Learning to Guide Particle Search for Dynamic Multi-objective Optimization

Wei Song, Member, IEEE, Shaocong Liu, Xinjie Wang, Yinan Guo, Member, IEEE, Shengxiang Yang, Senior Member, IEEE, and Yaochu Jin, Fellow, IEEE

Abstract—Dynamic multi-objective optimization problems (DMOPs) are characterized by multiple objectives that change over time in varying environments. More specifically, environmental changes can be described as various dynamics. However, it is difficult for existing dynamic multi-objective algorithms (DMOAs) to handle DMOPs due to their inability to learn in different environments to guide the search. Besides, solving DMOPs is typically an online task, requiring low computational cost of a DMOA. To address the above challenges, we propose a particle search guidance network (PSGN), capable of directing individuals’ search actions including learning target selection and acceleration coefficient control. PSGN can learn the actions that should be taken in each environment through rewarding or punishing the network by reinforcement learning. Thus, PSGN is capable of tackling DMOPs of various dynamics. Additionally, we efficiently adjust PSGN hidden nodes and update the output weights in an incremental learning way, enabling PSGN to direct particle search at a low computational cost. We compare the proposed PSGN with seven state-of-the-art algorithms, and the excellent performance of PSGN verifies that it can handle DMOPs of various dynamics in a computationally very efficient way.

Index Terms—Dynamic multi-objective optimization, neural network, incremental learning, particle swarm optimization, reinforcement learning.

I. INTRODUCTION

ANY optimization problems in science and engineering fields involve multiple conflicting objectives that change over time [1]. Such optimization problems are called dynamic multi-objective optimization problems (DMOPs). A real-world instance is a raw ore allocation (ROA) optimization problem in mineral processing, which is crucial for promoting the utilization ratio of nonrenewable raw mineral resources.

This work was supported by the National Natural Science Foundation of China (62076110), the National Foreign Experts Program (DL20231440011L), the Natural Science Foundation of Jiangsu Province (BK20181341), and the Fundamental Research Funds for the Central Universities (JUSRP221027).

(Corresponding authors: Wei Song.)

W. Song, S. Liu are with the School of Artificial Intelligence and Computer Science, the Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, Wuxi, 214122, China (e-mail: songwei@jiangnan.edu.cn; scliu@stu.jiangnan.edu.cn).

X. Wang is with key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai, 200237, China (e-mail: xinjiewang@ecust.edu.cn).

Y. Guo is with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, 221116, China (e-mail: guoyinhan@cumt.edu.cn).

S. Yang is with the Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U.K. (e-mail: syang@dmu.ac.uk).

Y. Jin is with the Trustworth and General Artificial Intelligence Laboratory, School of Engineering, Westlake University, Hangzhou 331712, China (e-mail: jinyaochu@westlake.edu.cn).

This optimization problem refers to multiple goals, i.e., improving the concentrate yield, concentrate grade, decreasing the beatification ratio, and production cost via assigning different quantities of various types of raw ore. Moreover, variations in equipment capability and runtime result in changes in some model parameters [2].

In the literature, most existing dynamic multi-objective algorithms (DMOAs) randomly reinitialize the population [3], predict a new population [1] and/or reuse historical individuals archived [4] when the environment changes. Some static multi-objective optimization algorithms (SMOAs) [5], [6] are then employed to track the varying Pareto-optimal front (POF). However, in the field of dynamic multi-objective optimization, environmental changes can be described as different dynamics forming different levels of difficulties to DMOA [1]. Faced with various dynamics, these DMOAs are difficult to handle due to inability of learning in each environment to guide the search. Thus, guiding individuals’ search that can handle various dynamics poses the first challenge for DMOAs. To track the optima in environments of various dynamics, a neural network-guided particle search method is proposed in this article as it can automatically excavate useful evolution information in each environment, no matter which dynamic occurs. In addition, solving DMOPs is a typically online task with high demand for time. However, in response to environmental changes, lots of prediction-based DMOAs among the latest progress require a great deal of parameter calculation in the prediction models, such as iterative training (e.g., backpropagation, BP [7]), exhaustive search (e.g., grid search [8]) and transfer learning (e.g., transfer component analysis [9]), which are generally time-consuming. Therefore, such a circumstance poses the second challenge to satisfy the low time cost in solving DMOPs.

To make the response to environmental changes and track the varying POF, many DMOAs have been devised. Generally, they can be classified into three categories: 1) diversity-based algorithms [3], [10], 2) memory-based algorithms [4], [11], and 3) prediction-based algorithms [2], [12]. On the detection of an environmental change, diversity-based algorithms reinitialize the population by introducing some random individuals to tackle diversity loss [3], [13]. However, these algorithms may cause an imbalance between the convergence and diversity of the population when the random individuals are introduced in excess [1]. Memory-based algorithms make use of one or more memory pools to record historical Pareto-
optimal solutions (POS), with the goal of generating a feasible population at the beginning of each environment [4]. Whereas, such algorithms can usually be applied to environments of periodic dynamic and will be invalid for those of nonperiodic dynamic, because historical solutions cannot adapt to the following nonperiodic environments [1]. Based on the experience gained from historical evolution information, prediction-based algorithms can estimate high-quality initial populations under environments that change linearly [8]. Namely, these algorithms can predict the initial populations expected to approach the real POFs in linearly varying environments. However, if an environment changes nonlinearly, the algorithms are difficult to estimate high-quality initial populations, affecting the overall performance adversely [14]. Therefore, if a neural network-guided DMOA can be devised capable of directing individuals’ search by excavating useful evolution information in each environment, it would facilitate the enhancement of the performance in tackling DMOPs of various dynamics.

Tracking the varying POF in dynamic environments is a typical online task with a high demand for time. In the existing DMOAs that employ machine learning methods (e.g., some prediction-based DMOAs) [8], [15], a widely used parameter learning method is the BP [16] where gradients can be calculated by propagation from the output layer to the input layer of a network. However, the BP usually requires an iterative process to update network parameters, which is time-consuming and substantially increases the total time of solving DMOPs. It should be mentioned that it is difficult to select an appropriate learning rate in the BP. If the learning rate is too small, the parameter learning converges very slowly. Namely, this circumstance increases the number of iterations and hence lengthens the training time. By contrast, if the learning rate is too large, the parameter learning becomes unstable and diverges [17]. Different from the BP, the grid search is used in the support vector machine to find the optimal value of the regularization coefficient [8]. Because the grid search is an exhaustive method, it is generally time-consuming. In addition, the transfer learning maps the POF obtained from the previous environment to a latent space where high-quality individuals can be identified to form a initial population in the new environment [9]. However, identifying the optimal latent space increases the time cost of solving DMOPs greatly. That is because a large amount of parameter calculation is required in the transfer component analysis. Therefore, if a DMOA with a low time cost of parameter learning can be elaborately designed, it would be helpful to meet the high demand of time in solving DMOPs.

