓)	0.7619 ± 0.2086 (↓)	**0.9044** ± **0.0287**
Blood	0.7583 ± 0.0754 (↑)	**0.7677** ± **0.0196** (↑)	0.7021 ± 0.1462 (○)	0.6753 ± 0.1990 (○)	0.7609 ± 0.0252 (↑)	0.5247 ± 0.2671 (↓)	0.7387 ± 0.0357
WBC	0.9085 ± 0.0635 (↓)	0.9498 ± 0.0222 (○)	0.9538 ± 0.0159 (○)	0.7906 ± 0.2658 (↓)	**0.9580** ± **0.0157** (○)	0.6359 ± 0.2753 (↓)	0.9566 ± 0.0167
Haberman	0.6308 ± 0.2153 (○)	0.6812 ± 0.1301 (○)	0.6239 ± 0.1900 (↓)	0.6348 ± 0.1795 (↓)	0.6185 ± 0.1937 (↓)	0.4438 ± 0.2313 (↓)	**0.7221** ± **0.0391**
Pima	0.6617 ± 0.0484 (↓)	0.7049 ± 0.0407 (↓)	0.7155 ± 0.0312 (○)	0.7014 ± 0.0479 (↓)	0.7088 ± 0.0402 (↓)	0.6603 ± 0.0321 (↓)	**0.7255** ± **0.0322**
Iris	0.9911 ± 0.0161 (↓)	0.9978 ± 0.0068 (○)	**1.0000** ± **0.0000** (↑)	0.9689 ± 0.0841 (↓)	**1.0000** ± **0.0000** (↑)	0.6637 ± 0.0644 (↓)	0.9963 ± 0.0084
Wine	0.8358 ± 0.1395 (↓)	0.9604 ± 0.0570 (○)	0.9220 ± 0.0507 (↓)	0.7503 ± 0.1483 (↓)	**0.9704** ± **0.0246** (○)	0.7805 ± 0.1343 (↓)	0.9686 ± 0.0268
Balance	0.9144 ± 0.0184 (↑)	0.9215 ± 0.0166 (↑)	0.9151 ± 0.0197 (↑)	0.9177 ± 0.0199 (↑)	0.9110 ± 0.0205 (↑)	**0.9220** ± **0.0147** (↑)	0.8991 ± 0.0157
Vehicle	0.7513 ± 0.0175 (↓)	0.7446 ± 0.0319 (↓)	0.7530 ± 0.0285 (↓)	0.7458 ± 0.0217 (↓)	0.7520 ± 0.0220 (↓)	0.7552 ± 0.0192 (↓)	**0.7640** ± **0.0227**
Newthyroid	0.7713 ± 0.1776 (↓)	0.8892 ± 0.0697 (↓)	0.8969 ± 0.0368 (↓)	0.8251 ± 0.1782 (↓)	0.8949 ± 0.0516 (↓)	0.6262 ± 0.2603 (↓)	**0.9169** ± **0.0427**
Postoperative	0.6846 ± 0.1122 (○)	0.6833 ± 0.0780 (○)	**0.7103** ± **0.0843** (↑)	0.7026 ± 0.0654 (↑)	0.6244 ± 0.1236 (○)	0.6833 ± 0.0951 (○)	0.6654 ± 0.0910
Ecoli	0.8667 ± 0.1036 (↓)	0.9350 ± 0.0858 (○)	0.9459 ± 0.0425 (○)	0.8314 ± 0.1547 (↓)	0.9459 ± 0.0375 (○)	0.7403 ± 0.1808 (↓)	**0.9561** ± **0.0209**
Zoo	0.9133 ± 0.1366 (↓)	0.9778 ± 0.0331 (↓)	0.9722 ± 0.0392 (↓)	0.7011 ± 0.1964 (↓)	0.9889 ± 0.0220 (↓)	0.8156 ± 0.1861 (↓)	**0.9967** ± **0.0102**
Yeast	0.8477 ± 0.0216 (↓)	0.8634 ± 0.0217 (↓)	0.8628 ± 0.0236 (↓)	0.8564 ± 0.0199 (↓)	0.8596 ± 0.0166 (↓)	0.8457 ± 0.0172 (↓)	**0.8739** ± **0.0155**
↑/↓/○	2/16/2	2/11/7	3/12/5	2/17/1	3/10/7	1/18/1	-

thus, their obtained Pareto fronts are incomplete. The solution diversity of the CDMOEA is much better than that of the other MOEAs on the dataset, as the competitive decomposition mechanism can effectively avoid the population degeneration triggered by the invalid solutions and the Pareto front can be retained in the final phase of the CDMOEA.