This article proposes a particle search guidance network (PSGN) for dynamic multi-objective optimization. We briefly summarize the functions of PSGN as follows: PSGN is capable of directing the search actions of each individual input including learning target selection and acceleration coefficient control. In the objective subspace to which an input individual belongs, a hidden node center close to the weight vector tends to be selected (with a high probability) as the local learning target of the individual. Besides, the population selects a hidden node center from the objective subspace that contains the fewest non-dominated solutions as the global learning target. Through the feedforward calculation of PSGN, the acceleration coefficients of each individual input can be controlled. Thus, based on the learning target selection and acceleration coefficient control, PSGN is capable of directing individuals’ search.

The main contributions of this article are summarized as follows. First, we leverage the reinforcement learning [18]–[20] to gain the expected output of the network. PSGN can learn the actions that should be taken in each environment on the basis of the expected output. In consequence, it can direct the search of each individual input regardless of which dynamic happens and hence can handle DMOPs under environments of various dynamics. Second, the hidden nodes of PSGN are grewed in batches and there is no network update required after pruning some insignificant hidden nodes. The incremental extreme learning machine [21] is introduced to train the network at an extremely fast learning speed. Thus, the hidden nodes and weights of PSGN can be adjusted efficiently, enabling PSGN to solve DMOPs at a low time cost. The experimental results especially performance comparisons with seven state-of-the-art competitors demonstrate that PSGN significantly improves the performance in solving DMOPs.

The remainder of this article is organized as follows. In Section II we briefly review the related work including DMOAs, other learning-based optimization algorithms, and incremental extreme learning machine (ELM). Section III describes the details of the proposed PSGN. In Section IV we compare PSGN with seven state-of-the-art algorithms and two ablation algorithms. Additionally, the parameter sensitivity analysis of PSGN on different test problems is presented. Section V draws a summary of this article and outlines future work.

II. RELATED WORK

A. Dynamic Multi-objective Optimization Algorithms

As one of representative diversity-based algorithms, the dynamic non-dominated sorting genetic algorithm II (DNSGA-II) [4] replaces a number of individuals in the population by a random reinitialization, with the goal of improving the diversity of the population. However, when the reinitialization is implemented in excess, it will reduce the efficiency of searching the POS. Similar to DNSGA-II, the similarity-based cooperative co-evolutionary algorithm [22] randomly reinitializes some individuals to improve the population diversity. Besides, such an algorithm leverages a strategy on the basis of change intensity to track the POF. However, this strategy is computationally intensive to search for optimal solutions in a group that is weakly impacted by the changing interval parameter [23].

Different from the diversity-based algorithms, the dynamic multi-objective evolutionary algorithm [13] employs an adaptive population management strategy to reuse excellent individuals that are stored in an archive in response to environmental changes. In addition to a single archive, double archives are also taken into account where the first one has the same function to the single archive (storing excellent individuals near POF) while the second one stores diverse individuals considering the diversity of the population [24]. Though the
use of archive(s), these algorithms can be interpreted as memory-based ones. However, memory-based algorithms tend to become ineffective for DMOPs in nonperiodic environments, because the archived historical solutions cannot fit nonperiodic environments [1].

Based on the idea of prediction, historical knee points are considered as high-quality solutions to perform transfer learning, which contributes to preventing negative transfers [15]. However, if an environment changes non-linearly, it will be difficult to predict a high-quality initial population on detecting an environmental change. In addition, some historical information (such as center points and manifolds) is used expecting to estimate a high-quality initial population in each new environment [25]. Besides the difficulty in predicting high-quality initial populations in non-linearly varying environments, historical information is insufficient in some early environments, affecting the overall performance in solving DMOPs.

In view of the issues of the above DMOAs, especially the difficulty of handling various dynamics, in this article we propose a neural network-guided DMOA (i.e., PSGN) capable of directing individuals’ search by excavating useful evolution information in each environment, no matter which dynamic occurs. Thus, the proposed PSGN can handle DMOPs under environments of various dynamics.

B. Other Learning-based Optimization Algorithms

In the field of optimization, some studies consider accelerating the search by learning. In the accelerated evolutionary search strategy [25], a multi-layer perception with one hidden layer is trained by poor and elitist solution pairs. In consequence, a gradient-descent-like direction vector can be learned for each solution, which contributes to enhancing the search efficiency. Motivated by the integration of learning and evolution, the learning-aided evolutionary optimization framework leverages the evolution knowledge learned by an artificial neural network from the evolution process [27]. The above two algorithms have been demonstrated to be successful in handling static optimization problems. Faced with dynamic environments, their performance may be affected due to insufficient training data, especially in frequently changing environment. By contrast, in this article we define comprehensive evolution criteria to assess the evolution effect of each individual. Moreover, the reinforcement learning is utilized to obtain the expected output of PSGN not only by rewarding, but also by punishing the network, providing adequate data for training. Thus, PSGN can learn the actions that should be taken for different iterations in each environment, which facilitates handling DMOPs of various dynamics.

C. Incremental Extreme Learning Machine

ELM is an extremely fast learning method for single-hidden-layer feedforward networks (SLFNs) in which the output weights linking the hidden layer to the output layer can be analytically determined with very few steps and very low computational cost [17]. The learning speed of ELM can be thousands of times faster than traditional network learning algorithms such as the BP algorithm [17]. Moreover, ELM not only tends to reach the smallest training error but also the smallest norm of weights, resulting in the good generalization performance of SLFNs based on Bartlett’s theory [28]. The output weight matrix $\beta \in \mathbb{R}^{L \times M}$ of a SLFN can be analytically calculated by ELM as:

$$\beta = H^T F$$

(1)

where

$$H = \begin{bmatrix} h_1(X_1) & \cdots & h_L(X_1) \\ \vdots & \ddots & \vdots \\ h_1(X_N) & \cdots & h_L(X_N) \end{bmatrix}$$

(2)

where $H \in \mathbb{R}^{N \times L}$ is the output matrix of the hidden layer for $N$ samples input, $L$ is the number of hidden nodes, $h_l(X_j)$ is the output of the $l$th hidden node to the $j$th sample input $X_j$, $H^\dagger \in \mathbb{R}^{L \times N}$ is the Moore-Penrose generalized inverse of $H$, $F \in \mathbb{R}^{N \times M}$ is the expected output matrix, and $M$ is the number of output nodes. ELM directly calculates the output weights of all the hidden nodes in an analytical way.

Incremental learning focuses on updating the output weights of incremental hidden nodes in a fast recurrence manner [21]. Incremental ELM analytically determines the output weights of the hidden nodes incremented, reducing the computational complexity significantly [21]. Suppose $K$ hidden nodes are incremented, the output weight matrix $\beta' \in \mathbb{R}^{(L+K) \times M}$ can be calculated by incremental ELM as:

$$\beta' = H'^T F = \begin{bmatrix} U \\ D \end{bmatrix} F$$

(3)

$$U = H'^\dagger (I - \delta H^T D)$$

(4)

$$D = ((I - HH^\dagger)\delta H)^\dagger$$

(5)

where

$$H' = \begin{bmatrix} h_1(X_1) & \cdots & h_{L+K}(X_1) \\ \vdots & \ddots & \vdots \\ h_1(X_N) & \cdots & h_{L+K}(X_N) \end{bmatrix}$$

(6)

where $H' \in \mathbb{R}^{N \times (L+K)}$ is the output matrix of the hidden layer after increasing $K$ hidden nodes, $H'^\dagger \in \mathbb{R}^{(L+K) \times N}$ is the Moore-Penrose generalized inverse of $H'$, $\delta H \in \mathbb{R}^{N \times K}$ is the output matrix of the $K$ hidden nodes incremented, $I$ is an identity matrix, $U \in \mathbb{R}^{L \times N}$ and $D \in \mathbb{R}^{K \times N}$ are the upper and lower component matrices of $H'^\dagger$, respectively. Compared with ELM that recalculates the output weights based on the entire new hidden layer output matrix whenever the network architecture is changed, incremental ELM reduces the computational complexity by only updating the output weights incrementally each time [21].

III. PROPOSED ALGORITHM

A. Framework of PSGN

The proposed PSGN is a single hidden layer neural network, which receives the position of an individual each time as its input. Thus, the number of the nodes in the input layer corresponds to the dimension of each individual. In PSGN,
Each hidden node contains a center set with a non-dominated solution on behalf of a candidate learning target of individuals and a width set to a fixed value. Each individual input selects its local and global learning targets from the hidden node centers of PSGN. Through the feedforward calculation, PSGN can control the local and global acceleration coefficients of each individual. After selecting learning targets and controlling acceleration coefficients, we update the position and multi-objective fitness values. Moreover, the evolution effect that reflects environment feedback can be acquired via comprehensive evolution criteria. On the basis of environment feedback, the reinforcement learning is utilized to obtain the expected output of PSGN by rewarding or punishing the network. Guided by the expected output, the hidden nodes and output weights of PSGN can be adjusted, enabling the network to direct individuals’ search. Fig. 1 outlines the framework of PSGN. For the readability, the associated symbols used in PSGN are provided in Table I.

**B. Hidden Node Initialization**

We divide the objective space into \( K \) subspaces through the use of the weight vector generation method. In each of the subspaces, the archived non-dominated solution nearest to the corresponding weight vector is selected as the center of an initial hidden node. The weight vector generation method specifies \( K \) uniformly distributed weight vectors, which contributes to obtaining \( K \) initial hidden node centers that are distant from each other in the objective space and hence the evenly distributed POF, because the individuals in population will be guided by the hidden node centers on behalf of learning targets. Based on the specified weight vector set \( \mathbf{W} = \{ \mathbf{w}_1, ..., \mathbf{w}_j, ..., \mathbf{w}_K \} \) where \( \mathbf{w}_j \in \mathbb{R}^m \) represents the \( j \)th weight vector and \( m \) is the number of objectives, the objective subspace index \( s \) of a solution \( \mathbf{x} \) can be calculated as:

\[
 s = \arg\max_{1 \leq k \leq K} \langle F(\mathbf{x}, t), \mathbf{w}_j \rangle
\]

where \( \langle F(\mathbf{x}, t), \mathbf{w}_j \rangle \) is the cosine of the angle between \( \mathbf{w}_j \) and the reference line formed by the objective vector \( F(\mathbf{x}, t) \) and the ideal point \( \mathbf{z} \in \mathbb{R}^m \) at time step \( t \). Specifically, \( \langle F(\mathbf{x}, t), \mathbf{w}_j \rangle \) is calculated as:

\[
 \langle F(\mathbf{x}, t), \mathbf{w}_j \rangle = \frac{(F(\mathbf{x}, t) - \mathbf{z}) \cdot \mathbf{w}_j}{\|F(\mathbf{x}, t) - \mathbf{z}\|_2 \times \|\mathbf{w}_j\|_2}
\]

where \( \mathbf{z} = (z_1, ..., z_i, ..., z_m) \), \( z_i \) indicates the minimum value of the \( i \)th objective \( f_i \) so far, and \( \| \cdot \|_2 \) is the 2-norm of a vector. Note that an individual might locate at the junction of some adjacent objective subspaces. In this circumstance, we randomly select one of the adjacent objective subspaces. In this way, we can divide a two-dimensional objective space where \( K = 6 \). As aforementioned, in each objective subspace we select the archived non-dominated solution nearest to the corresponding weight vector as the center of an initial hidden node. Accordingly, in the \( j \)th objective subspace the archived non-dominated solution \( \mathbf{n}_s \) with the largest \( \langle F(\mathbf{n}_s, t), \mathbf{w}_j \rangle \) value is selected as the center of the initial hidden node and totally \( K \) initial hidden nodes can be set from \( K \) objective subspaces. For the \( k \)th hidden node, the response \( \varphi_k(\mathbf{x}_i) \in \mathbb{R}^D \) to the \( i \)th input individual \( \mathbf{x}_i \in \mathbb{R}^D \) in \( D \)-dimensional decision space is calculated as:

\[
 \varphi_k(\mathbf{x}_i) = \exp \left( -\frac{\|\mathbf{x}_i - \mathbf{\mu}_k\|^2}{\sigma_k^2} \right)
\]

where \( \sigma_k \in \mathbb{R} \) and \( \mathbf{\mu}_k \in \mathbb{R}^D \) are the width and center of the \( k \)th hidden node, respectively.
In each iteration, the non-dominated sorting is employed to obtain non-dominated solutions to be archived. If the previous non-dominated solutions archived become dominated, they will be removed from the archive. When the archive size exceeds its limit, we repeat the following steps until the archive size satisfies the limit: first, we select the objective subspace with the largest number of archived non-dominated solutions and second, randomly remove one non-dominated solution from the objective subspace.

C. Search Guidance

The proposed PSGN is capable of directing the search actions of each individual input, i.e., selecting its learning targets and controlling its acceleration coefficients, which is detailed as follows:

1) Selecting Learning Targets: In PSGN, each individual input selects a hidden node center based on its selection probability from the objective subspace the individual belongs to as its local learning target \(Lbest\). In the \(l\)th objective subspace, the selection probability \(P^i_l\) of the \(i\)th hidden node center \(\mu_i\) is defined as:

\[
P^i_l = \frac{\langle F(\mu_i, t), w_l \rangle}{\sum_{j=1}^{k} \langle F(\mu_j, t), w_l \rangle}
\]

where \(k\) is the number of the hidden nodes within the \(l\)th objective subspace, and \(w_l \in \mathbb{R}^m\) is the \(l\)th weight vector. The smaller the angle between a hidden node center and the corresponding weight vector, the greater the selection probability, facilitating the exploitation of this objective subspace.