In addition, the accuracies of the DNM and LCC optimized by different algorithms are compared in Tables IV and V. Concerning the accuracy of the DNM, the number of problems for which the CDMOEA has the best performance is 10 (of 20), which is much larger than those of the other MOEAs. According to the statistical results, the number of problems on which the CDMOEA performs significantly better than its competitors is always larger than the number of problems on which the CDMOEA performs significantly worse. This finding indicates that the CDMOEA can achieve better optimization performance for the DNM than the other MOEAs on most of the datasets. The advantages of the CDMOEA are more obvious in terms of the accuracy of the LCC. Specifically, the CDMOEA performs best on 14 (of 20) benchmark problems, while the other algorithms perform best on only 0–3 problems. In addition, the accuracy of the LCC optimized by the CDMOEA is significantly superior to that of the other MOEAs on most of these datasets because the solutions of the other algorithms always have relatively low architectural complexity; these oversimplified solutions result in degraded model performance when the corresponding architecture of the DNM is transformed into the LCC. Because the CDMOEA can achieve a relatively complete Pareto front, the solutions with better generalization performance can be selected from it for each problem. In general, it can be concluded that the CDMOEA can achieve better optimization performance than the other state-of-the-art MOEAs in solving the multiobjective DNM architecture search problem. Furthermore, the CDMOEA is compared with the backpropagation algorithm and several state-of-the-art single-objective evolutionary algorithms. Due to space limitations, the results are presented in the supplementary materialNote that,, showing that the CDMOEA still provides superior optimization performance.

TABLE VI
PARAMETER INITIALIZATION RANGES OF MACHINE-LEARNING TECHNIQUES

Algorithm	Parameter	Range	Attribute
KNN	Number of nearest neighbours (k)	[1, 50]	Integer
MLR	Type of model	Nominal	Categorical
RF	Number of trees ($Ntree$)	[10, 100]	Integer
DT	Minimum leaf size (Mls)	[1, 10]	Integer
	Maximum tree depth (Mtd)	[10, 20]	Integer
RBF	Spread (Sp)	[0.1, 2.0]	Real
	Maximum number of neurons (MN)	[5, 100]	Integer
MLP	Learning rate (Lr)	[0.01, 0.20]	Real
	Number of hidden layers (Hid)	[10, 30]	Integer
	Number of iterations ($Itera$)	[1000, 3000]	Integer
SVM	Kernel function	RBF	Categorical
	Regularization parameter (C)	[2^{-24}, 2^{25}]	Real
	Kernel parameter (γ)	[2^{-24}, 2^{25}]	Real
MODE	Scale factor (F)	[0.5, 1.0]	Real
	Crossover rate (CR)	[0.5, 1.0]	Real
	Archive size (As)	[50, 100]	Integer
CDMOEA	Scale factor (F)	[0.5, 1.0]	Real
	Crossover rate (CR)	[0.5, 1.0]	Real
	Maximum complexity (ψ)	(0, 1]	Real

D. Comparisons of Machine-Learning Techniques

To further verify the classification performance of both the DNM and LCC optimized by the proposed algorithm, seven commonly used machine-learning techniques are also used as competitors: 1) the *k*-nearest neighbors (KNNs); 2) decision tree (DT); 3) multilayer perceptron (MLP); 4) multinomial logistic regression (MLR); 5) radial basis network (RBF); 6) support vector machine (SVM); and 7) random forest (RF) algorithms. In addition, the DNM and LCC trained by a Pareto-based multiobjective differential evolution (MODE) algorithm also serve as the competitors [20].

Considering that the hyperparameters of the compared machine-learning techniques have a significant effect on their classification performances, some techniques have been proposed to tune the hyperparameters automatically, including grid search, random search, the Bayesian optimization strategy [59], and the orthogonal experiment-based approach [60]. The Bayesian optimization strategy is adopted in this study. The average of the 5-fold cross-validation (CV) approach serves as the objective function [61], and the number of evaluations is set to 30. Note that a 5-fold CV is executed only on the training dataset. The search ranges of the hyperparameters for each approach are presented in Table VI. To achieve a reliable comparison, the machine-learning techniques are compared with the DNM and LCC on each benchmark dataset using the most suitable hyperparameters.