Additionally, to select the global learning target, we consider the distribution of the archived non-dominated solutions in the objective space. Specifically, we first select the objective subspace that contains the fewest non-dominated solutions archived. In such an objective subspace, the hidden node center nearest to the corresponding weight vector is selected as the global learning target \(Gbest\), which contributes to the even distribution of the obtained solutions in the objective space.

2) Controlling Acceleration Coefficients: The output layer of PSGN consists of three output nodes, which represent increasing, decreasing and keeping the local acceleration coefficient, respectively. For each individual input, PSGN selects the output node with the maximal output as the action node to control the local acceleration coefficient. In the output layer, the actual output \(y_i \in \mathbb{R}^3\) for the \(i\)th individual input \(x_i\) is calculated by:

\[
y_i = \sum_{k=1}^{L} \alpha_k \varphi_k(x_i)
\]

where \(\alpha_k \in \mathbb{R}^3\) is the weight vector connecting the \(k\)th hidden node to the three output nodes, \(L\) is the number of the hidden nodes. In the output layer, the Softmax function \([21]\) is employed to normalize each of the elements in \(y_i\) into range \([0, 1]\):

\[
o_i = \text{Softmax}(y_i)
\]

where \(o_i \in \mathbb{R}^3\) is the normalized actual output vector for \(x_i\). We select the output node with the maximal output value as the action node. Based on the selected action node, we adjust the local acceleration coefficient \(c_l\) as:

\[
c_l = \begin{cases} 
  c_l \times (1 + \Delta) & \text{(If action node denotes increasing } c_l) \\
  c_l \times (1 - \Delta) & \text{(If action node denotes decreasing } c_l) \\
  c_l & \text{(If action node denotes keeping } c_l) 
\end{cases}
\]

where \(\Delta\) is the change of \(c_l\). The three cases in (13) represent increasing \(c_l\), decreasing \(c_l\), and keeping \(c_l\) unchanged, respectively. The adjustment range of \(c_l\) is \([0, C]\), where \(C\) is the constriction coefficient set with a constant \([30, 31]\) capable of controlling \(c_l\) and the global acceleration coefficient \(c_g\) interpreted as the weights of pulling each individual towards the local learning target and global learning target, respectively. Accordingly, \(c_g\) is calculated by:

\[
c_g = C - c_l
\]

Following the particle swarm optimization algorithm, which leverages velocity and position updates \([30]\), in this article \(x_i\) is repositioned as:

\[
v_i(t + 1) = w \times v_i(t) + c_g \times (Gbest(t) - x_i(t)) + c_i \times (Lbest_i(t) - x_i(t))
\]

\[
x_i(t + 1) = x_i(t) + v_i(t + 1)
\]

where \(w\) is the inertia weight, \(v_i(t)\) is the velocity of \(x_i(t)\) at time \(t\), \(Lbest_i(t)\) and \(Gbest(t)\) are the local learning target and the global learning target of \(x_i(t)\). Once the position of \(x_i\) in the decision space changes, we update its multi-objective fitness values.
D. Preliminary Weight Update

Recently, the reinforcement learning has been utilized to solve many optimal control problems [32]–[34]. For instance, it makes great success in the famous AlphaGo through the interactions with environments [35]. As one of representative learning paradigms, the reinforcement learning enables a model to learn strategies by receiving environmental rewards or punishments (environmental interactions) [36]. In this article, through rewarding or punishing the network by the reinforcement learning, we can obtain the expected output of PSGN. Based on the expected output, the network can be guided to learn the actions that should be taken in each environment. Specifically, to assess the evolution effect of each individual that can reflect the environmental feedback as useful evolution information, we define comprehensive evolution criteria considering non-dominant relationships and crowded distances:

1) After updating the position and fitness values of an individual input, its historical personal best position changes.
2) The input individual is a dominated solution and it becomes a non-dominated solution after updating its position and fitness values.
3) The individual is still a non-dominated solution and its crowded distance increases after updating the position and fitness values.

If at least one of the evolution criteria is satisfied (i.e., receiving progressive effect), the action node that represents the determined action in the output layer will obtain a positive feedback for the individual input. Otherwise (i.e., receiving retrogressive effect), it will obtain a negative feedback. Accordingly, the action node is rewarded by setting its expected output to 1 on obtaining a positive feedback. Conversely, the action node is punished by setting its expected output to 0 on obtaining a negative feedback. Besides, the expected output of other two output nodes (non-action nodes) is set with their actual output. The Softmax function is then utilized to normalize the expected output of each individual. Based on the normalized actual output (i.e., each row in $Y$) in (11), we can obtain the expected output in an acceptable training error. Accordingly, based on the obtained $\beta'$ by incremental ELM in (3), the actual output $Y \in \mathbb{R}^{N \times 3}$ of PSGN is calculated as:

$$Y = H' \beta'$$

where $H' \in \mathbb{R}^{N \times (L+K)}$ is the output matrix of the hidden layer for $N$ individuals input, $\beta' \in \mathbb{R}^{(L+K) \times 3}$ is the output weight matrix updated, $L$ is the number of hidden nodes before growing the current batch of hidden nodes. Through the use of the Softmax function for the network output of each individual input (i.e., each row in $Y$) in (11), we can obtain the normalized actual output $O$ of the population. Based on the obtained $O$, the root mean squared error can be recalculated by (17) as the training error. The above steps are repeated until the training error is acceptable (less than or equal to an error threshold $\epsilon$). The pseudo-code of growing hidden nodes and calculating the output weights and training error is presented in Algorithm 1.

E. Efficient Adjustment of Hidden Nodes and Output Weights

1) Growing Hidden Nodes: Updating output weights preliminarily by the ELM [17] may not approximate the expected output in an acceptable training error, affecting the performance of PSGN on directing particle search. Since abundant experiment evidence as well as overwhelming experience indicates that increasing network width can improve network approximation ability [21], we consider growing PSGN hidden nodes. Based on the normalized actual output $O \in \mathbb{R}^{N \times 3}$ and expected output $T \in \mathbb{R}^{N \times 3}$ of the population, the training error $E$ of PSGN is first calculated as:

$$E = \|O - T\|$$

where $O = [o_1, o_2, ..., o_N]$ and $\|O - T\|$ represents the root mean squared error of $O - T$.

2) Pruning Hidden Nodes: As unilateral increment of hidden nodes may introduce some redundancy, we consider pruning some insignificant hidden nodes. For the sake of finding insignificant hidden nodes, we define the significance of a hidden node with the contribution of the node to the output of PSGN averaged over all the input individuals. Specifically, the significance $S_j$ of the $j$th hidden node is defined as:

$$S_j = \|\alpha_j\|_2 \left(\sum_{i=1}^N \frac{\varphi_j(x_i)}{N}\right)^{1/2}$$
Algorithm 2: PruningPSGN

Input: The PSGN to be pruned, the population $P$, the number of the hidden nodes $L$.

Output: The pruned PSGN.

1 Sort all hidden nodes in ascending order;
2 for $j = 1 : L$ do
3    if the subspace only contains the $j$th node then
4        continue;
5    end
6    Calculate the training error $E$ without the $j$th node;
7    if $E \leq \epsilon$ (error threshold) then
8        Remove the $j$th hidden node;
9    else
10       break;
11    end
12 end
13 return PSGN

where $\alpha_j \in \mathbb{R}^3$ is the weight vector connecting the $j$th hidden node to the three output nodes, $\varphi_j(x_i) \in \mathbb{R}$ is the response of the $j$th hidden node to $x_i$, $N$ is the number of individuals in the population, and $\|\alpha_j\|_2$ is the 2-norm of $\alpha_j$. Based on the significance of each hidden node, the hidden nodes of PSGN are sorted in ascending order. When the training error is smaller than an error threshold $\epsilon$, we remove hidden nodes of low significance one by one according to the order while ensuring at least one node in each objective subspace. On the one hand, removing hidden nodes of low significance can generate a compact architecture of PSGN without obviously deteriorating the network training accuracy. On the other hand, ensuring at least one node in each objective subspace contributes to the even distribution of hidden nodes on the whole in objective space. Thus, the actual output $Y_p \in \mathbb{R}^{N \times 3}$ of PSGN after pruning the $j$th hidden node is calculated as:

$$Y_p = Y - Y_j$$  \hspace{1cm} (20)

where $Y_j \in \mathbb{R}^{N \times 3}$ is the actual network output caused by the $j$th hidden node for $N$ individuals input, that is, $Y_j = [\alpha_j \varphi_j(x_1), \alpha_j \varphi_j(x_2), \ldots, \alpha_j \varphi_j(x_N)]$. The Softmax function is used to normalize the network output of each individual input (i.e., each row in $Y_p$) and the normalized actual output $O$ of the population can be obtained. Similar to the process of growing hidden nodes, the root mean squared error can be recalculated by (17) as the residual training error. The pseudo-code of pruning hidden nodes and calculating the residual training error is presented in Algorithm 2.

Algorithm 3: PSGN

Input: The dynamic multi-objective optimization function $F(x, t)$, population size $N$, the number of objective subspaces $K$.

Output: Approximated POF set $POF_s$ and POS set $POS_s$.

1 Initialize the population $P$ randomly, generate $K$ uniformly distributed weight vectors by the weight vector generation method, initialize $K$ hidden nodes, and set the weights between the input layer and hidden layer all with 1, randomly initialize the output weights within $(0, 1]$;
2 $t = 0$;
3 while stopping criterion not met do
4    if Environmental change is detected then
5        Save $POF_t$ and $POS_t$;
6        $t = t + 1$;
7        Reconstruct population $P$;
8        Update archive and ideal point;
9    end
10   Select $G_best$ of the population in (15);
11   for $i = 1 : N$ do
12      Select $L_best$ of $x_i$ in (15);
13      Calculate $y_i$ by (11);
14      Adjust $c_1^x$ and $c_2^x$ by (13) and (14);
15      Update $x_i$ by (16) and its fitness values;
16   end
17   Update archive and ideal point;
18   Calculate the subspace index by (7);
19   Calculate the expected output $T$ of the population;
20   Initialize $K$ hidden nodes;
21   Update the output weights $\beta$ by (1);
22   PSGN = GrowingPSGN(PSGN, $K$);
23   PSGN = PruningPSGN(PSGN, $P$, $L$);
24 end
25 return $POF_s = \{POF_0, POF_1, \ldots, POF_t\}$,

with a relatively small value of $0.2 \times N$ \[3\], where $N$ is the number of individuals in the population. Considering the correlation between two consecutive environments in a DMOP, the non-dominated solutions of the previous environment may quickly adapt to the current environment. Thus, the left part of the reconstructed population (i.e., $0.8 \times N$) are selected randomly from the non-dominated solutions archived. The pseudo-code of the proposed PSGN is presented in Algorithm 3.

F. Environmental Response

When an environmental change is detected, the population is reconstructed as a response to cope with the change. Specifically, we introduce some random individuals as part of the reconstructed population to deal with the population diversity loss. However, excessive introduction of random individuals will affect the search efficiency, so the number of the random individuals introduced to the reconstructed population is set
generically, we introduce some random individuals as part of the reconstructed population to deal with the population diversity loss. However, excessive introduction of random individuals will affect the search efficiency, so the number of the random individuals introduced to the reconstructed population is set in the population (i.e., $0.8 \times N$) are selected randomly from the non-dominated solutions archived. The pseudo-code of the proposed PSGN is presented in Algorithm 3.

G. Computational Complexity

For a DMOP with $D$ dimensions and the population with $N$ individuals, initializing the population in each new environment requires $O(ND)$. After initializing the population, calculating the objective subspace index requires $O(NK)$, where $K$ is the number of objective subspaces. Obtaining the non-dominated solutions by the non-dominated sorting costs $O(N^2)$ \[1\]. Since the number of the input nodes and that...
of the output nodes in PSGN are $D$ and $3$, respectively, the feedforward calculation of PSGN for all the individuals in the population requires $O(NDL)$, where $L$ is the number of the hidden nodes. Selecting the local learning targets of all the individuals costs $O(K)$. Selecting the local learning targets of all the individuals costs $O(NL)$. Calculating the crowded distance of the archived non-dominated solutions requires $O(MN\log N)$ [6], where $M$ is the number of objectives. For the archive with a storage capacity $A$, the cost of updating the archive is $O(NA)$. Training the output weights of PSGN preliminarily by ELM costs $O(NK)$. After growing hidden nodes, training the output weights of PSGN by incremental ELM costs $O(NL)$. Since $K$, $N$, $M\log N$ and $A$ are smaller than $DL$, the total computational complexity of PSGN is $O(NDL)$.