The results of all the classifiers are summarized in Table VII. The DNM optimized by the CDMOEA achieves the best performance on eight benchmark datasets, performing much better than any other machine-learning technique. Interestingly, the LCC optimized by the CDMOEA outperforms all other classifiers, including the DNM, on three datasets. This finding implies that using the CDMOEA as the optimization algorithm does not affect the model performance when the DNM is implemented in hardware and that the generalization performance of the LCC can be even better. In addition, Friedman tests [62] are employed to statistically compare these classifiers through the KEEL software [63]. The significance level is set to 0.1. Moreover, Finner's procedure [64] is applied to control the familywise error rate by adjusting the *P*-value. In Table VIII, the DNM optimized by the CDMOEA ranks first, followed by the RF and LCC optimized by the CDMOEA. According to the adjusted *P*-values, the performance of the DNM trained by the proposed algorithm is significantly better than that of the other machine-learning techniques; the LCC also achieves competitive classification performance. In addition, the testing times of all the classifiers on the 20 benchmark problems are compared in Table IX. As a simplified version of the DNM, the LCC clearly has the highest computational speed among these classifiers. The LCC is approximately one order of magnitude faster than the second-ranked MLR. Note that considering the massively parallel computing in the actual hardware devices, the LCC has an obvious advantage in terms of computational speed compared with all the other classifiers.

E. Hyperparameter Sensitivity Analysis

The performance of the CDMOEA may be sensitive to three crucial hyperparameters, namely: 1) the subpopulation size Ps; 2) individual interval I_{ind}; and 3) overlap interval I_{over}. Accordingly, the analysis of the hyperparameter sensitivity is presented in this section. Since the CDMOEA allocates each subpopulation with a fixed number of solutions, Ps directly determines the population size. A larger population size means that the algorithm pays more attention to exploration, while a smaller population size enhances the exploitation ability of the algorithm. As such, P_s influences the balance between the exploration and exploitation of the CDMOEA. The individual interval I_{ind} determines the search range of each subproblem. The overlap interval I_{over} determines the degree of competitiveness between the subproblems. In the experiments, all the hyperparameters are set to four different values. As one hyperparameter changes, the others remain unchanged. The results are presented in Tables S.V–S.VII in the supplementary material. There are no obvious advantages among the values of each hyperparameter, suggesting that the CDMOEA is insensitive to these hyperparameters. In addition, the initial population variations may influence the performance of the proposed algorithm. The specific initialization mechanism of the CDMOEA is presented in Algorithm 3. We compare it with the commonly used initial population strategy of the other MOEAs, whereby all the solutions are initialized in a random way without considering their distribution in each subproblem. The results are presented in Table S.I. Clearly, the CDMOEA performs slightly better when using the initialization mechanism in Algorithm 3. However, there is no obvious performance degeneration when using a random initialization mechanism, implying that the CDMOEA is robust to initial population variations.

V. CONCLUSION

In this study, the CDMOEA was proposed to solve the large-scale DNM architecture search problem. Compared with