IV. EXPERIMENTS

A. Benchmark Problems

In our experiments, we adopt fourteen DMOPs from the well-known CEC 2018 dynamic multi-objective optimization benchmark test suite DF [37]. It is worth noting that in the first nine problems (i.e., DF1-DF9) the objective number is two and in the left five problems (i.e., DF10-DF14) the objective number is three. For the problems with the same number of objectives, they are characterized by different dynamics [37]. To simulate real-world DMOPs, the time parameter $t$ of the fourteen problems is defined as $(1/n_t)\lceil\tau/t\rceil$, where $\tau$, $n_t$, and $\tau_t$ refer to the iteration counter, severity of change and frequency of change, respectively. A low $n_t$ implies a severely changing environment and a low $\tau_t$ denotes a fast changing environment. $\lceil\cdot\rceil$ is the floor operator. To study the influence of $n_t$ and $\tau_t$ on DMOPs, the three pairs of parameter settings as commonly chosen in the literature are considered: $(n_t = 5$, $\tau_t = 10)$, $(n_t = 10$, $\tau_t = 10)$, and $(n_t = 10$, $\tau_t = 20)$ [9]. As a result, there are forty-two instances in our experiments. The total number of generations is fixed with $50*\tau_t$, which indicates there are 50 environmental changes in each instance.

B. Competitors and Parameter Settings

In our experiments, six state-of-the-art DMOAs are first selected as the algorithms to be compared, including the dynamic multi-objective evolutionary algorithm based on decision variable classification (DMOEADVC) [38], the knowledge guided transfer strategy for evolutionary dynamic multi-objective optimization (KTS-DMOEA) [39], the predictive dynamic multi-objective EA based on the Kalman filter (MOEA/D-KF) [40], the steady state and generational evolutionary algorithm (SGEA) [41], the transfer learning-based dynamic multi-objective optimization algorithm (Tr-DMOEA) [9], and the evolutionary dynamic multi-objective optimization assisted by a support vector regression predictor (SVR-MOEAD) [8]. Besides, the particle swarm optimization with adaptive learning weights tuned by a multiple-input multiple-output fuzzy logic controller (MFCPSO) [42] is also compared. For easy implementation in dynamics environments, MFCPSO is performed in conjunction with the same environmental response of this article (mentioned in Section III-F). Thus, we compare the proposed PSGN with the seven competitors in our experiments.

For a fair comparison, most of the parameters of the competitors are set according to the original references. The population size of all the algorithms is set to 100 (for two objectives) or 105 (for three objectives), referring to [38]. The archive capacity $A$ equals the population size and the number of the decision variables is set to 10 in all of the experiments. All the algorithms run 20 times independently for each instance. In this article, all the experiments are conducted in a MATLAB R2020a environment using a personal laptop with an Intel Core i7-9750H CPU @ 2.60GHz and 16 GB RAM.

The connections between the input layer and hidden layer of PSGN just play the role of transferring data, so the weights between the first two layers are all set with value 1. $C$ in [14] equals 2.0 referring to [31]. $c_2$ and $c_4$ are both set to 1.0 at the beginning of each environment to balance the learning towards $Gbest$ and $Lbest$. $w$ in $[15]$ is set to 0.6, which has been widely verified that such a setting is beneficial for individual evolution [43]. $\epsilon$ in Algorithms 1 and 2 is set to a small value 0.01 to ensure the approximation capacity [21]. The number of objective subspaces $K$ is set to 20 (for two objectives) or 21 (for three objectives) based on the parameter sensitivity analysis which is detailed later.

C. Performance Metric

As a variant of the inverted generational distance (IGD), mean IGD (MIGD) is defined as the average of the IGD values over the number of environmental changes [1]:

$$MIGD(P^*_t, P_t) = \frac{1}{|T|} \sum_{t \in T} IGD(P^*_t, P_t)$$

where $P^*_t$ is the real POF of the $t$-th environment, $P_t$ is an approximation set of $P^*_t$ obtained by a DMOA, $T$ refers to a set of discrete-time points, and $|T|$ is the number of environmental changes. IGD evaluates the performance of a DMOA comprehensively in terms of convergence and diversity, which can be calculated as:

$$IGD(P^*_t, P^*_t) = \frac{\sum_{v \in P^*_t} d(v, P^*_t)}{|P^*_t|}$$

D. Comparisons between PSGN and Competitors

The MIGD and standard deviation values of the eight algorithms on the fourteen DF problems are reported in Table II. Additionally, the average ranks of the eight competitors are also listed in the table. On each instance of the test problems, the best MIGD value is highlighted in bold. Note that the DF problems with the same number of objectives are characterized
by different dynamics \cite{37}. The Wilcoxon rank-sum test \cite{45} is performed at a commonly used significance level of 5\% to evaluate the statistical significance of differences between PSGN and the seven competitors. In Table \textbf{II} (+), (=) and (-) indicate that PSGN is significantly better, insignificant and significantly worse than the competitor, respectively.

From Table \textbf{II} we observe that PSGN performs best on 30 out of 42 test instances in terms of MIGD, which exhibits the superiority of our algorithm considering convergence and diversity. In particular, PSGN performs significantly better than the seven competitors on all the three test instances of DF1, DF2, DF7, DF8, DF9, DF10, DF11, and DF13. Besides, we observe that PSGN falls slightly behind two competitors and four competitors on only one test instance of DF5 and DF12, respectively, indicating the competitive performance of our algorithm.

PSGN seems slightly inefficient on DF4 compared to KTSDMOEA. In this problem, the length and position of the POS change over time \cite{37}, imposing high demand for the diversity of the population. KTSDMOEA selects valuable knowledge in a knowledge pool, which facilitates the uniform distribution of the population on the POF of a new environment and hence the improvement of the population diversity. Indeed, the use of suitable knowledge in KTSDMOEA contributes to solving DF4. Comparing to the first two instances in DF4, bigger $\tau_t$ (i.e., $\tau_t = 20$) is conducive to generating more non-dominated solutions due to more generations, providing more diverse candidate learning targets through growing hidden nodes in PSGN. Thus, PSGN performs best on the third test instance of DF4.

As for DF6, it is a challenging problem due to knee regions/points and long tails \cite{37}. As a direct result, a small number of non-dominated solutions can be obtained under fast changing environments (i.e., the first two instances of DF6 when $\tau_t = 10$). In such a circumstance, PSGN leaves some room to be desired. As aforementioned, when $\tau_t$ becomes bigger (i.e., $\tau_t = 20$), more non-dominated solutions can be obtained, contributing to the performance improvement of PSGN.

PSGN performs slightly worse than three competitors on the three instances of DF3, but still performs better than other four competitors. That is because the decision variables of DF3 are correlated \cite{37}, which facilitates the better performance of some prediction-based algorithms including MOEA/D-KF, TrDMOEA, and SVR-MOEAD. The POF of DF14 can be degenerated \cite{37}, generating deteriorated distribution of POF in the objective subspaces and hence affecting the performance of PSGN. In general, based on the MIGD values from Table \textbf{II} the proposed PSGN performs better than the seven competitors in handling the DF problems of various dynamics.