TABLE VII
CLASSIFICATION PERFORMANCE ACHIEVED BY THE COMPARED CLASSIFIERS ON 20 BENCHMARK PROBLEMS

	KNN	SVM	RBF	MLP	MLR	RF
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
Liver	0.6396 ± 0.0485	0.6202 ± 0.0648	0.5681 ± 0.0362	0.5921 ± 0.0340	0.6712 ± 0.0275	0.6848 ± 0.0313
Cancer	0.9639 ± 0.0098	0.9619 ± 0.0114	0.9653 ± 0.0096	0.9456 ± 0.0370	0.9595 ± 0.0125	**0.9676 ± 0.0313**
Australian	0.6808 ± 0.0170	0.7947 ± 0.0561	0.5547 ± 0.0349	0.8629 ± 0.0098	0.8614 ± 0.0236	0.8643 ± 0.0160
Japanese	0.6663 ± 0.0326	0.7686 ± 0.0578	0.5548 ± 0.0290	0.8453 ± 0.0177	0.8290 ± 0.0968	0.8604 ± 0.0203
Vertebral	0.8318 ± 0.0386	**0.8473 ± 0.0464**	0.6634 ± 0.0340	0.7933 ± 0.0473	0.8419 ± 0.0359	0.8239 ± 0.0188
Website	0.8951 ± 0.0101	0.8919 ± 0.0176	0.9054 ± 0.0056	0.8646 ± 0.0124	0.8719 ± 0.0171	0.9070 ± 0.0071
Ionosphere	0.8585 ± 0.0392	0.8886 ± 0.0291	**0.9368 ± 0.0196**	0.8996 ± 0.0309	0.3552 ± 0.0414	0.9244 ± 0.0209
Blood	0.7463 ± 0.0104	0.7647 ± 0.0215	0.7565 ± 0.0197	0.7680 ± 0.0270	0.7763 ± 0.0161	0.7567 ± 0.0160
WBC	**0.9743 ± 0.0116**	0.9585 ± 0.0184	0.9432 ± 0.0106	0.9665 ± 0.0113	0.9654 ± 0.0109	0.9696 ± 0.0095
Haberman	0.7457 ± 0.0315	0.7467 ± 0.0328	0.7174 ± 0.0294	0.7313 ± 0.0331	0.7217 ± 0.0391	0.6915 ± 0.0315
Pima	0.7316 ± 0.0207	0.7335 ± 0.0490	0.6478 ± 0.0270	0.7469 ± 0.0223	0.7600 ± 0.0248	0.7528 ± 0.0173
Iris	0.9929 ± 0.0210	0.9756 ± 0.0773	0.9982 ± 0.0056	0.9720 ± 0.0427	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
Wine	0.9268 ± 0.0211	0.9377 ± 0.0445	0.6830 ± 0.0721	0.9702 ± 0.0213	0.9585 ± 0.0278	0.9732 ± 0.0307
Balance	0.9169 ± 0.0088	0.9213 ± 0.0164	0.9151 ± 0.0178	0.9197 ± 0.0163	0.9170 ± 0.0091	0.9178 ± 0.0137
Vehicle	0.7587 ± 0.0233	0.7748 ± 0.0337	0.7449 ± 0.0281	0.7643 ± 0.0212	**0.7902 ± 0.0224**	0.7850 ± 0.0197
Newthyroid	0.9320 ± 0.0293	**0.9492 ± 0.0281**	0.8240 ± 0.0456	0.8898 ± 0.0444	0.8831 ± 0.0524	0.9354 ± 0.0303
Postoperative	0.6823 ± 0.0753	0.6846 ± 0.0596	0.6892 ± 0.0698	0.6562 ± 0.0655	0.6346 ± 0.0836	0.6169 ± 0.0426
Ecoli	0.9552 ± 0.0144	0.9535 ± 0.0288	0.9659 ± 0.0151	0.9556 ± 0.0123	**0.9663 ± 0.0116**	0.9519 ± 0.0143
Zoo	0.9907 ± 0.0164	0.9900 ± 0.0316	0.9813 ± 0.0322	0.9747 ± 0.0434	0.9967 ± 0.0105	0.9953 ± 0.0063
Yeast	**0.8871 ± 0.0102**	0.8681 ± 0.0311	0.8853 ± 0.0084	0.8572 ± 0.0105	0.8710 ± 0.0140	0.8830 ± 0.0098

	DT	DNM-MODE	LCC-MODE	DNM-CDMOEA	LCC-CDMOEA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
Liver	0.5990 ± 0.0398	0.6365 ± 0.0255	0.6227 ± 0.0292	**0.6873 ± 0.0349**	0.6219 ± 0.0342
Cancer	0.9348 ± 0.0135	0.9543 ± 0.0109	0.9494 ± 0.0113	0.9623 ± 0.0115	0.9597 ± 0.0105
Australian	0.8290 ± 0.0217	0.8537 ± 0.0216	0.8544 ± 0.0211	**0.8678 ± 0.0191**	0.8656 ± 0.0187
Japanese	0.8043 ± 0.0259	0.8488 ± 0.0091	0.8486 ± 0.0091	0.8626 ± 0.0125	**0.8633 ± 0.0129**
Vertebral	0.7925 ± 0.0333	0.8194 ± 0.0335	0.8004 ± 0.0385	0.8402 ± 0.0199	0.8105 ± 0.0155
Website	0.8978 ± 0.0086	0.8747 ± 0.0156	0.8658 ± 0.0108	**0.9072 ± 0.0103**	0.8632 ± 0.0168
Ionosphere	0.8724 ± 0.0324	0.4147 ± 0.1055	0.4162 ± 0.1068	0.9261 ± 0.0240	0.9246 ± 0.0235
Blood	0.7536 ± 0.0220	0.7675 ± 0.0247	0.6857 ± 0.0561	**0.7843 ± 0.0268**	0.7507 ± 0.0185
WBC	0.9468 ± 0.0135	0.9595 ± 0.0101	0.9576 ± 0.0107	0.9662 ± 0.0135	0.9602 ± 0.0115
Haberman	0.6815 ± 0.0290	0.7261 ± 0.0362	0.6752 ± 0.0567	**0.7485 ± 0.0351**	0.7385 ± 0.0325
Pima	0.6974 ± 0.0294	0.7356 ± 0.0172	0.7281 ± 0.0182	**0.7638 ± 0.0207**	0.7558 ± 0.0200
Iris	**1.0000 ± 0.0000**	0.9996 ± 0.0014	0.9987 ± 0.0042	0.9991 ± 0.0019	0.9933 ± 0.0076
Wine	0.9509 ± 0.0270	0.9717 ± 0.0146	0.9679 ± 0.0197	**0.9755 ± 0.0149**	0.9660 ± 0.0223
Balance	0.9074 ± 0.0155	0.9173 ± 0.0194	0.9165 ± 0.0188	**0.9251 ± 0.0116**	0.9250 ± 0.0131
Vehicle	0.7555 ± 0.0265	0.7699 ± 0.0160	0.7618 ± 0.0195	0.7827 ± 0.0239	0.7705 ± 0.0220
Newthyroid	0.8969 ± 0.0423	0.9009 ± 0.0344	0.8938 ± 0.0406	0.9388 ± 0.0152	0.9095 ± 0.0294
Postoperative	0.6577 ± 0.0984	0.6677 ± 0.0579	0.6723 ± 0.0775	0.7200 ± 0.0660	**0.7208 ± 0.0746**
Ecoli	0.9198 ± 0.0278	0.9493 ± 0.0145	0.9467 ± 0.0150	0.9646 ± 0.0162	0.9531 ± 0.0201
Zoo	0.9833 ± 0.0360	0.9920 ± 0.0098	0.9913 ± 0.0114	0.9960 ± 0.0064	**0.9987 ± 0.0028**
Yeast	0.8584 ± 0.0109	0.8700 ± 0.0120	0.8672 ± 0.0108	0.8852 ± 0.0129	0.8722 ± 0.0163

TABLE VIII
STATISTICAL RESULTS OF ALL CLASSIFIERS ON 20 BENCHMARK PROBLEMS

	Ranking	*Z-value*	$P_{unadjusted}$	P_{Finner}	$\alpha = 0.1$
KNN	6.20	3.813850	0.000137	**0.000274**	Yes
SVM	5.95	3.575485	0.000350	**0.000499**	Yes
RBF	7.70	5.244044	0.000000	**0.000001**	Yes
MLP	7.00	4.576620	0.000005	**0.000012**	Yes
MLR	5.25	2.908061	0.003637	**0.004544**	Yes
RF	4.10	1.811579	0.070051	**0.070051**	Yes
DT	8.55	6.054487	0.000000	**0.000000**	Yes
DNM-MODE	6.15	3.766177	0.000166	**0.000276**	Yes
LCC-MODE	7.95	5.482410	0.000000	**0.000000**	Yes
DNM-CDMOEA	**2.20**	-	-	-	-
LCC-CDMOEA	4.95	2.622022	0.008741	**0.009707**	Yes