Fig. \textbf{3} compares the eight algorithms with the IGD values counted by 50 environmental changes on the fourteen DF problems at the configuration ($\tau_t = 10$ and $n_t = 10$). We observe from Fig. \textbf{3} that most of the curves generated by PSGN are located at the bottom and are smoother than other curves, which exhibits the excellent performance of PSGN to approximate the POFs. Since the decision variables of DF3 are correlated, PSGN performs not as good as some prediction-based algorithms in tracking the real POFs in several early environments. With the increase of iterations, the emergence of high-quality non-dominated solutions contributes to reconstructing the good population and hence improves the performance of PSGN in the following environments. PSGN performs with big fluctuations on DF6 due to knee regions/points and long tails of such a problem, so a small number of high-quality non-dominated solutions can be obtained, causing the unstable performance of PSGN. Although the POF of DF14 is degenerated \cite{37}, the POS can still be predicted, which facilitates the better performance of some prediction-based algorithms in terms of IGD. What we want to point out is that PSGN can achieve the best performance on most of the fourteen DF problems.

To meet the high time demand in solving DMOPs, DMOAs are required to run at low time cost. We compare the eight algorithms in terms of CPU time on the fourteen DF problems. As reported in Table \textbf{III} PSGN is faster than Tr-DMOEA and SVR-MOEAD, yet consumes slightly more time than other five algorithms. As two typical prediction-based algorithms, Tr-DMOEA and SVR-MOEAD rank 4.31, 3.98 in Table \textbf{II}. However, they are much slower than other six algorithms due to a large amount of parameter calculation, i.e., the transfer component analysis in Tr-DMOEA \cite{9} and the exhaustive search in SVR-MOEAD \cite{8}. Among the seven competitors, MFCPSO performs better than other six algorithms based on the average rank 3.79, and it is slower than KTSDMOEA with the average rank 4.17. Based on the analysis of the computational complexity in Section \textbf{III-C}, the feedforward calculation of PSGN, the selection of global/local learning targets and the weight update by ELM/incremental ELM do not consume too much computational resource, ensuring that PSGN can run at low time cost. Moreover, PSGN performs best among the eight algorithms due to its excellent performance as shown in Table \textbf{II}.

E. Ablation Study

To verify the effectiveness of the two components in PSGN (i.e., the components of growing hidden nodes and pruning hidden nodes), we construct two ablation algorithms, each of which discards one of the two components. The first ablation algorithm (PSGN w/o G) discards the component of growing hidden nodes, while the second ablation algorithm (PSGN w/o P) discards the component of pruning hidden nodes, where “w/o” stands for “without”. In the ablation study, PSGN w/o G and PSGN w/o P are compared with PSGN on the fourteen DF problems at the configuration ($\tau_t = 10$ and $n_t = 10$). The MIGD values obtained by the three algorithms are reported in Table \textbf{IV}.

We observe from Table \textbf{IV} that PSGN performs better than PSGN w/o G and PSGN w/o P on the vast majority of the DF problems in terms of MIGD, implying the crucial role of each component to PSGN. To discuss why PSGN performs better than PSGN w/o G, the component of growing hidden nodes not only provides diverse candidate learning targets for the population, but also can improve PSGN approximation capacity to
### Table II

<table>
<thead>
<tr>
<th>Prob. $r_s$, $n_t$</th>
<th>PSGN</th>
<th>DMOA-DVC</th>
<th>KTS-DMOA</th>
<th>MOEA-KF</th>
<th>SGA+T</th>
<th>DMOA-SV</th>
<th>SVR-DMOA</th>
<th>MFPCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10,10$</td>
<td>$5.15E-02$</td>
<td>$1.93E-02$</td>
<td>$2.34E-02$</td>
<td>$3.02E-02$</td>
<td>$2.45E-02$</td>
<td>$3.18E-02$</td>
<td>$2.52E-02$</td>
<td>$2.52E-02$</td>
</tr>
<tr>
<td>$20,10$</td>
<td>$9.32E-02$</td>
<td>$1.70E-01$</td>
<td>$1.20E-01$</td>
<td>$2.99E-01$</td>
<td>$2.14E-01$</td>
<td>$2.64E-01$</td>
<td>$2.30E-01$</td>
<td>$2.25E-01$</td>
</tr>
<tr>
<td>$30,10$</td>
<td>$1.12E-01$</td>
<td>$1.37E-01$</td>
<td>$8.57E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
</tr>
<tr>
<td>$40,10$</td>
<td>$1.46E-01$</td>
<td>$6.49E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
</tr>
<tr>
<td>$50,10$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
</tr>
<tr>
<td>$60,10$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
</tr>
</tbody>
</table>

### Table III

**Running Time of Eight Algorithms on Fourteen DF Problems (Unit: Seconds)***

<table>
<thead>
<tr>
<th>Prob. $r_s$, $n_t$</th>
<th>PSGN</th>
<th>DMOA-DVC</th>
<th>KTS-DMOA</th>
<th>MOEA-KF</th>
<th>SGA+T</th>
<th>DMOA-SV</th>
<th>SVR-DMOA</th>
<th>MFPCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10,10$</td>
<td>$5.15E-02$</td>
<td>$1.93E-02$</td>
<td>$2.34E-02$</td>
<td>$3.02E-02$</td>
<td>$2.45E-02$</td>
<td>$3.18E-02$</td>
<td>$2.52E-02$</td>
<td>$2.52E-02$</td>
</tr>
<tr>
<td>$20,10$</td>
<td>$9.32E-02$</td>
<td>$1.70E-01$</td>
<td>$1.20E-01$</td>
<td>$2.99E-01$</td>
<td>$2.14E-01$</td>
<td>$2.64E-01$</td>
<td>$2.30E-01$</td>
<td>$2.25E-01$</td>
</tr>
<tr>
<td>$30,10$</td>
<td>$1.12E-01$</td>
<td>$1.37E-01$</td>
<td>$8.57E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
<td>$3.84E-02$</td>
</tr>
<tr>
<td>$40,10$</td>
<td>$1.46E-01$</td>
<td>$6.49E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
<td>$9.51E-02$</td>
</tr>
<tr>
<td>$50,10$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
<td>$1.25E-01$</td>
</tr>
<tr>
<td>$60,10$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
<td>$1.79E-01$</td>
</tr>
</tbody>
</table>

---

**F. Parameter Sensitivity Analysis**

As a key parameter of PSGN that affects the initial node centers in each environment, the number of objective subspaces $K$ is investigated. Following the weight vector generation method [1, 2], we set $K$ from 2 to 100 and from 3 to 105 when the objective numbers are 2 and 3, respectively. The minimum of $K$ equals the smallest number of hidden nodes in objective space. This component is designed mainly to reduce the architecture complexity of PSGN and hence the computational complexity, without obviously reducing its accuracy. Additionally, hidden node(s) in one objective space can also be maintained after pruning. Thus, PSGN w/o P performs worse than PSGN but better than PSGN w/o G in terms of MIGD.
local learning targets. When $K$ increases to about 20, more number of diverse hidden node centers can be selected in the objective space, contributing to the improvement of the population diversity. However, the further increase of $K$ causes fewer individuals in some objective subspaces due to the limited size of the population, which affects the exploitation in these objective subspaces adversely. As for DF4, the length and position of its POS change over time $t$, which imposes high demand for the diversity of the population to adapt to the change of POS. With the increase of $K$, more number of diverse hidden node centers can be selected as local learning targets, improving the population diversity and hence the performance of PSGN. Based on the above investigation, we recommend to set $K$ to be 20 for two objectives or 21 for three objectives considering the number of weight vectors in the space of two objectives or three objectives.