TABLE IX
RUNNING TIME OF ALL THE CLASSIFIERS ON 20 BENCHMARK PROBLEMS

	KNN	SVM	RBF	MLP	MLR	RF	DT	DNM	LCC
Liver	0.0846	0.0872	0.5885	0.5945	0.0202	4.2444	0.0225	0.0514	**0.0022**
Cancer	0.1145	0.0951	0.5603	0.6385	0.0095	7.9338	0.0166	0.1095	**0.0016**
Australian	0.1836	0.0974	0.9331	0.6180	0.0160	9.3741	0.0185	0.1167	**0.0005**
Japanese	0.1960	0.2485	0.5526	0.7898	0.0091	2.6915	0.0176	0.0976	**0.0026**
Vertebral	0.1124	0.0593	0.5516	0.6337	0.0087	1.8425	0.0163	0.0204	**0.0017**
Website	0.1920	0.2424	0.5834	0.7685	0.0136	3.9291	0.0212	0.1529	**0.0007**
Ionosphere	0.1069	0.0725	0.5599	0.7371	0.0105	2.5743	0.0217	0.2088	**0.0006**
Blood	0.1351	0.1651	0.5579	0.6716	0.0101	1.1305	0.0166	0.0417	**0.0031**
WBC	0.1446	0.1249	0.5633	0.9213	0.0089	7.6689	0.0164	0.0571	**0.0018**
Haberman	0.0992	0.0538	0.5491	0.6840	0.0157	6.2155	0.0170	0.0159	**0.0010**
Pima	0.1218	0.1745	0.5418	0.6151	0.0114	8.5289	0.0250	0.0639	**0.0006**
Iris	0.1057	0.0219	0.5558	0.6391	0.0097	7.0797	0.0128	0.0085	**0.0004**
Wine	0.0968	0.0260	0.5521	0.6044	0.0089	9.6331	0.0282	0.0339	**0.0014**
Balance	0.1230	0.1920	0.5589	0.6160	0.0088	11.213	0.0153	0.0371	**0.0017**
Vehicle	0.1551	0.2428	0.5528	0.6343	0.0109	1.5034	0.0205	0.1564	**0.0009**
Newthyroid	0.1456	0.0284	0.5363	0.6225	0.0119	5.6514	0.0209	0.0123	**0.0012**
Postoperative	0.1233	0.0128	0.5372	0.6097	0.0085	3.0483	0.0136	0.0108	**0.0004**
Ecoli	0.1016	0.0654	0.5323	0.6111	0.0115	1.8111	0.0153	0.0297	**0.0006**
Zoo	0.1165	0.0240	0.5240	0.6964	0.0099	7.3059	0.0224	0.0453	**0.0004**
Yeast	0.1932	0.3095	0.5441	0.5902	0.0112	4.5397	0.0218	0.1178	**0.0011**
Total (s)	2.6515	2.3435	11.435	13.296	0.2250	107.92	0.3802	1.3877	**0.0235**

other MOEAs, the CDMOEA can find more solutions distributed over the entire Pareto front. After optimization by the CDMOEA, the DNM and LCC also achieve very competitive classification performance compared to several commonly used classifiers. The threats to validity in implementing the proposed approach are as follows: the internal threats to validity include the randomness and errors caused by the stochastic operators in the experiments. Each experiment was conducted independently 30 times, which can reduce the likelihood of anomalous results. Moreover, all the results were double-checked to mitigate internal threats. In addition, the reliability and generalizability of the results can be regarded as external threats. Comprehensive comparisons between the proposed approach and state-of-the-art techniques, which contain single-objective and MOEAs, were conducted in our experiments.

These comparisons can greatly improve the reliability of our conclusions. Moreover, a large number of publicly available datasets were used in our experiments, which can enhance the generalizability of the proposed approach. Consequently, it can be concluded that the proposed CDMOEA is a reliable algorithm for solving the architecture search problem of the DNM. To the best of our knowledge, the DNM is the only ANN that can be completely implemented on an FPGA for parallel computation.

However, it is notable that all the classification problems adopted in this study are low dimensional. Compared with deep learning techniques, the DNM and LCC cannot yet provide satisfactory results for complex practical problems, especially with high-dimensional features. An overly large number of features leads to vanishing gradient problems that result in the stagnation of the gradient-based algorithm and produce an exponentially growing search space that causes the CDMOEA's search to be obscure. These disadvantages will make it challenging to optimize the neural architecture of the DNM. Although not all practical applications are large scale, and the conventional machine-learning approaches still play a crucial role in real-world problems, this weakness limits the popularity of the DNM and LCC. In our future work, we will attempt to overcome this weakness via, for example, structural modification, input feature resampling, and the development of effective excitation functions and specific optimization algorithms.

REFERENCES

[1] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, Sep. 2014.

[2] D. Laney, *The Importance of 'Big Data': A Definition*, Gartner, Stamford, CT, USA, 2012.

[3] E. Q. Wu *et al.*, "Nonparametric Bayesian prior inducing deep network for automatic detection of cognitive status," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5483–5496, Nov. 2021.

[4] E. Q. Wu *et al.*, "Self-paced dynamic infinite mixture model for fatigue evaluation of pilots' brains," *IEEE Trans. Cybern.*, early access, Dec. 7, 2020, doi: 10.1109/TCYB.2020.3033005.

[5] J. Von Neumann and R. Kurzweil, *The Computer and the Brain*. New Haven, CT, USA: Yale Univ. Press, 2012.

[6] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[7] I. Boybat *et al.*, "Neuromorphic computing with multi-memristive synapses," *Nat. Commun.*, vol. 9, no. 1, pp. 1–12, 2018.

[8] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nat. Nanotechnol.*, vol. 11, no. 8, p. 693, 2016.

[9] Y.-H. Liu and X.-J. Wang, "Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron," *J. Comput. Neurosci.*, vol. 10, no. 1, pp. 25–45, 2001.

[10] S. Binczak, S. Jacquir, J.-M. Bilbault, V. B. Kazantsev, and V. I. Nekorkin, "Experimental study of electrical FitzHugh–Nagumo neurons with modified excitability," *Neural Netw.*, vol. 19, no. 5, pp. 684–693, 2006.

[11] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[12] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, p. 500, 1952.

[13] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," 2017, *arXiv:1705.06963*.

[14] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, "A review of learning in biologically plausible spiking neural networks," *Neural Netw.*, vol. 122, pp. 253–272, Feb. 2020.

[15] E. Q. Wu, C.-T. Lin, L.-M. Zhu, Z. Tang, Y.-W. Jie, and G.-R. Zhou, "Fatigue detection of pilots' brain through brains cognitive map and multilayer latent incremental learning model," *IEEE Trans. Cybern.*, early access, May 7, 2021, doi: 10.1109/TCYB.2021.3068300.

[16] Z. Tang, H. Tamura, O. Ishizuka, and K. Tanno, "A neuron model with interaction among synapses," *IEEJ Trans. Electron. Inf. Syst.*, vol. 120, no. 7, pp. 1012–1019, 2000.

[17] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, Jan. 2016.

[18] L. Luo and D. D. O'Leary, "Axon retraction and degeneration in development and disease," *Annu. Rev. Neurosci.*, vol. 28, pp. 127–156, Jul. 2005.

[19] J. Ji, S. Song, Y. Tang, S. Gao, Z. Tang, and Y. Todo, "Approximate logic neuron model trained by states of matter search algorithm," *Knowl. Based Syst.*, vol. 163, pp. 120–130, Jan. 2019.

[20] J. Ji *et al.*, "Accuracy versus simplification in an approximate logic neural model," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 11, pp. 5194–5207, Nov. 2021.

[21] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

[22] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, "Unsupervised learnable neuron model with nonlinear interaction on dendrites," *Neural Netw.*, vol. 60, pp. 96–103, Dec. 2014.

[23] Y. Todo, Z. Tang, H. Todo, J. Ji, and K. Yamashita, "Neurons with multiplicative interactions of nonlinear synapses," *Int. J. Neural Syst.*, vol. 29, no. 8, 2019, Art. no. 1950012.

[24] C. Tang, J. Ji, Y. Tang, S. Gao, Z. Tang, and Y. Todo, "A novel machine learning technique for computer-aided diagnosis," *Eng. Appl. Artif. Intell.*, vol. 92, Jun. 2020, Art. no. 103627.

[25] Z. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE Trans. Inf. Syst.*, vol. 98, no. 7, pp. 1365–1376, 2015.

[26] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowl. Based Syst.*, vol. 105, pp. 214–224, Aug. 2016.

[27] Y. Tang, J. Ji, Y. Zhu, S. Gao, Z. Tang, and Y. Todo, "A differential evolution-oriented pruning neural network model for bankruptcy prediction," *Complexity*, vol. 2019, Aug. 2019, Art. no. 8682124.

[28] Z. Song, Y. Tang, J. Ji, and Y. Todo, "Evaluating a dendritic neuron model for wind speed forecasting," *Knowl. Based Syst.*, vols. 201–202, Aug. 2020, Art. no. 106052.

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

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

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

[32] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[33] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, pp. 1–24, 2016.

[34] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

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

[36] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multiobjective optimization based on problem transformation," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 260–275, Apr. 2018.

[37] C. He *et al.*, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 949–961, Dec. 2019.

[38] R. Chai, A. Savvaris, A. Tsourdos, Y. Xia, and S. Chai, "Solving multiobjective constrained trajectory optimization problem by an extended evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1630–1643, Apr. 2020.

[39] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. P. Chen, "Multiobjective optimal parking maneuver planning of autonomous wheeled vehicles," *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10809–10821, Dec. 2020.

[40] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, "Trajectory optimization of space maneuver vehicle using a hybrid optimal control solver," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 467–480, Feb. 2019.