V. CONCLUSIONS

In this article, a particle search guidance network named PSGN is proposed to tackle DMOPs. PSGN is capable of guiding the search actions of each individual input including learning target selection and acceleration coefficient control. The reinforcement learning is introduced to obtain the expected network output by rewarding or punishing PSGN, enabling the network to learn the actions that should be taken for different iterations in each environment, no matter which dynamic occurs. Thus, PSGN is capable of tackling DMOPs of

### Table IV

<table>
<thead>
<tr>
<th>Prob.</th>
<th>PSGN w/o G</th>
<th>PSGN w/o P</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF1</td>
<td>1.22E-02 ±6.03E-04</td>
<td>1.50E-02 ±7.53E-04 (+)</td>
</tr>
<tr>
<td>DF2</td>
<td>1.25E-02 ± 6.82E-04</td>
<td>1.80E-02 ±7.15E-04 (+)</td>
</tr>
<tr>
<td>DF3</td>
<td>7.34E-02 ±3.45E-02</td>
<td>7.73E-02 ±3.69E-02 (+)</td>
</tr>
<tr>
<td>DF4</td>
<td>1.05E-01 ±2.75E-03</td>
<td>1.33E-01 ±5.99E-03 (+)</td>
</tr>
<tr>
<td>DF5</td>
<td>1.20E-02 ±8.18E-04</td>
<td>3.04E-02 ±1.73E-03 (+)</td>
</tr>
<tr>
<td>DF6</td>
<td>2.59E+00 ±4.63E-01</td>
<td>3.79E+00 ±5.24E-01 (+)</td>
</tr>
<tr>
<td>DF7</td>
<td>2.69E-02 ±1.94E-02</td>
<td>2.59E-02 ±5.76E-03 (+)</td>
</tr>
<tr>
<td>DF8</td>
<td>9.37E-03 ±4.02E-04</td>
<td>1.27E-02 ±5.95E-04 (+)</td>
</tr>
<tr>
<td>DF9</td>
<td>5.36E-02 ±4.75E-03</td>
<td>7.61E-02 ±5.87E-03 (+)</td>
</tr>
<tr>
<td>DF10</td>
<td>8.12E-02 ±6.64E-03</td>
<td>8.72E-02 ±2.23E-03 (+)</td>
</tr>
<tr>
<td>DF11</td>
<td>5.65E-02 ±1.21E-03</td>
<td>9.98E-02 ±4.55E-03 (+)</td>
</tr>
<tr>
<td>DF12</td>
<td>8.08E-02 ±8.58E-02</td>
<td>1.10E-01 ±5.56E-03 (+)</td>
</tr>
<tr>
<td>DF13</td>
<td>1.36E-01 ±4.74E-03</td>
<td>1.67E-01 ±5.37E-03 (+)</td>
</tr>
<tr>
<td>DF14</td>
<td>2.59E-01 ±4.09E-02</td>
<td>1.39E+00 ±2.16E-01 (+)</td>
</tr>
</tbody>
</table>

Fig. 3. IGD values of eight algorithms on fourteen DF problems at configuration ($\tau_t = 10$ and $n_t = 10$).

On the whole the MIGD value in Fig. 4 decreases when $K$ increases from the minimum (i.e., 2 or 3) to about 20, and the value becomes bigger with the further increase of $K$ except DF4. Based on [10], hidden node centers near to the corresponding weight vectors have high selection probabilities (for exploitation in each objective subspace), and hence a small $K$ causes the fact that in the whole objective space less number of diverse hidden node centers can be ultimately selected as
various dynamics. Additionally, we adjust PSGN hidden nodes efficiently and update the output weights in an incremental learning way, so as to solve DMOPs at low time cost. In our experiments, we compare the proposed PSGN with seven state-of-the-art competitors on the DF benchmark test suite. Experimental results demonstrate that the proposed PSGN significantly improves the performance of solving DMOPs. It is worth noting that the proposed PSGN is capable of directing individuals’ search through learning in each environment, so PSGN can handle DMOPs under periodic, nonperiodic, linearly changing, and nonlinearly changing environments. Moreover, neither historical environmental information nor prior knowledge is required for training the network. Thus, PSGN is applicable to a broad range of real-world DMOP scenarios in principle. In the future, we will examine our algorithm on real-world DMOPs [46], [47]. In addition, it is also meaningful to investigate the applicability of our algorithm to dynamic constrained multiobjective optimization problems [48] and dynamic many-objective optimization problems [49] in the future.

REFERENCES


Yinan Guo (Member, IEEE) received the B.Sc. degree in automation and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, in 1997 and 2003, respectively. She is currently a Professor of Computational Intelligence and Machine Learning with the School of Information and Control Engineering, China University of Mining and Technology. Her current research interests include computation intelligence in dynamic and uncertain optimization and its applications in scheduling and path planning.

Shengxiang Yang (Senior Member, IEEE) received the Ph.D. degree from Northeastern University, Shenyang, China, in 1999. He is currently a Professor of Computational Intelligence and the Deputy Director of the Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, U.K. He has over 430 publications with an H-index of 72 according to Google Scholar. His current research interests include evolutionary computation, swarm intelligence, artificial neural networks, data mining and data stream mining, and relevant real-world applications. He serves as an Associate Editor/Editorial Board Member of a number of international journals, such as the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS, INFORMATION SCIENCES, and ENTERPRISE INFORMATION SYSTEMS.

Yaochu Jin (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in automatic control from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree in neuroinformatics from Ruhr-University Bochum, Bochum, Germany, in 2001. He joined the School of Engineering, Westlake University, Hangzhou, China, in October 2023. Prior to that, he was an Alexander von Humboldt Professor of AI with the Faculty of Technology, Bielefeld University, Bielefeld, Germany, and a Distinguished Chair of Computational Intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K. He was a Finland Distinguished Professor in Finland and a Changjiang Distinguished Visiting Professor in China. His main research interests include data-driven evolutionary optimization, trustworthy machine learning, multiobjective evolutionary learning, and evolutionary developmental systems.

Dr. Jin is the recipient of the 2018, 2021, and 2023 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, and the 2015, 2017, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award. He was named “Highly Cited Researcher” from 2019 to 2023 by Clarivate Analytics. He is currently the Editor-in-Chief of Complex & Intelligent Systems. He is a member of Academia Europaea.