[41] R. Chai, A. Tsourdos, A. Savvaris, Y. Xia, and S. Chai, "Real-time reentry trajectory planning of hypersonic vehicles: A two-step strategy incorporating fuzzy multiobjective transcription and deep neural network," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 6904–6915, Aug. 2020.

[42] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 169–190, Apr. 2017.

[43] B. H. Nguyen, B. Xue, P. Andreae, H. Ishibuchi, and M. Zhang, "Multiple reference points-based decomposition for multiobjective feature selection in classification: Static and dynamic mechanisms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 170–184, Feb. 2020.

[44] H. Xu, B. Xue, and M. Zhang, "A duplication analysis based evolutionary algorithm for bi-objective feature selection," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 205–218, Apr. 2021.

[45] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," 2018, *arXiv:1804.09081*.

[46] Z. Yang *et al.*, "CARS: Continuous evolution for efficient neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1829–1838.

[47] Z. Lu *et al.*, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 419–427.

[48] X. Chu, B. Zhang, and R. Xu, "Multi-objective reinforced evolution in mobile neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 99–113.

[49] M. Wu, K. Li, S. Kwong, and Q. Zhang, "Evolutionary many-objective optimization based on adversarial decomposition," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 753–764, Feb. 2020.

[50] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 525–537, Jun. 2019.

[51] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://archive.ics.uci.edu/ml/index.php

[52] C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," in *Proc. 3rd Int. Conf. Evol. Multi-Criterion Optim. (EMO)*, vol. 216, 2005, p. 240.

[53] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point," in *Proc. 10th ACM SIGEVO Workshop Found. Genet. Algorithms*, 2009, pp. 87–102.

[54] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[55] Q. Lin, J. Li, Z. Du, J. Chen, and Z. Ming, "A novel multi-objective particle swarm optimization with multiple search strategies," *Eur. J. Oper. Res.*, vol. 247, no. 3, pp. 732–744, 2015.

[56] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3696–3708, Aug. 2020.

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

[58] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm. Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

[59] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25. Red Hook, NY, USA: Curran Assoc., 2012, pp. 2951–2959.

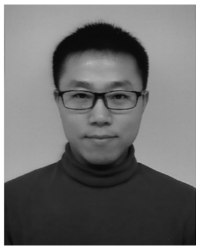
[60] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1400–1413, Apr. 2022.

[61] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Stat. Surveys*, vol. 4, pp. 40–79, Mar. 2010.

[62] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

[63] J. Alcalá-Fdez *et al.*, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009.

[64] H. Finner, "On a monotonicity problem in step-down multiple test procedures," *J. Amer. Stat. Assoc.*, vol. 88, no. 423, pp. 920–923, 1993.

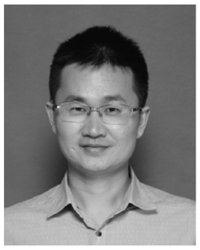


Junkai Ji received the B.S. degree from the Hefei University of Technology, Hefei, Anhui, China, in 2013, and the M.S. and D.E. degrees from the University of Toyama, Toyama, Japan, in 2016 and 2018, respectively.

In 2019, he joined Shenzhen University, Shenzhen, China, where he is currently a Research Fellow with the College of Computer Science and Software Engineering. His current research interests include neural networks, evolutionary computation, and computer-aided drug design.

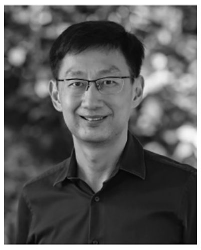
Jiajun Zhao received the B.S. degree from Guangdong Polytechnic Normal University, Guangzhou, Guangdong, China, in 2019. He is currently pursuing the M.S. degree with Shenzhen University, Shenzhen, Guangdong, China.

His main research interests include neural networks and evolutionary computation.



Qiuzhen Lin (Member IEEE) received the B.S. degree from Zhaoqing University, Zhaoqing, China, in 2007, the M.S. degree from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2014.

He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University. He has published over 60 research papers since 2008. His current research interests include artificial immune system, multiobjective optimization, and dynamic system.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (First Class Hons.) and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently a Chair Professor (Computational Intelligence) with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He has published over 300 refereed articles and seven books.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He served as the Editor-in-Chief for the *IEEE Computational Intelligence Magazine* from 2010 to 2013 and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020. He currently serves as an Editorial Board Member for more than ten journals. He is an IEEE Distinguished Lecturer Program Speaker and the Chief Co-Editor of Springer Book Series on *Machine Learning: Foundations, Methodologies, and Applications*.